Dynamic Programming (Chapter 6 in *Interactive Operations Research with Maple: Methods and Models*)

Mahmut Parlar

June 2, 2000

ABSTRACT This document uses the “Springer Verlag Conference” Style. Replace this text with your own abstract.

Contents

List of Tables

List of Figures

1
Dynamic Programming

"*Remember, today is the first day of the rest of your life.*"
Popular mantra—*circa* 1970

1.1 Introduction

Dynamic programming (DP) is a simple yet powerful approach for solving certain types of sequential optimization problems. Most real-life decision problems are sequential (dynamic) in nature since a decision made now usually affects future outcomes and payoffs. An important aspect of optimal sequential decisions is the desire to balance present costs with the future costs. A decision made now that minimizes the current cost only without taking into account the future costs may not necessarily be the optimal decision for the complete multiperiod problem. For example, in an inventory control problem it may be optimal to order more than the current period's demand and incur high inventory carrying costs now in order to lower the costs of potential shortages that may arise in the future. Thus, in sequential problems it may be optimal to have some "short-term pain" for the prospect of "long-term gain."

Dynamic programming is based on relatively few concepts. The **state** variables of a dynamic process completely specify the process and provide information on all that needs to be known in order to make a decision. For example, in an inventory control problem the state variable x_t may be the inventory level of the product at the start of period t. Additionally, if the supplier of the product is not

always available, then the supplier's availability status may also be another state variable describing the inventory system.

In the inventory control example, ordering decisions are made at certain times. Such times are called **stages** and they may be discrete or continuous. If the inventory is managed using a periodic-review system where decisions are made, say, once every week, then the stage variable would be discrete and it may be sequentially numbered by the integers $t = 0, 1, 2, \ldots$. If, however, a continuous-review (transactions reporting) system is in place, then the order decisions may be made at any time and the stage variable would be continuous.

Having defined the state and stage variables, we now introduce the concept of a **decision**, which is an opportunity to change the state variable. For example, if the inventory level at time t is x_t units, then ordering u_t units is a decision that increases the current inventory to $x_t + u_t$ units. The decision to order u_t units in period t results in a **cost** in the current period and depending on the demand w_t, period t may end with negative or positive inventory resulting in shortages or surpluses that may give rise to additional costs for the current period t. If the process continues for, say, N periods into the future, the optimization problem is to find the best order policy that will minimize the total cost of the inventory system. In general, the total cost incurred over N periods can be written as $\sum_{t=0}^{N-1} L_t(x_t, u_t, w_t) + L_N(x_N)$ where $L_t(\cdot)$ is the cost incurred in period t.

The processes that are studied by dynamic programming pass from stage to stage and thus the states undergo a **transformation** represented by the equation $x_{t+1} = f(x_t, u_t, w_t)$. In the inventory example, after a demand realization of w_t units in period t, the next period's state is computed from $x_{t+1} = x_t + u_t - w_t$ where the transformation function takes the form $f(x_t, u_t, w_t) = x_t + u_t - w_t$. Depending on the nature of the demand w_t, the transformation may be deterministic or stochastic. In the former case, the next period's state variable x_{t+1} is known with certainty, but in the latter case x_{t+1} is a random variable whose distribution can be obtained as a function of x_t, u_t, and w_t.

Finally, we define an optimal **policy** as a decision rule that computes an optimal decision u_t for each conceivable value of the state variable x_t at stage $t = 0, 1, \ldots, N-1$, i.e., $\pi = [\mu_0(x_0), \mu_1(x_1), \ldots, \mu_{N-1}(x_{N-1})]$. Using the optimal policy π, the optimal decision is computed from $u_t = \mu_t(x_t)$. For example, in the inventory problem, an optimal policy may be shown to be of the form

$$\mu_t(x_t) = \begin{cases} S_t - x_t & \text{if } x_t < S_t, \\ 0 & \text{otherwise} \end{cases}$$

where S_t is the "base-stock" level computed using the problem parameters [Sca60]. At any stage, the optimal decision (a number) is computed using the above policy. If at the start of period 4 we have $x_4 = 3$ and, say, $S_4 = 10$, then the optimal order quantity u_4 in period 4 is obtained as $u_4 = \mu_4(3) = 10 - 3 = 7$ units.

Using the "principle of optimality"[1] that was first enunciated by Richard Bellman [Bel52], [Bel57, p. 83], dynamic programming is used to "divide and conquer" a sequential optimization problem by decomposing it into a series of smaller problems. Once the smaller problems are solved, they are combined to obtain the solution of the complete problem.

For example, consider a three-period inventory problem with $t = 0$ as the beginning of December, $t = 1$ as the beginning January and $t = 2$ as the beginning February. Using the "backward-recursion" approach of DP, we would first find the optimal solution for February—the last stage—and compute the optimal decision $u_2 = \mu_2(x_2)$ as a function of any possible value of the inventory x_2 at the start of February. With the information on the optimal decision available, we would then compute the minimum cost $V_2(x_2)$ for February. Stepping back one month, the optimal policy $\mu_1(x_1)$ for January would be found by adding the costs for that month to the minimum cost $V_2(x_2)$ that was found for February. This solution would then be used to compute the minimum cost $V_1(x_1)$ ("cost to go") for periods 1 and 2 (January and February). Finally, the optimal solution for December would be computed taking into account the costs for that month plus the cost to go $V_1(x_1)$ for the remaining months of January and February. Once the optimal policy is found for each month as a function of the entering inventory for that month, as time progresses and state variables x_t are observed, actual decisions would be computed using the optimal policy functions $\mu_t(x_t)$, $t = 0, 1, 2$.

As we indicated, the three-month inventory control example uses the backward-recursion approach of DP. For an interesting discussion of a puzzle and a mathematical proof using the backward approach, see Nemhauser [Nem66, pp. 19–22]. When the process studied is deterministic, it may also be possible to use the "forward-recursion" approach, which can sometimes be more intuitive. For details of the forward approach, we refer the reader to Bertsekas [Ber95, p. 52], Cooper and Cooper [CC81, p. 114–122], Danø [Dan75, pp. 150–155] and Larson and Casti [LC78, p. 233].

Sequential optimization problems with discrete stages can be conveniently formulated using the state-equation approach as in Bertsekas [Ber95]. The state transformation of the basic problem is represented by a discrete-time dynamic system

$$x_{t+1} = f_t(x_t, u_t, w_t), \quad t = 0, 1, \ldots, N-1 \tag{1.1}$$

where x_t is the state (with x_0 a given constant), u_t is the control and w_t is the uncontrollable exogenous disturbance parameter that may be deterministic or random. There may be constraints on the control variables such as $u_t \geq 0$ so that u_t may have to be an element of a constraint space $\mathcal{C}_t$. When the problem is deterministic, there may also be constraints on the state variables so that x_t may have to be an element of the constraint space $\mathcal{S}_t$. If $L_t(x_t, u_t, w_t)$ is the cost incurred

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

during period t, then the objective would be to minimize

$$\sum_{t=0}^{N-1} L_t(x_t, u_t, w_t) + L_N(x_N), \tag{1.2}$$

which is the total cost function for the N-stage sequential problem. Note that if the disturbance w_t is a random variable, then the proper statement of the objective would involve expectations with respect to all the random variables, i.e.,

$$E_{w_0, w_1, \ldots, w_{N-1}} \sum_{t=0}^{N-1} L_t(x_t, u_t, w_t) + L_N(x_N).$$

To solve the sequential optimization problems given by (1.1)–(1.2), we will first define the **cost to go** (or **value**) function $V_t(x_t)$ as the minimum (expected) cost that can be obtained by using an optimal sequence of decisions for the remainder of the process starting from an arbitrary state x_t in an arbitrary stage $t = 0, 1, \ldots, N-1, N$. This function can be written as

$$\begin{aligned} V_t(x_t) &= \min_{u_t, \ldots, u_{N-1}} \sum_{i=t}^{N-1} L_i(x_i, u_i, w_i) + L_N(x_N), \quad t = 0, 1, \ldots, N-1 \\ V_N(x) &= L_N(x_N) \end{aligned}$$

where the decision u_i must be chosen from the feasible set $\mathcal{C}_i$, $i = t, \ldots, N-1$. Using the separability properties of this additive cost function and the principle of optimality, it can be shown that (see, for example, Bertsekas [Ber95, p. 19], Nemhauser [Nem66, pp. 28–31], Larson and Casti [LC78, pp. 45–47]) the value function $V_t(x_t)$ is the solution of a functional equation given as

$$\begin{aligned} V_t(x_t) &= \min_{u_t \in \mathcal{C}_t} \left\{ L_t(x_t, u_t, w_t) + V_{t+1}[f_t(x_t, u_t, w_t)] \right\}, \; t = 0, 1, \ldots, N-1 \\ V_N(x_N) &= L_N(x_N). \end{aligned}$$

We note here that when w_t is random, we minimize the expected value of the term inside the brackets, i.e., $E_{w_t} \{ L_t(x_t, u_t, w_t) + V_{t+1}[f_t(x_t, u_t, w_t)] \}$, where, of course, the expectation is taken with respect to the probability distribution of w_t. In this case $V_t(x_t)$ is defined as the minimum *expected* cost to go for stages $t, t+1, \ldots, N$.

If the objective function is in the multiplicative form, i.e., if we are trying to optimize $\prod_{t=0}^{N} L_t(x_t, u_t, w_t)$, then the functional equations for the value function are obtained as

$$\begin{aligned} V_t(x_t) &= \min_{u_t \in \mathcal{C}_t} \left\{ L_t(x_t, u_t, w_t) \cdot V_{t+1}[f_t(x_t, u_t, w_t)] \right\}, \; t = 0, 1, \ldots, N-1 \\ V_N(x_N) &= \min_{u_N \in \mathcal{C}_N} L_N(x_N, u_N, w_N). \end{aligned}$$

For details of this problem, see Larson and Casti [LC78, pp. 48–49] and Nemhauser [Nem66, p. 37]. The multiplicative objective was originally used by Bellman and Dreyfus [BD58] to model some types of reliability problems.

We will now discuss some examples of sequential decision problems and present their analytic solution using Maple. We will see that Maple's ability to manipulate symbolic expressions makes it an ideal tool for extracting the optimal policies for these problems.

1.2 Stagecoach Problem

Our first example is a simple but illustrative deterministic dynamic programming problem that is known in the operations research literature as the "stagecoach problem." It deals with a hypothetical 19th-century stagecoach company that transports passengers from California to New York. Although the starting point (California) and the destination (New York) are fixed, the company can choose the intermediate **states** to visit in each **stage** of the trip.

We assume that the trip is completed in four stages (legs) where stage 1 starts in California, stage 2 starts in one of three states in the Mountain Time Zone (say, Arizona, Utah or Montana), stage 3 starts in one of three states in the Central Time Zone (say, Oklahoma, Missouri or Iowa) and stage 4 starts in one of two states in the Eastern Time Zone (North Carolina or Ohio). When stage 4 ends, the stagecoach reaches New York, which is the final destination.

Since in those days travel by stagecoach was rather dangerous because of attacks by roaming criminals, life insurance was offered to the traveling passengers. Naturally, the cost of the insurance **policy** was higher on those portions of the trip where there was more danger. The stagecoach company thus faced the problem of choosing a route that would be cheapest and thus safest for its passengers.

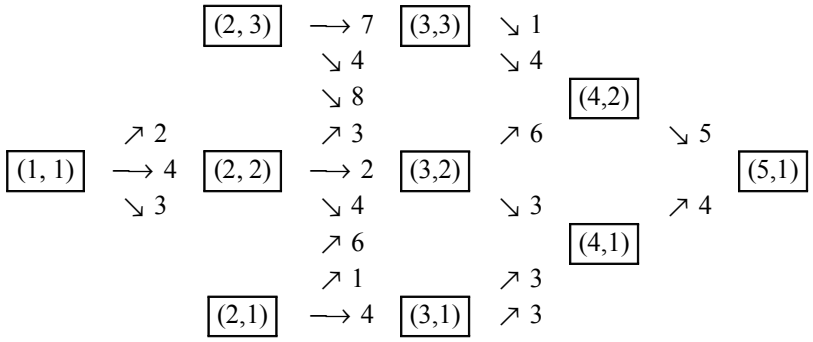

TABLE 1.1. Data for the stagecoach problem.

The nodes in the diagram in Table 1.1 are denoted by (i, j) where i corresponds to a stage ($i = 1, 2, \ldots, 5$) and j corresponds to a state in state i. Given the state j in stage i, the decision to travel to state k in the next stage $i + 1$ results in a cost (i.e., insurance premium) of $c_{ij}(k)$. These costs are indicated next to the arrows corresponding to each decision. For example, $c_{11}(2) = 4$ and $c_{21}(3) = 6$. The problem is to find the route that results in the minimum cost.

For this problem, the value function V_{ij} is the minimum cost from state (i, j) to the final state (5, 1) using the optimal policy. Thus, the dynamic programming recursive equations are written as

$$V_{51} = 0$$

$$V_{ij} = \min_k \{c_{ij}(k) + V_{i+1,k}\}$$

where the expression inside the parenthesis is minimized by a suitable choice of the decision variable k.

In the Maple worksheet that follows we enter the cost data as lists, e.g., `c[1,1] :=[3,4,2]`. This means that if we are now in state (1, 1), the cost of going to states (2,1), (2,2) and (2,3) are 3, 4 and 2, respectively.

```
> restart: # StageCoach.mws
> c[1,1]:=[3,4,2];
```

$$c_{1,1} := [3, 4, 2]$$

```
> c[2,1]:=[4,1,6]; c[2,2]:=[4,2,3];
  c[2,3]:=[8,4,7];
```

$$c_{2,1} := [4, 1, 6]$$

$$c_{2,2} := [4, 2, 3]$$

$$c_{2,3} := [8, 4, 7]$$

```
> c[3,1]:=[3,3]; c[3,2]:=[3,6]; c[3,3]:=[4,1];
```

$$c_{3,1} := [3, 3]$$

$$c_{3,2} := [3, 6]$$

$$c_{3,3} := [4, 1]$$

```
> c[4,1]:=[4]; c[4,2]:=[5];
```

$$c_{4,1} := [4]$$

$$c_{4,2} := [5]$$

For stage 5, the boundary condition for the value function is trivially obtained as 0.

```
> i:=5;
```

$$i := 5$$

```
> V[i,1]:=0;
```

$$V_{5,1} := 0$$

Stage 4 computations are also trivial since, once in stage 4, the stagecoach must travel to the last state.

```
> i:=4;
```

$$i := 4$$

```
> for j from 1 to 2 do
> V[i,j]:=min(seq(c[i,j][k]+V[i+1,k],k=1..1));
> TC[i,j]:=[seq(c[i,j][k]+V[i+1,k],k=1..1)];
> for k from 1 to 1 do
  u[i,j,k]:=is(TC[i,j][k]=V[i,j]) od od;
```

$$V_{4,1} := 4$$

$$TC_{4,1} := [4]$$

$$V_{4,2} := 5$$

$$TC_{4,2} := [5]$$

```
> for j from 1 to 2 do for k from 1 to 1 do
  print([i,j,k,u[i,j,k],TC[i,j][k]]) od od;
```

$$[4,\ 1,\ 1,\ \mathit{true},\ 4]$$

$$[4,\ 2,\ 1,\ \mathit{true},\ 5]$$

For stage 3, we find the value function for each state and the corresponding optimal decision. For example, $[3, 1, 1, true, 7]$ means that in state $[3, 1]$ it is optimal to choose $k = 1$, which results in a minimum overall cost of 7. Thus, the word $true$ next to a decision (the third component in the list) implies the optimality of that decision. On the other hand, the word $false$ implies that the corresponding decision is not optimal.

```
> i:=3;
```

$$i := 3$$

```
> for j from 1 to 3 do
> V[i,j]:=min(seq(c[i,j][k]+V[i+1,k],k=1..2));
> TC[i,j]:=[seq(c[i,j][k]+V[i+1,k],k=1..2)];
> for k from 1 to 2 do
  u[i,j,k]:=is(TC[i,j][k]=V[i,j]) od od;
```

$$V_{3,1} := 7$$

$$TC_{3,1} := [7,\ 8]$$

$$V_{3,2} := 7$$

$$TC_{3,2} := [7,\ 11]$$

$$V_{3,3} := 6$$

$$TC_{3,3} := [8, 6]$$

```
> for j from 1 to 3 do for k from 1 to 2 do
  print([i,j,k,u[i,j,k],TC[i,j][k]]) od od;
```

$$[3, 1, 1, \mathit{true}, 7]$$

$$[3, 1, 2, \mathit{false}, 8]$$

$$[3, 2, 1, \mathit{true}, 7]$$

$$[3, 2, 2, \mathit{false}, 11]$$

$$[3, 3, 1, \mathit{false}, 8]$$

$$[3, 3, 2, \mathit{true}, 6]$$

Stage 2 and stage 1 calculations are performed in a similar manner:

```
> i:=2;
```

$$i := 2$$

```
> for j from 1 to 3 do
> V[i,j]:=min(seq(c[i,j][k]+V[i+1,k],k=1..3));
> TC[i,j]:=[seq(c[i,j][k]+V[i+1,k],k=1..3)];
> for k from 1 to 3 do
  u[i,j,k]:=is(TC[i,j][k]=V[i,j]) od od;
```

$$V_{2,1} := 8$$

$$TC_{2,1} := [11, 8, 12]$$

$$V_{2,2} := 9$$

$$TC_{2,2} := [11, 9, 9]$$

$$V_{2,3} := 11$$

$$TC_{2,3} := [15, 11, 13]$$

```
> for j from 1 to 3 do for k from 1 to 2 do
  print([i,j,k,u[i,j,k],TC[i,j][k]]) od od;
```

$$[2, 1, 1, \mathit{false}, 11]$$

$$[2, 1, 2, \mathit{true}, 8]$$

$$[2, 2, 1, \mathit{false}, 11]$$

$$[2, 2, 2, \mathit{true}, 9]$$

$$[2, 3, 1, \mathit{false}, 15]$$

$$[2, 3, 2, \mathit{true}, 11]$$

```
> i:=1;
```

$$i := 1$$

```
> for j from 1 to 1 do
```

```
> V[i,j]:=min(seq(c[i,j][k]+V[i+1,k],k=1..3));
> TC[i,j]:=[seq(c[i,j][k]+V[i+1,k],k=1..3)];
> for k from 1 to 3 do
  u[i,j,k]:=is(TC[i,j][k]=V[i,j]) od od;
```

$$V_{1,1} := 11$$

$$TC_{1,1} := [11,\ 13,\ 13]$$

```
> for j from 1 to 1 do for k from 1 to 3 do
  print([i,j,k,u[i,j,k],TC[i,j][k]]) od od;
```

$$[1,\ 1,\ 1,\ \mathit{true},\ 11]$$

$$[1,\ 1,\ 2,\ \mathit{false},\ 13]$$

$$[1,\ 1,\ 3,\ \mathit{false},\ 13]$$

We can now extract the optimal solution from the results. From the initial state (1,1), it is optimal to travel to (2,1) at a one-stage cost of 3 since we find [1,1,1,*true*,11].[2] Next, from (2,1) it is optimal to go to (3,2) at a cost of 1 since [2,1,2,*true*,8].[3] From (3,2), the stagecoach should travel to (4,1) at a cost of 3, and finally from (4,1) it will go to (5,1) at a cost of 4. Adding the one-stage costs we find $3 + 1 + 3 + 4 = 11$ as we found from [1,1,1,*true*,11].

1.3 Models with a Linear System and Quadratic Cost

Now, consider the sequential problem with the cost function $\sum_{t=0}^{2}(x_t^2 + u_t^2) + x_3^2$ and the system equations $x_{t+1} = x_t + u_t$, $t = 0, 1, 2$ with x_0 given as a constant. In this deterministic problem with quadratic costs and linear system dynamics, it is desirable to bring the state as close as possible to the origin in the cheapest possible way. Deviations from the origin for both the state x_t and the decision u_t are penalized using quadratic terms. We will assume that $x_0 \neq 0$ (otherwise the solution would be trivial) and that there are no constraints on the state and the control.

Although this model with linear system and quadratic cost appears simple, its generalizations have found applications in economics and operations research; see Bensoussan, Hurst and Näslund [BHN74, Chapter 3], Holt, Modigliani, Muth and Simon [HMMS60], Parlar [Par82], Parlar and Gerchak [PG89] and Parlar and Rempała [PR92].

Forming the DP functional equation, we have

$$V_t(x_t) \quad = \quad \min_{u_t} [x_t^2 + u_t^2 + V_{t+1}(x_t + u_t)], \quad t = 0, 1, 2$$

[2] This also indicates that the overall minimum cost is 11.

[3] If the trip had started in state (2,1), the overall minimum cost to New York would be 8.

$$V_3(x_3) \quad = \quad x_3^2.$$

We will now use Maple to solve this functional equation and generate the optimal policy for the problem.

```
> restart: # LQ2.mws
```

We begin by informing Maple that the boundary condition is $V_3(x) = x^2$.

```
> V[3]:=x->x^2;
```

$$V_3 := x \to x^2$$

Now, for period 2 the cost expression that should be minimized is $c_2 = x_2^2 + u_2^2 + V_3(x_2 + u_2)$, i.e., the sum of the current cost and the future minimum cost with $x_3 = x_2 + u_2$ as the state for stage 3.

```
> c[2]:=x^2+u^2+V[3](x+u);
```

$$c_2 := x^2 + u^2 + (x+u)^2$$

Differentiating this expression, equating the result to zero and solving gives the optimal policy in period 2 as $\mu_2 = -\frac{1}{2}x$.

```
> deriv[2]:=diff(c[2],u);
```

$$deriv_2 := 4\,u + 2\,x$$

```
> mu[2]:=solve(deriv[2],u);
```

$$\mu_2 := -\frac{1}{2}\,x$$

Using these results, the value function $V_2(x)$ for period 2 is found as a quadratic function of the state variable.

```
> V[2]:=unapply(subs(u=mu[2],c[2]),x);
```

$$V_2 := x \to \frac{3}{2}\,x^2$$

Repeating the same procedure for the other stages, the optimal decision and the value function is computed explicitly as a function of the state variable in each stage:

```
> c[1]:=x^2+u^2+V[2](x+u);
```

$$c_1 := x^2 + u^2 + \frac{3}{2}\,(x+u)^2$$

```
> deriv[1]:=diff(c[1],u);
```

$$deriv_1 := 5\,u + 3\,x$$

```
> mu[1]:=solve(deriv[1],u);
```

$$\mu_1 := -\frac{3}{5}\,x$$

```
> V[1]:=unapply(subs(u=mu[1],c[1]),x);
```

$$V_1 := x \to \frac{8}{5}x^2$$

```
> c[0]:=x^2+u^2+V[1](x+u);
```

$$c_0 := x^2 + u^2 + \frac{8}{5}(x+u)^2$$

```
> deriv[0]:=diff(c[0],u);
```

$$deriv_0 := \frac{26}{5}u + \frac{16}{5}x$$

```
> mu[0]:=solve(deriv[0],u);
```

$$\mu_0 := -\frac{8}{13}x$$

```
> V[0]:=unapply(subs(u=mu[0],c[0]),x);
```

$$V_0 := x \to \frac{21}{13}x^2$$

The results (to three significant digits) are summarized in the table below.

t	$\mu_t(x)$	$V_t(x)$
0	$-0.615x$	$1.615x^2$
1	$-0.6x$	$1.6x^2$
2	$-0.5x$	$1.5x^2$
3		x^2

Thus, if the initial value of the state is, say, $x_0 = 1$, then using the optimal decisions prescribed by the policy

$$\pi = (\mu_0(x), \mu_1(x), \mu_2(x)) = (-0.615x, -0.6x, -0.5x),$$

one would obtain a total minimum cost of $V_0(1) = 1.615$ for the periods 0, 1, 2 and 3. For this case, the numerical values of the optimal decisions are obtained as follows:

t	x_t	u_t
0	1	-0.615
1	0.385	-0.231
2	0.154	$-0.077.$
3	0.077	

Now suppose that the initial decision is not made optimally and instead of $u_0 = -0.615$ we use, say, $u_0 = -0.5$, which brings the system to $x_1 = 1 - 0.5 = 0.5$. What is the best decision now at the start of period 1 given this value of the state? As the principle of optimality states, "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." Thus, in period 1 we make the optimal decision for that period using $u_1 = \mu_1(0.5) = -0.6 \times 0.5 = -0.3$. This is the power and elegance of the

dynamic programming approach: Regardless of what may have transpired in the past, at the start of any period we know exactly what to do since DP makes the optimal policy available for all periods and for any value of the state.

In this example we can see another aspect of the power and flexibility of dynamic programming. Now consider the following case: After applying the correct decision $u_0 = -0.615$, what happens if for some reason there is an unexpected random disturbance on the system and we find $x_1 = 0.4$ (instead of 0.385)? The answer is fortunately very simple. Since we have the optimal policy available to us as $\mu_1(x_1) = -0.6x_1$, we just compute the new (optimal) decision as $u_1 = -0.6 \times 0.4 = -0.24$. Even though the correct initial decision $u_0 = -0.615$ resulted in an unexpected value of the state $x_1 = 0.4$, we can still make the next decision in stage 1 optimally using the optimal policy $\mu_1(x_1)$ for that stage.

Solution as a Nonlinear Programming Problem

We note that the sequential optimization model described above can also be solved as a standard nonlinear programming (NLP) problem with six decision variables $(\mathbf{x}, \mathbf{u}) = (x_1, x_2, x_3, u_0, u_1, u_2)$, the objective function $f(\mathbf{x}, \mathbf{u}) = \sum_{t=0}^{2}(x_t^2 + u_t^2) + x_3^2$ and three equality constraints $h_1 = x_1 - x_0 - u_0 = 0$, $h_2 = x_2 - x_1 - u_1 = 0$ and $h_3 = x_3 - x_2 - u_2 = 0$ with x_0 as a given constant. The methods discussed in Chapter **??**, Nonlinear Programming, can be applied and using the method of Lagrange multipliers, the optimal solution can be found.

Writing the Lagrangian as

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) = f(\mathbf{x}, \mathbf{u}) + \sum_{t=1}^{3} \theta_t h_t(\mathbf{x}, \mathbf{u})$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ are the Lagrange multipliers, we obtain the solution for this problem using Maple as follows.

```
> restart: # LQ2NLP.mws
> x[0]:=1:
> f:=sum(x[t]^2+u[t]^2,t=0..2)+x[3]^2:
> h[1]:=x[1]-x[0]-u[0]:
> h[2]:=x[2]-x[1]-u[1]:
> h[3]:=x[3]-x[2]-u[2]:
> X:=seq(x[t],t=1..3):
> U:=seq(u[t],t=0..2):
> L:=f+sum(theta[k]*h[k],k=1..3):
> L_x:=seq(diff(L,x[k]),k=1..3):
> L_u:=seq(diff(L,u[k]),k=0..2):
> L_theta:=seq(diff(L,theta[k]),k=1..3):
> Digits:=3:
```

```
> evalf(solve({L_x,L_u,L_theta},{X,U,theta[1],
  theta[2],theta[3]}));
```

$\{u_2 = -.0769,\ \theta_1 = -1.23,\ x_1 = .385,\ u_1 = -.231,\ \theta_2 = -.462,\ x_2 = .154,$
$x_3 = .0769,\ \theta_3 = -.154,\ u_0 = -.615\}$

Naturally, the solution here is the same as the one we found before. However, note that the NLP solution is in some sense "frozen" because unlike the DP solution it cannot directly deal with a situation where after applying $u_0 = -0.615$ there is an unexpected random disturbance on the system and we find $x_1 = 0.4$ (instead of 0.385). To answer this question, the NLP problem must be re-solved for stages 1, 2 and 3 with $x_1 = 0.4$ as the initial value of the state in stage 1. This shows that the DP approach is, in general, much more powerful than a competing optimization technique since the DP solution is obtained in terms of dynamic policies rather than the static—frozen—decisions produced by the competing techniques.[4]

The simple model described and solved using DP can easily be generalized using Maple. For example, one may assume that there are ideal (i.e., target) levels $\{\hat{x}_t,\ t = 1, 2, \ldots, N\}$ and $\{\hat{u}_t,\ t = 0, 1, \ldots, N-1\}$ for the states and decisions, respectively, and that deviations from these trajectories are to be penalized quadratically at a cost of q_t for the states and r_t for the decisions. If the disturbances also follow a particular trajectory $\{w_t,\ t = 0, \ldots, N-1\}$, then the sequential optimization problem can be written

$$\min \sum_{t=0}^{N-1} \{q_t(x_t - \hat{x}_t)^2 + r_t(u_t - \hat{u}_t)^2\} + q_N(x_N - \hat{x}_N)^2$$

subject to the linear constraints

$$x_{t+1} = a_t x_t + b_t u_t + w_t, \quad t = 0, 1, \ldots, N-1$$

with given a_t and b_t, $t = 0, 1, \ldots, N-1$. Following is a Maple program that automates the solution of this more general problem via DP where $N = 3$ with the arrays `xHat`, `q`, `uHat`, `r`, `a`, `b` and `w` having the obvious meanings.

```
> restart: # LQ2Auto.mws
> N:=3: #Assume N=3
> xHat:=array(0..N,[4,3,2,1]):
  q:=array(0..N,[1,2,1,2]):
> uHat:=array(0..N-1,[1,2,3]):
  r:=array(0..N-1,[2,2,3]):
> a:=array(0..N-1,[1,7,2]):
  b:=array(0..N-1,[4,2,1]):
```

[4] In engineering terminology, nonlinear programming would produce "open-loop" decisions whereas dynamic programming gives rise to "closed-loop" policies; see Bertsekas [Ber95, p. 4] and Kirk [Kir70, pp. 14–16].

```
> w:=array(0..N-1,[2,-3,1]):
> Digits:=3:
> V[N]:=unapply(q[N]*(x-xHat[N])^2,x);
```

$$V_3 := x \to 2\,(x-1)^2$$

```
> for t from N-1 by -1 to 0 do
> c[t]:=q[t]*(x-xHat[t])^2+r[t]*(u-uHat[t])^2
  +V[t+1](a[t]*x+b[t]*u+w[t]
  );
> deriv[t]:=diff(c[t],u);
> mu[t]:=evalf(solve(deriv[t],u));
> V[t]:=
  unapply(evalf(simplify(subs(u=mu[t],c[t]))),x);
> od;
```

$$c_2 := (x-2)^2 + 3\,(u-3)^2 + 2\,(2\,x+u)^2$$

$$deriv_2 := 10\,u - 18 + 8\,x$$

$$\mu_2 := 1.80 - .800\,x$$

$$V_2 := x \to 5.80\,x^2 + 10.4\,x + 14.8$$

$$c_1 := 2\,(x-3)^2 + 2\,(u-2)^2 + 5.80\,(7\,x + 2\,u - 3)^2 + 72.8\,x + 20.8\,u - 16.4$$

$$deriv_1 := 50.4\,u - 56.8 + 162.\,x$$

$$\mu_1 := 1.13 - 3.21\,x$$

$$V_1 := x \to 24.6\,x^2 + .192\,x + 29.8$$

$$c_0 := (x-4)^2 + 2\,(u-1)^2 + 24.6\,(x + 4\,u + 2)^2 + .192\,x + .768\,u + 30.2$$

$$deriv_0 := 791.\,u + 391. + 197.\,x$$

$$\mu_0 := -.494 - .249\,x$$

$$V_0 := x \to 1.12\,x^2 - 6.51\,x + 50.3$$

The results for the optimal policy and the value function for each period $t = 0, \ldots, 3$ are summarized in the following table.

t	$\mu_t(x)$	$V_t(x)$
0	$-0.249x - 0.494$	$1.12x^2 - 6.51x + 50.3$
1	$-3.21x + 1.13$	$24.6x^2 + 0.192x + 29.8$
2	$-0.8x + 1.8$	$5.80x^2 + 10.4x + 14.8$
3		$2(x-1)^2$

It is worth noting that in this more general problem the optimal policy is linear and the value function is quadratic in the state variable. This result would be valid even when the states and the controls are multidimensional and the disturbances are random provided that the cost is quadratic and the system equations are linear with no constraints on the states and controls. For a discussion of this issue, see Bertsekas [Ber95, Section 4.1].

1.3.1 The Infinite-Stage Problem

In many realistic sequential optimization problems arising in business and economics, there may either be a very large number or infinitely many stages. For example, a long-term planning problem involving the next 10 years where a decision has to be made each month would have 120 stages. Although this number is finite, it is large enough so that approximating the problem with an infinite number of stages may be useful in understanding the structure of the optimal policy.

As an example, consider the simple sequential problem discussed above. If we now assume that $N \to \infty$, the problem can be written as

$$\min \sum_{t=0}^{\infty} (x_t^2 + u_t^2)$$

subject to

$$x_{t+1} = x_t + u_t, \quad t = 0, 1, \ldots .$$

For such a problem, the functional equations $V_t(x_t)$ for, say, $t = 100$ and $t = 99$ would be given by

$$V_{100}(x_{100}) = \min_{u_{100}} [x_{100}^2 + u_{100}^2 + V_{101}(x_{101})]$$

and

$$V_{99}(x_{99}) = \min_{u_{99}} [x_{99}^2 + u_{99}^2 + V_{100}(x_{100})].$$

Since there is some regularity in the system equations $x_{t+1} = f_t(x_t, u_t) = x_t + u_t$ and the cost function $L_t = x_t^2 + u_t^2$ (i.e., that t does not appear explicitly in f_t or L_t), under certain conditions the functional equation for the value function can be written as

$$V(x) = \min_{u} \{L(x, u) + V[f(x, u)]\},$$

which can be solved (usually iteratively) for the unknown function $V(x)$.[5]

[5] In order for $V(x)$ to remain finite, three conditions are required: (i) $L(x, u)$ must be bounded for finite x and u, (ii) a state $\bar{x}$ and control $\bar{u}$ must exist such that $L(\bar{x}, \bar{u}) = 0$ and (iii) the specified state $\bar{x}$ must be reachable from any admissible state by applying a finite number of admissible controls; see, Larson and Casti [LC78, p. 222].

Successive Approximations

One of the methods that can be used to solve this problem is known as "successive approximations" or "approximation in function space." The method starts with a guess $V^{(0)}(x)$ for the solution $V(x)$. Using this guess, a new value of the function $V^{(1)}(x)$ is computed from

$$V^{(1)}(x) = \min_u \{L(x,u) + V^{(0)}[f(x,u)]\} \tag{1.3}$$

where the corresponding policy $\mu^{(1)}(x)$ is found as the value of u that minimizes the right-hand-side of (1.3) for each x. The successive approximations method continues in this manner while computing the value function's nth approximation $V^{(n)}$ from $V^{(n-1)}$ using $V^{(n)}(x) = \min_u \{L(x,u) + V^{(n-1)}[f(x,u)]\}$. Here, $\mu^{(n-1)}$ is determined as the value of u that minimizes the right-hand side $L(x,u) + V^{(n-1)}[f(x,u)]$ for each x. This is a relatively easy method to apply using Maple.

```
> restart: # LQInf.mws
> Digits:=4;
```

$$Digits := 4$$

In the finite horizon version of this problem we had observed that the value function was quadratic in the state variable x. Armed with this information, we make the initial guess for the value function as $V^{(0)} = 2x^2$. The following lines automate the successive approximation of the value function and the policy μ.

```
> L:=(x,u)->x^2+u^2;
```

$$L := (x, u) \to x^2 + u^2$$

```
> V[0]:=x->2.*x^2;
```

$$V_0 := x \to 2.\,x^2$$

```
> for n from 0 to 5 do
> c[n]:=(x,u)->L(x,u)+V[n](x+u):
> diff(c[n](x,u),u):
> uApp:=solve(%,u);
> subs(u=uApp,c[n](x,u)):
> V[n+1]:=unapply(simplify(%),x);
> od;
```

$$c_0 := (x, u) \to \mathrm{L}(x, u) + V_n(x + u)$$

$$6.\,u + 4.\,x$$

$$uApp := -.6667\,x$$

$$1.667\,x^2$$

$$V_1 := x \to 1.667\,x^2$$

$$c_1 := (x,\ u) \to \mathrm{L}(x,\ u) + V_n(x + u)$$

$$5.334\,u + 3.334\,x$$

$$uApp := -.6250\,x$$

$$1.625\,x^2$$

$$V_2 := x \to 1.625\,x^2$$

$$c_2 := (x,\ u) \to \mathrm{L}(x,\ u) + V_n(x + u)$$

$$5.250\,u + 3.250\,x$$

$$uApp := -.6190\,x$$

$$1.619\,x^2$$

$$V_3 := x \to 1.619\,x^2$$

$$c_3 := (x,\ u) \to \mathrm{L}(x,\ u) + V_n(x + u)$$

$$5.238\,u + 3.238\,x$$

$$uApp := -.6182\,x$$

$$1.618\,x^2$$

$$V_4 := x \to 1.618\,x^2$$

$$c_4 := (x,\ u) \to \mathrm{L}(x,\ u) + V_n(x + u)$$

$$5.236\,u + 3.236\,x$$

$$uApp := -.6180\,x$$

$$1.618\,x^2$$

$$V_5 := x \to 1.618\,x^2$$

$$c_5 := (x,\ u) \to \mathrm{L}(x,\ u) + V_n(x + u)$$

$$5.236\,u + 3.236\,x$$

$$uApp := -.6180\,x$$

$$1.618\,x^2$$

$$V_6 := x \to 1.618\,x^2$$

We thus see that, after about five iterations, the solution quickly converges to $V(x) = 1.618x^2$ and $\mu(x) = -0.618x$.

A Closed-Form Solution for a More General Cost Function

Now consider a more general problem where we wish to minimize $\sum_{t=0}^{\infty}(Qx_t^2 + Ru_t^2)$ subject to the usual state dynamics $x_{t+1} = x_t + u_t$. Here, deviations from the origin for the state and control variables are penalized differently.

In this case the functional equation for the value function assumes the form $V(x) = \min_u [Qx^2 + Ru^2 + V(x+u)]$. Using the information obtained for the structure of the optimal policy and the form of the value function, we assume that $V(x) = Ax^2$ and attempt to find a closed-form formula for the coefficient A. The symbolic manipulation of the expressions to solve this problem is relegated to Maple, which finds the solution easily.

```
> restart: # LQClosedForm.mws
> V:=x->A*x^2;
```

$$V := x \to A\,x^2$$

```
> c:=Q*x^2+R*u^2+V(x+u);
```

$$c := Q\,x^2 + R\,u^2 + A\,(x+u)^2$$

```
> cu:=diff(c,u);
```

$$cu := 2\,R\,u + 2\,A\,(x+u)$$

```
> usol:=solve(cu,u);
```

$$usol := -\frac{A\,x}{R+A}$$

```
> RHS:=normal(subs(u=usol,c));
```

$$RHS := \frac{x^2\,(A\,Q + R\,A + Q\,R)}{R+A}$$

```
> Asol:=solve(V(x)=RHS,A);
```

$$Asol := \frac{1}{2}\,Q + \frac{1}{2}\sqrt{Q^2 + 4\,Q\,R},\ \frac{1}{2}\,Q - \frac{1}{2}\sqrt{Q^2 + 4\,Q\,R}$$

```
> subs(A=Asol[1],usol);
```

$$-\frac{(\frac{1}{2}\,Q + \frac{1}{2}\sqrt{Q^2 + 4\,Q\,R})\,x}{R + \frac{1}{2}\,Q + \frac{1}{2}\sqrt{Q^2 + 4\,Q\,R}}$$

Note that since $Q - \sqrt{Q^2 + 4QR} < 0$ and since the minimum cost $V(x)$ must be nonnegative, we choose the first solution `Asol[1]` for A in the above computations.

```
> V:=subs(A=Asol[1],V(x));
```

$$V := (\frac{1}{2}\,Q + \frac{1}{2}\sqrt{Q^2 + 4\,Q\,R})\,x^2$$

To summarize, the optimal policy and the value function for this general case are obtained as $\mu(x) = -Ax/(R + A)$ and $V(x) = Ax^2$ where

$$A = \frac{1}{2}\left(Q + \sqrt{Q^2 + 4QR}\right).$$

Naturally, for the special case when $Q = R = 1$, we obtain $A = 1.618$ so that $V(x) = 1.618x^2$ and $\mu(x) = -0.618x$ as we had found previously.

The Model with a Discount Factor

In most practical problems with long planning horizons, it would be important to discount the future cash flows to account for the concept of time value of money. Thus, if $\beta = 1/(1 + r)$ is the discount factor with r as the interest rate and $\beta \in (0, 1)$, then the objective would be to minimize the net present value of the future costs, which would now be written as $\min \sum_{t=0}^{\infty} \beta^t (x_t^2 + u_t^2)$. With this objective, the functional equation for the value function assumes the form $V(x) = \min_u \{L(x, u) + \beta V[f(x, u)]\}$, which can be solved for $V(x)$ in the usual way. Naturally, in this case the value function $V(x)$ would be defined as the minimum *discounted* cost that can be obtained by using an optimal sequence of decisions for the remainder of the process starting from an arbitrary state x in an arbitrary stage.

Although we do not present it here, it would be a simple matter to apply the successive approximation method to solve a more general problem with discounted cost and stationary target values, i.e.,

$$\min \sum_{t=0}^{\infty} \beta^t \{Q(x_t - \hat{x})^2 + R(u_t - \hat{u})^2\}$$

subject to

$$x_{t+1} = x_t + u_t, \quad t = 0, 1, \ldots.$$

For example, assuming $Q = 2$, $\hat{x} = 1$, $R = 3$, $\hat{u} = 1.5$, and $\beta = 0.9$ and starting with the initial guess of $V^{(0)} = 4x^2 + 2x + 60$, after about seven iterations the method converges to the value function $V(x) = 3.55x^2 + 1.15x + 61.6$ and to policy $\mu(x) = 0.642 - 0.516x$.

1.4 Continuous-Time Dynamic Programming

A different type of infinite-stage sequential optimization problem arises when the system evolves continuously over time and a decision must be made at each instant. In this case, even if the decision horizon is finite, the continuous nature of the process results in an infinite number of decisions.

Let us now suppose that the system evolves according to the differential equation $\dot{x}(t) = f[x(t), u(t), t]$, $0 \le t \le T$ with the initial value of the state $x(0) =$

x_0 given as a constant. This differential equation is the continuous-time analogue of the discrete-time state equations that were written as $x_{t+1} = f(x_t, u_t, t)$, $t = 0, 1, \ldots, N-1$ in Section 1.3. We wish to minimize the cost functional $\int_0^T L[x(t), u(t), t]\,dt + S[x(T), T]$, which is the continuous-time analogue of the sum $\sum_{t=0}^{N-1} L_t(x_t, u_t, w_t) + L_N(x_N)$.

Similar to the discrete-time formulation, let us define the value function $V(t, x)$ as the minimum cost that can be obtained starting in state $x(t)$ at time $t \in [0, T]$. The value function can then be written as

$$V(t, x) = \min_u \int_t^T L(x, u, \tau)\,d\tau + S[x(T), T] \tag{1.4}$$

subject to

$$\dot{x} = f(x, u, \tau), \quad x(t) = x$$

with the condition that at the final time $t = T$, we have $V[x(T), T] = S[x(T), T]$.

If we write the integral in (1.4) as

$$V(t, x) = \min_u \left(\int_t^{t+\Delta t} L\,d\tau + \int_{t+\Delta t}^T L\,d\tau + S \right)$$

where Δt is an infinitesimally small interval, then by the principle of optimality we obtain

$$V(t, x) = \min_{\substack{u \\ t \le \tau \le t+\Delta t}} \left[\int_t^{t+\Delta t} L\,d\tau + \left(\min_{\substack{u \\ t+\Delta t \le \tau \le T}} \int_{t+\Delta t}^T L\,d\tau + S \right) \right] \tag{1.5}$$

subject to

$$\dot{x} = f(x, u, \tau), \quad x(t + \Delta t) = x + \Delta x.$$

This follows from the principle of optimality because the control $u(\tau)$ for $t+\Delta t \le \tau \le T$ is optimal for the problem starting at time $t + \Delta t$ in state $x(t + \Delta t) = x + \Delta x$.

Now using the definition of $V(t, x)$, we can rewrite (1.5) as

$$V(t, x) = \min_{\substack{u \\ t \le \tau \le t+\Delta t}} \left[\int_t^{t+\Delta t} L\,d\tau + V(t + \Delta t, x + \Delta x) \right].$$

Assuming $V(x, t)$ to be twice differentiable and expanding $V(t + \Delta t, x + \Delta x)$ using Taylor's theorem gives

$$\begin{aligned} V(t, x) &= \min_u [L(x, u, t)\Delta t + V(t, x) + V_t(t, x)\Delta t \\ &\qquad + V_x(t, x)\Delta x + \text{higher order terms}]. \end{aligned}$$

Subtracting $V(t, x)$ from both sides, dividing by Δt and letting $\Delta t \to 0$ we obtain

$$\begin{aligned} 0 &= \min_u [L(x, u, t) + V_t(t, x) + V_x(t, x)\dot{x}] \\ &= \min_u [L(x, u, t) + V_t(t, x) + V_x(t, x) f(x, u, t)]. \end{aligned}$$

Since the term $V_t(t,x)$ does not involve the decision u, we move it to the left and obtain the partial differential equation (PDE) for the value function $V(t,x)$ as

$$-V_t(t,x) = \min_u [L(x,u,t) + V_x(t,x) f(x,u,t)]$$

with the boundary condition $V[x(T),T] = S[x(T),T]$. The resulting PDE is known as the *Hamilton-Jacobi-Bellman* (HJB) equation since it was originally developed by Bellman [Bel53] to solve problems in the calculus of variations. For discussions of the development of this PDE and examples of its solution, see Bertsekas [Ber95, p. 91–92], Bryson and Ho [BH75, p. 135], Kamien and Schwartz [KS81, pp. 238–240], Kirk [Kir70, pp. 86–90] and Sage and White [SW77, pp. 76–77].

A Remark on Discounting

Note that if we have an infinite horizon problem and if continuous discounting is applied with a discount factor e^{-rt}, then the optimization problem would be to minimize $\int_0^\infty e^{-rt} L(x,u,t)\,dt$ subject to the state equation $\dot{x} = f(x,u)$ with the initial state as $x(0) = x_0$. In this case, the HJB equation would be written as

$$-V_t(t,x) = \min_u [e^{-rt} L(x,u,t) + V_x(t,x) f(x,u,t)]$$

without any boundary condition on the value function. Here the value function is defined as the minimum cost *discounted to time zero* that can be obtained starting at time t in state x and using the optimal policy. However, it can be shown that for "autonomous" systems where time does not appear in the cost function (except for the discount term) and in the system equations, a similar *ordinary* differential equation (ODE) can be found for the HJB problem where the value function $V(x)$ would be defined as the minimum cost discounted to the *current time*.

In autonomous systems, the problem is to minimize $\int_0^\infty e^{-rt} L(x,u)\,dt$ subject to $\dot{x} = f(x,u)$. For this problem we write the HJB equation where discounting is to the current time (rather than to time zero) as

$$V(t,x) = \min_u [L(x,u) + e^{-r\Delta t} V(t+\Delta t, x+\Delta x)].$$

Since $e^{-r\Delta t} \approx 1 - r\Delta t$ and the value function at $(t+\Delta t, x+\Delta x)$ is expanded as

$$V(t+\Delta t, x+\Delta x) \approx V(t,x) + V_t \Delta t + V_x \Delta x,$$

after simplifications and usual limiting operations the HJB equation is obtained as an ODE in terms of $V(x)$ as follows:[6]

$$rV(x) = \min_u [L(x,u) + V'(x) f(x,u)].$$

[6]Note that in the limiting operations we have $\lim_{\Delta t \to 0} \Delta x/\Delta t = \dot{x} = f(x,u)$.

For a more detailed discussion of this issue, see Beckmann [Bec68, Part IV] and Kamien and Schwartz [KS81, pp. 241–242].

We now consider two examples where the HJB partial differential equation can be solved explicitly to find the value function and the optimal policy.

1.4.1 A Problem with Linear System and Quadratic Cost

Consider the following continuous-time version of the sequential optimization problem with quadratic cost and linear state equation adapted from Kamien and Schwartz [KS81, pp. 240–241]. We wish to minimize

$$\int_0^\infty e^{-rt}(Qx^2 + Ru^2)\,dt$$

subject to $\dot{x} = u$ with the initial condition $x(0) = x_0 > 0$. We now solve the HJB equation $-V_t = \min_u[e^{-rt}(Qx^2 + Ru^2) + V_x u]$ for this problem using Maple.

```
> restart: # LQPDE.mws
> L:=(x,u)->exp(-r*t)*(Q*x^2+R*u^2);
```

$$L := (x,\ u) \to e^{(-r\,t)}\,(Q\,x^2 + R\,u^2)$$

```
> f:=u;
```

$$f := u$$

```
> RHS:=L(x,u)+diff(V(t,x),x)*f;
```

$$RHS := e^{(-r\,t)}\,(Q\,x^2 + R\,u^2) + (\tfrac{\partial}{\partial x}\,\mathrm{V}(t,\ x))\,u$$

The optimal policy is obtained in terms of V_x as follows.

```
> uSol:=combine(solve(diff(RHS,u),u),exp);
```

$$uSol := -\frac{1}{2}\,\frac{(\frac{\partial}{\partial x}\,\mathrm{V}(t,\ x))\,e^{(r\,t)}}{R}$$

Using this policy we develop the HJB partial differential equation and attempt to solve it using Maple's `pde()` function.

```
> HJB:=-diff(V(t,x),t)=L(x,uSol)
  +diff(V(t,x),x)*uSol;
```

$$HJB := -(\tfrac{\partial}{\partial t}\,\mathrm{V}(t,\ x)) =$$

$$e^{(-r\,t)}\left(Q\,x^2 + \frac{1}{4}\,\frac{(\frac{\partial}{\partial x}\,\mathrm{V}(t,\ x))^2\,(e^{(r\,t)})^2}{R}\right) - \frac{1}{2}\,\frac{(\frac{\partial}{\partial x}\,\mathrm{V}(t,\ x))^2\,e^{(r\,t)}}{R}$$

```
> pdesolve(HJB, V(t,x));

Error, (in pdesolve/exact/charac) dsolved returned
multiple answers,
[{t(_F1) = -_F1+_C4}, {P[2](_F1) =
DESol({(diff(diff(_Y(_F1),_F1),_F1)*R
+r*diff(t(_F1),_F1)*diff(_Y(_F1),
```

```
_F1)*R+Q*_Y(_F1))/R},{_Y(_F1)})
*_C3, x(_F1) =-1/2*exp(r*t(_F1))/Q
*diff(DESol({(diff(diff(_Y(_F1),_F1),_F1)*R
+r*diff(t(_F1),_F1)*diff(_Y(_F1),_F1)*R
+Q*_Y(_F1))/R},{_Y(_F1)}),_F1)*_C3},
{P[1](_F1) =Int(1/4*r*(4*exp(-2*r*t(_F1))
*Q*x(_F1)^2*R+P[2](_F1)^2)*exp(r*t(_F1))/
R,_F1)+_C2}, {U(_F1) =
Int((-P[1](_F1)*R+1/2*P[2](_F1)^2
*exp(r*t(_F1)))/R,_F1)+_C1}]
```

Unfortunately, Maple fails to find a solution to this PDE and returns an error message. Thus, using our intuition obtained from the discrete-time version of this type of problem with quadratic cost and linear system dynamics, we try a solution in the form $V(t,x) = e^{-rt}Ax^2$ where the constant A is to be determined.

```
> V:=(t,x)->exp(-r*t)*A*x^2; Vt:=diff(V(t,x),t);
  Vx:=diff(V(t,x),x);
```

$$V := (t,\ x) \to e^{(-r\,t)}\,A\,x^2$$

$$Vt := -r\,e^{(-r\,t)}\,A\,x^2$$

$$Vx := 2\,e^{(-r\,t)}\,A\,x$$

```
> HJB;
```

$$r\,e^{(-r\,t)}\,A\,x^2 =$$
$$e^{(-r\,t)}\,(Q\,x^2 + \frac{(e^{(-r\,t)})^2\,A^2\,x^2\,(e^{(r\,t)})^2}{R}) - 2\,\frac{(e^{(-r\,t)})^2\,A^2\,x^2\,e^{(r\,t)}}{R}$$

```
> simplify(HJB);
```

$$r\,e^{(-r\,t)}\,A\,x^2 = -\frac{e^{(-r\,t)}\,x^2\,(-Q\,R + A^2)}{R}$$

Substituting and simplifying, we finally reduce the problem to the solution of a quadratic equation in terms of the unknown A:

```
> ASol:=solve(HJB,A);
```

$$ASol := -\frac{1}{2}\,r\,R + \frac{1}{2}\,\sqrt{r^2\,R^2 + 4\,Q\,R},\ -\frac{1}{2}\,r\,R - \frac{1}{2}\,\sqrt{r^2\,R^2 + 4\,Q\,R}$$

Since the value function must be positive, we choose the first solution of `ASol` so that we have $A = -\frac{1}{2}rR + \frac{1}{2}\sqrt{r^2R^2 + 4QR}$. The optimal policy is then obtained as $u = -(A/R)x$.

```
> uSol:=combine(uSol);
```

$$uSol := -\frac{A\,x}{R}$$

1.4.2 A Problem with Quadratic and Linear Costs

Many of the examples discussed are somewhat standard because they involve a quadratic cost function with linear state equations. We now describe the solution of a more difficult problem where the cost function consists of a *mixture* of quadratic and linear terms. We will see that Maple is again very helpful in manipulating expressions that give rise to the value function and the optimal policy for the model.

Consider the following problem that appears as an exercise in Kamien and Schwartz [KS81, p. 242]. We wish to minimize the cost functional $\int_0^T (c_1 u^2 + c_2 x)\, dt$ subject to $\dot{x} = u$ with $x(0) = 0$ and $x(T) = B$. Kamien and Schwartz suggest using $V(t,x) = a + bxt + hx^2/t + kt^3$ as the trial form for the value function. Keeping this hint in mind, we proceed with the development of the HJB partial differential equation.

```
> restart: # LPDE.mws
> L:=(x,u)->c[1]*u^2+c[2]*x;
```

$$L := (x,\, u) \to c_1\, u^2 + c_2\, x$$

```
> f:=u;
```

$$f := u$$

The right-hand side of the HJB equation that will be minimized is $L(x,u) + V_x f(x,u)$. Differentiating this expression, equating to zero and solving for u gives the optimal policy function in terms of the unknown V_x.

```
> RHS:=L(x,u)+diff(V(t,x),x)*f;
```

$$RHS := c_1\, u^2 + c_2\, x + (\tfrac{\partial}{\partial x}\, \mathrm{V}(t,\, x))\, u$$

```
> uSol:=solve(diff(RHS,u),u);
```

$$uSol := -\frac{1}{2}\, \frac{\frac{\partial}{\partial x}\, \mathrm{V}(t,\, x)}{c_1}$$

```
> HJB:=-diff(V(t,x),t)=L(x,uSol)
  +diff(V(t,x),x)*uSol;
```

$$HJB := -(\tfrac{\partial}{\partial t}\, \mathrm{V}(t,\, x)) = -\frac{1}{4}\, \frac{(\frac{\partial}{\partial x}\, \mathrm{V}(t,\, x))^2}{c_1} + c_2\, x$$

An attempt to solve the PDE explicitly using Maple fails.

```
> pdesolve(HJB, V(t,x));

Error, (in pdesolve/exact/charac) dsolved returned
multiple answers,
[{P[1](_F1) = _C4}, {P[2](_F1) = -c[2]*_F1+_C3},
{U(_F1) =Int((-P[1](_F1)*c[1]
+1/2*P[2](_F1)^2)/c[1],_F1)+_C2}, {x(_F1) =
Int(1/2/c[1]*P[2](_F1),_F1)+_C1},
{t(_F1) = -_F1+_C5}]
```

At this stage we make use of the trial form of the value function and compute its derivatives with respect to t and x.

```
> V:=(t,x)->a+b*x*t+h*x^2/t+k*t^3;
  Vt:=diff(V(t,x),t); Vx:=diff(V(t,x),x);V(T,B);
```

$$V := (t,\ x) \to a + b\,x\,t + \frac{h\,x^2}{t} + k\,t^3$$

$$Vt := b\,x - \frac{h\,x^2}{t^2} + 3\,k\,t^2$$

$$Vx := b\,t + 2\,\frac{h\,x}{t}$$

$$a + b\,B\,T + \frac{h\,B^2}{T} + k\,T^3$$

```
> HJB:=lhs(HJB)-rhs(HJB);
```

$$HJB := -b\,x + \frac{h\,x^2}{t^2} - 3\,k\,t^2 + \frac{1}{4}\,\frac{(b\,t + 2\,\frac{h\,x}{t})^2}{c_1} - c_2\,x$$

The HJB equation is now reduced to a nonlinear algebraic equation in terms of x, t^2 and x^2/t^2 with unknown constants A, B, H and K, which we now compute analytically.

The coefficients of x and t^2 are easily extracted using the Maple `coeff()` function.

```
> cx:=coeff(HJB,x);
```

$$cx := -b + \frac{b\,h}{c_1} - c_2$$

```
> ct2:=coeff(HJB,t^2);
```

$$ct2 := -3\,k + \frac{1}{4}\,\frac{b^2}{c_1}$$

To extract the coefficient of x^2/t^2, we use a somewhat obscure Maple function `algsubs()` that performs more general algebraic substitutions than the substitutions `subs()` function can make. This substitution will have a temporary effect of defining x^2/t^2 as z^2, thus making the expression a polynomial so that the `coeff()` function can be applied.

```
> x2t2:=algsubs(x^2/t^2=z^2,expand(HJB));
```

$$x2t2 := -\frac{(b\,c_1 - b\,h + c_2\,c_1)\,x}{c_1} - \frac{1}{4}\,\frac{(12\,k\,c_1 - b^2)\,t^2}{c_1} + \frac{h\,z^2\,(c_1 + h)}{c_1}$$

```
> cx2t2:=coeff(x2t2,z^2);
```

$$cx2t2 := \frac{h\,(c_1 + h)}{c_1}$$

To make sure that we have not missed any terms while extracting the coefficients, we perform a simple check and observe that everything is in order.

```
> simplify(cx*(x)+ct2*(t^2)+cx2t2*(x^2/t^2)-HJB);
```

$$0$$

Now that we have the algebraic expressions for the coefficients of x, t^2 and x^2/t^2, (i.e., `cx`, `ct2` and `cx2t2`) and the final time condition that $V[T, x(T)] = V(T, B) = 0$, we can solve the resulting system of four nonlinear equations in the four unknowns a, b, h and k.

```
> solve({cx,ct2,cx2t2,V(T,B)},{a,b,h,k});
```

$$\{h = 0,\ k = \frac{1}{12}\,\frac{{c_2}^2}{c_1},\ a = -\frac{1}{12}\,\frac{c_2\,T\,(-12\,B\,c_1 + c_2\,T^2)}{c_1},\ b = -c_2\},\ \{h = -c_1,$$

$$k = \frac{1}{48}\,\frac{{c_2}^2}{c_1},\ b = -\frac{1}{2}\,c_2,\ a = -\frac{1}{48}\,\frac{-24\,c_2\,B\,T^2\,c_1 - 48\,{c_1}^2\,B^2 + {c_2}^2\,T^4}{T\,c_1}\}$$

Maple finds two solutions but we use the one where the solution is nonzero.

```
> assign(%[2]);
> a; b; h; k;
```

$$-\frac{1}{48}\,\frac{-24\,c_2\,B\,T^2\,c_1 - 48\,{c_1}^2\,B^2 + {c_2}^2\,T^4}{T\,c_1}$$

$$-\frac{1}{2}\,c_2$$

$$-c_1$$

$$\frac{1}{48}\,\frac{{c_2}^2}{c_1}$$

The value function $V(t, x)$ and the optimal production rate $\dot{x} = u$ are now easily computed as functions of the stage t and state x.

```
> V(t,x); # Value function
```

$$-\frac{1}{48}\,\frac{-24\,c_2\,B\,T^2\,c_1 - 48\,{c_1}^2\,B^2 + {c_2}^2\,T^4}{T\,c_1} - \frac{1}{2}\,c_2\,x\,t - \frac{c_1\,x^2}{t} + \frac{1}{48}\,\frac{{c_2}^2\,t^3}{c_1}$$

```
> Vx;
```

$$-\frac{1}{2}\,c_2\,t - 2\,\frac{c_1\,x}{t}$$

```
> expand(simplify(uSol)); # Optimal production rate
```

$$\frac{1}{4}\,\frac{t\,c_2}{c_1} + \frac{x}{t}$$

To obtain an analytic expression for the optimal production rate $u(t)$ and for the optimal inventory level $x(t)$, we solve $\dot{x}(t) = u(t)$ using the form of the optimal policy just determined.

```
> dsolve({diff(x(t),t)=(t*c[2]/(4*c[1]))+x(t)/t,
  x(0)=0, x(T)=B},x(t));
```

$$\mathrm{x}(t) = \frac{1}{4}\,\frac{t^2\,c_2}{c_1} - \frac{1}{4}\,\frac{t\,(c_2\,T^2 - 4\,B\,c_1)}{T\,c_1}$$

```
> assign(%);
> xOpt:=unapply(normal(x(t)),t);
  uOpt:=unapply(normal(diff(x(t),t)),t);
```

$$xOpt := t \to -\frac{1}{4}\,\frac{t\,(-c_2\,t\,T + c_2\,T^2 - 4\,B\,c_1)}{T\,c_1}$$

$$uOpt := t \to -\frac{1}{4}\,\frac{-2\,c_2\,t\,T + c_2\,T^2 - 4\,B\,c_1}{T\,c_1}$$

1.5 A Constrained Work Force Planning Model

Many of the models discussed so far were relatively simple since the costs were quadratic and there were no constraints on the state or the decision variables. We now present a more complicated sequential decision problem of work force planning with constraints on both state and decision variables. For a discussion of similar models, see Bensoussan, Hurst and Näslund [BHN74, pp. 58–72], Hillier and Lieberman [HL86, pp. 345–350], Holt, Modigliani, Muth and Simon [HMMS60] and Sasieni, Yaspan and Friedman [SYF59, p. 280].

The work force requirements in a particular factory fluctuate due to seasonality of the demand. The estimated work force requirements during each of the four seasons for the indefinite future are as follows:

Season	Summer	Fall	Winter	Spring
Requirements (r_t)	$r_1 = 210$	$r_2 = 250$	$r_3 = 190$	$r_4 = 260$

Due to the undesirability—and costliness—of hiring and firing of the workers, management wishes to minimize fluctuations in work force levels. It is estimated that the total cost of changing the work force level from one season to the next is $s = \$100$ times the square of the difference in the levels between the two seasons. For example, if the Summer employment level were 215 people and if an additional 10 workers were hired for the Fall, the cost of this change in the work force would be $\$100 \times (225 - 215)^2 = \$10000$. Any employment level that is above the levels given in the table is considered a "waste" that costs the company approximately $w = \$1000$/person/season. For example, a Summer level of 215 would result in a waste of $\$1000 \times (215 - 210) = \$5000$ for that season.

Although this is a problem with an infinite number of stages, each year starts an identical cycle. Since the cost data are assumed to be stationary, we can consider only one cycle of four seasons that end with Spring. We will assume that the Spring production is the most important and that the management has decided

to keep the Spring employment level at $u_4 = 260$. Also, note that since the estimate of 260 for the Spring is the highest, it would not be optimal to increase the employment level at any season above 260.

We denote the initial Summer season by $t = 1$ and the other periods as $t = 2, \ldots, 4$. If we define the decision variables u_t, $t = 1, \ldots, 4$ as the employment levels used in period t (with $u_4 = 260$), it is easy to see that this problem can be formulated as a nonlinear programming problem. For this problem the objective function would be

$$\min \sum_{t=1}^{4} [100(u_t - u_{t-1})^2 + 1000(u_t - r_t)]$$

subject to the constraints $210 \leq u_1 \leq 260$, $250 \leq u_2 \leq 260$, $190 \leq u_3 \leq 260$ (and $u_4 = 260$). If this problem were solved using the techniques of nonlinear programming, we would obtain the *open-loop* decisions as $u_1 = 252.5$, $u_2 = 250$, $u_3 = 252.5$ and $u_4 = 260$ with a minimum total cost of \$117,500. Fractional levels such 252.5 are assumed to be acceptable since they correspond to the employment of part-time workers.

We will now solve this problem with Maple's help using dynamic programming. Since in the current stage the state (i.e., the available information about the system) is the previous stage's decision (i.e., the employment level chosen), we write $x_{t+1} = u_t$, $t = 1, 2, 3$ as the state equations with $x_1 = 260$. This gives rise to the constraints on the states x_t, i.e., $210 \leq x_2 \leq 260$, $250 \leq x_3 \leq 260$, and $190 \leq x_4 \leq 260$. Defining $V_t(x_t)$ as the value function for stage t—the minimum cost obtainable for periods $t, t+1, \ldots, 4$—when the optimal policy is implemented, the DP functional equation becomes

$$V_t(x_t) = \min_{r_t \leq u_t \leq r_4} [100(u_t - x_t)^2 + 1000(u_t - r_t) + V_{t+1}(x_{t+1})], t = 1, \ldots, 4$$

with the boundary condition

$$\begin{aligned} V_4(x_4) &= 100(u_4 - x_4)^2 + 1000(u_4 - r_4) \\ &= 100(260 - x_4)^2 \end{aligned}$$

since in stage 4 we must have $u_4 = r_4 = 260$.

We start by assigning values to the seasonal work force requirements and compute the maximum requirement as $R = \max_t \{r_t\} = 260$.

```
> restart: # Manpower.mws
> with(linalg):
Warning, new definition for norm

Warning, new definition for trace
```

```
> s:=100; w:=1000;
```

$$s := 100$$

$$w := 1000$$

```
> r:=array(1..4,[210,250,190,260]);
  R:=max(seq(r[i],i=1..4));
```

$$r := [210,\ 250,\ 190,\ 260]$$

$$R := 260$$

For stage 4 the value function $V_4(x_4)$ is defined as the boundary condition.

```
> V[4]:=unapply(s*(r[4]-x)^2,x);
```

$$V_4 := x \rightarrow 100\,(260 - x)^2$$

For stage 3 the recursive relationship is

$$V_3(x_3) = \min_{190 \le u_3 \le 260} c_3(x_3, u_3)$$

where $c_3(x_3, u_3) = 100(u_3 - x_3)^2 + 1000(u_3 - 190) + V_4(x_4)$. We show that c_3 is convex in u_3 given x_3.

Differentiating c_3 with respect to $u_3 = u$, equating the result to 0 and solving gives an expression for u_3 (or the policy μ_3) in terms of the state variable $x_3 = x$.

Recall that u_3 is feasible in the interval from $r_3 = 190$ to $r_4 = R = 260$. Evaluating the policy μ_3 at the end points of x_3, we find $\mu_3(x_3 = r_2) = 252.5$ and $\mu_3(x_3 = R) = 257.5$. This shows that the policy μ_3 is in the feasible interval for any x_3; i.e., we have $\mu_3 = \frac{1}{2}x_3 + \frac{255}{2}$ for $250 \le x_3 \le 260$.

With this information about the optimal u_3 we easily compute the value function $V_3(x)$ for stage 3.

```
> c[3]:=s*(u-x)^2+w*(u-r[3])+V[4](u);
```

$$c_3 := 100\,(u - x)^2 + 1000\,u - 190000 + 100\,(260 - u)^2$$

```
> cp[3]:=diff(c[3],u); cpp[3]:=diff(c[3],u$2);
```

$$cp_3 := 400\,u - 200\,x - 51000$$

$$cpp_3 := 400$$

```
> mu[3]:=solve(cp[3],u);
```

$$\mu_3 := \frac{1}{2}\,x + \frac{255}{2}$$

```
> # u[3] is feasible from r[3]=190 to R=260
> evalf(subs(x=r[2],mu[3])); evalf(subs(x=R,mu[3]));
```

$$252.5000000$$

$$257.5000000$$

```
> V[3]:=unapply(subs(u=mu[3],c[3]),x);
```

$$V_3 := x \to 100\,(-\frac{1}{2}x + \frac{255}{2})^2 + 500\,x - 62500 + 100\,(\frac{265}{2} - \frac{1}{2}x)^2$$

For stage 2, the recursive equation becomes

$$V_2(x_2) = \min_{250 \le u_2 \le 260} c_2(x_2, u_2)$$

where $c_2(x_2, u_2) = 100(u_2 - x_2)^2 + 1000(u_2 - 250) + V_3(x_3)$. Here the analysis becomes more complicated.

Differentiating c_2 with respect to u_2 we get $\partial c_2/\partial u_2 = 300u_2 - 200x_2 + 24500$. Next, equating this result to zero and solving for u_2 we find the candidate policy as $\mu_2 = \frac{2}{3}x_2 + \frac{245}{3}$. Now, recall that u_2 is feasible in the interval from $r_2 = 250$ to $r_4 = R = 260$, but evaluating the policy μ_2 at the extreme points of x_2 we find $\mu_2(x_2 = r_1 = 210) = 221.67$ (infeasible) and $\mu_2(x_2 = R = 260) = 255$ (feasible).

To find the range of x_2 for which u_2 is feasible, we solve $\mu_2 = 250$ for x_2 and obtain $x_2 = 252.5$. This shows that the policy μ_2 applies only in the interval for x_2 from 252.5 to 260. For a given $x_2 \le 252.5$, we find that c_2 is convex and increasing in u_2. Hence, we set $\mu_2 = 250$ for $x_2 \le 252.5$. This gives a piecewise function for the policy μ_2; see Figure 1.1.

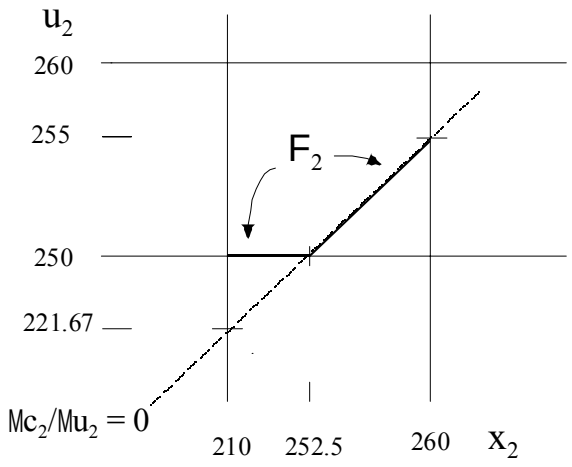

FIGURE 1.1. The form of the policy $\mu_2(x_2)$.

The value function $V_2(x_2)$ and c_1 are easily computed in terms of the `piecewise()` function. But we suppress the output of these functions in order to conserve space.

```
> c[2]:=s*(u-x)^2+w*(u-r[2])+V[3](u);
```

$$c_2 := 100\,(u-x)^2 + 1500\,u - 312500 + 100\,(-\frac{1}{2}\,u + \frac{255}{2})^2 + 100\,(\frac{265}{2} - \frac{1}{2}\,u)^2$$

```
> cp[2]:=diff(c[2],u); cpp[2]:=diff(c[2],u$2);
```

$$cp_2 := 300\,u - 200\,x - 24500$$

$$cpp_2 := 300$$

```
> mu[2]:=solve(cp[2],u);
```

$$\mu_2 := \frac{2}{3}\,x + \frac{245}{3}$$

```
> # u[2] is feasible from r[2]=250 to R=260
> evalf(subs(x=r[1],mu[2])); evalf(subs(x=R,mu[2]));
```

$$221.6666667$$

$$255.$$

```
> xLow:=evalf(solve(mu[2]=250,x));
```

$$xLow := 252.5000000$$

```
> xHigh:=evalf(min(R,solve(mu[2]=260,x)));
```

$$xHigh := 260.$$

```
> uLow:=subs(x=xLow,mu[2]);
```

$$uLow := 250.0000000$$

```
> uHigh:=subs(x=xHigh,mu[2]);
```

$$uHigh := 255.0000000$$

```
> mu[2]:=piecewise(x<=xLow,r[2],x<R,mu[2]);
```

$$\mu_2 := \begin{cases} 250 & x \le 252.5000000 \\ \frac{2}{3}\,x + \frac{245}{3} & x < 260 \end{cases}$$

```
> cp[2];
```

$$300\,u - 200\,x - 24500$$

```
> assume(210<x,x<252.5,250<u,u<260); is(cp[2]>0);
```

$$true$$

```
> x:='x'; u:='u';
```

$$x := x$$

$$u := u$$

```
> V[2]:=unapply(subs(u=mu[2],c[2]),x):
> c[1]:=s*(u-r[4])^2+w*(u-r[1])+V[2](u):
```

Although we do not present the graph of the c_1 function here, we include a Maple command `plot(c[1],u=190..260)` that can be used to note that c_1 is convex in u_1. Thus differentiating this function using the Maple `diff()` command and solving gives the optimal numerical value of the first period's decision as $u_1 = 252.5$. The minimum total cost for all four periods is now easily found as $V_1 = 117500$ by substituting the optimal u_1 into c_1.

```
> #plot(c[1],u=190..260,discont=true);
> cp[1]:=diff(c[1],u):
> mu[1]:=evalf(solve(cp[1],u)); u[1]:=mu[1];
```

$$\mu_1 := 252.5000000$$

$$u_1 := 252.5000000$$

```
> V[1]:=evalf(subs(u=mu[1],c[1]));
```

$$V_1 := 117500.0000$$

The optimal employment levels for the other periods are easily computed as $u_2 = 250$, $u_3 = 252.5$ and $u_4 = 260$.

```
> x:=mu[1]; u[2]:=mu[2];
```

$$x := 252.5000000$$

$$u_2 := 250$$

```
> x:=mu[2]; u[3]:=evalf(mu[3]);
```

$$x := 250$$

$$u_3 := 252.5000000$$

```
> x:=evalf(mu[3]); u[4]:=r[4];
```

$$x := 252.5000000$$

$$u_4 := 260$$

The following table summarizes the solution.

t	1	2	3	4
r_t	210	250	190	260
u_t	252.5	250	252.5	260

1.6 A Gambling Model with Myopic Optimal Policy

In his seminal paper entitled "A New Interpretation of Information Rate," Kelly [Kel56] considered an unscrupulous gambler who was receiving prior information concerning the outcomes of sporting events over a noisy communication channel (e.g., a phone line) before the results became common knowledge. After receiving the word "win" or "lose"—which he could hear incorrectly due to communication difficulties—the gambler would place his bet (a nonnegative amount up to his

present fortune) on the original odds. With p as the probability of correct transmission and $q = 1 - p$ the probability of an incorrect transmission, Kelly showed using calculus techniques that if $p > q$, the gambler's optimal wager u (i.e., the fraction of his capital bet each time) was simply $u = p - q$. If $p \leq q$, then $u = 0$. He also showed that the maximum growth of capital occurred at a rate equal to the capacity of the channel given as $G_{\max} = \log_2 2 + p \log_2 p + q \log_2 q$. (Interestingly, this was the same result obtained by Shannon [Sha48] from considerations of coding of information.)

Kelly's work attracted the attention of a large number of researchers including economists, e.g., Arrow [Arr71], psychologists, e.g., Edwards [Edw62] and applied mathematicians such as Bellman and Kalaba [BK57a], [BK57b], [Bel61] who extended and reinterpreted Kelly's results. In particular, Bellman and Kalaba re-solved the same problem after formulating it as a dynamic program with a logarithmic utility function for the terminal wealth of the gambler. They assumed that the gambler is allowed N plays (bets) and that at each play of the gamble, he could bet any nonnegative amount up to his present fortune.

In the formulation of the gambling problem using DP we define $V_n(x_n)$ as the maximum expected return if the gambler has a present fortune of x_n and has n gambles left. (This is a slight change in notation compared to what we had used before. In this model $n = N - t$ is the number of gambles *left* where $n = 0, 1, \ldots, N$.) At each play, the gambler could bet any nonnegative amount up to his present fortune and win the gamble with a probability of p. The objective is to maximize the terminal utility of his wealth which, is assumed to be logarithmic; i.e., when there are no gambles left, the gambler's utility of his final wealth is $V_0(x_0) = \log(x_0)$. When n gambles are left, we define u_n as the fraction of the current wealth to gamble with $0 \leq u_n \leq 1$. Then the state equation assumes the form $x_{n-1} = f_n(x_n, u_n, w_n) = x_n + u_n x_n w_n$, $n = 1, \ldots, N$ where the random variable w_n takes the values 1 or -1 with probability p and $q = 1 - p$, respectively. Thus, the DP functional equation is written as

$$\begin{aligned} V_n(x_n) &= \max_{0 \leq u_n \leq 1} E_{w_n} V_{n-1}(x_n + u_n x_n w_n) \\ &= \max_{0 \leq u_n \leq 1} [p V_{n-1}(x_n + u_n x_n) + q V_{n-1}(x_n - u_n x_n)] \end{aligned}$$

with the boundary condition $V_0(x_0) = \log(x_0)$.

We now describe the DP solution of Kelly's problem using Maple. Our notation and exposition closely follow those of Ross [Ros83, pp. 2–4].

First, we show that when $p \leq \frac{1}{2}$, it is optimal not to bet ($u = 0$) and that $V_n(x_n) = \log(x_n)$. Starting with $n = 1$, we see that the c_1 function is monotone decreasing in u, so the optimal decision is to bet 0. (Actually when $p = \frac{1}{2}$, setting $u = 2p - 1$ makes c_1 constant. But since $u = 2p - 1 = 0$, the statement about the optimal decision is still valid.) With $\mu_1(x_1) = 0$, we obtain $V_1(x_1) = \log(x_1)$. For other values of $n > 1$, the same solution can be shown to be valid proving that when $p \leq \frac{1}{2}$, $\mu_n(x_n) = 0$ and $V_n(x_n) = \log(x_n)$.

```
> restart: # Gamble.mws
```

```
> assume(0<=p,p<=1/2): additionally(0<=u,u<=1):
> q:=1-p;
```

$$q := 1 - p$$

```
> c[1]:=p*log(x+u*x)+q*log(x-u*x);
```

$$c_1 := p\ln(x + u\,x) + (1 - p)\ln(x - u\,x)$$

```
> c[1,1]:=normal(diff(c[1],u));
```

$$c_{1,1} := -\frac{2\,p - 1 - u}{(1 + u)\,(-1 + u)}$$

```
> is(c[1,1]<=0);
```

$$true$$

```
> V[1]:=simplify(subs(u=0,c[1]));
```

$$V_1 := \ln(x)$$

We now turn to the case where $p > \frac{1}{2}$.

```
> restart: # Gamble.mws (Part 2)
> assume(x>0,p>0);
> q:=1-p;
```

$$q := 1 - p$$

The boundary condition is the utility of the final fortune, which is defined as the logarithmic function in x.

```
> V[0]:=x->log(x);
```

$$V_0 := \log$$

When there is one play left, we find that it is optimal to bet $\mu_1 = 2p - 1 = p - (1 - p) = p - q$ fraction of the current wealth:

```
> c[1]:=p*log(x+u*x)+q*log(x-u*x);
```

$$c_1 := p\ln(x + u\,x) + (1 - p)\ln(x - u\,x)$$

```
> deriv[1]:=diff(c[1],u);
```

$$deriv_1 := \frac{p\,x}{x + u\,x} - \frac{(1 - p)\,x}{x - u\,x}$$

```
> mu[1]:=solve(deriv[1],u);
```

$$\mu_1 := 2\,p - 1$$

The value function $V_1(x)$ is obtained in terms of logarithms that can be written as $V_1(x) = C + \log(x)$ where $C = \log(2) + p\log(p) + \log(1-p) - p\log(1-p) = \log(2) + p\log(p) + q\log(q)$.

```
> V[1]:=unapply(expand(simplify(
  subs(u=mu[1],c[1]))),x);
```

$$V_1 := x \to p\ln(p) + \ln(2) + \ln(x\sim) + \ln(1 - p) - p\ln(1 - p)$$

When there are two plays left, the optimal fraction to bet is again $\mu_2 = p - q$ with the value function assuming the form $V_2(x) = 2C + \log(x)$.

```
> c[2]:=p*V[1](x+u*x)+q*V[1](x-u*x);
```

$$c_2 := p\,(p\ln(p) + \ln(2) + \ln(x + u\,x) + \ln(1-p) - p\ln(1-p)) + (1-p)\,(p\ln(p) + \ln(2) + \ln(x - u\,x) + \ln(1-p) - p\ln(1-p))$$

```
> deriv[2]:=diff(c[2],u);
```

$$deriv_2 := \frac{p\,x}{x + u\,x} - \frac{(1-p)\,x}{x - u\,x}$$

```
> mu[2]:=solve(deriv[2],u);
```

$$\mu_2 := 2\,p - 1$$

```
> V[2]:=unapply(expand(simplify(
  subs(u=mu[2],c[2]))),x);
```

$$V_2 := x \to 2\,p\ln(p) - 2\,p\ln(1-p) + 2\ln(2) + \ln(x\sim) + 2\ln(1-p)$$

This procedure can be automated to produce similar results.

```
> for n from 3 to 4 do:
> c[n]:=p*V[n-1](x+u*x)+q*V[n-1](x-u*x):
> deriv[n]:=diff(c[n],u):
> mu[n]:=solve(deriv[n],u):
> V[n]:=unapply(expand(simplify(
  subs(u=mu[n],c[n]))),x):
> od;
```

$$c_3 := p\,(2\,p\ln(p) - 2\,p\ln(1-p) + 2\ln(2) + \ln(x + u\,x) + 2\ln(1-p)) + (1-p)\,(2\,p\ln(p) - 2\,p\ln(1-p) + 2\ln(2) + \ln(x - u\,x) + 2\ln(1-p))$$

$$deriv_3 := \frac{p\,x}{x + u\,x} - \frac{(1-p)\,x}{x - u\,x}$$

$$\mu_3 := 2\,p - 1$$

$$V_3 := x \to 3\,p\ln(p) - 3\,p\ln(1-p) + 3\ln(2) + \ln(x\sim) + 3\ln(1-p)$$

$$c_4 := p\,(3\,p\ln(p) - 3\,p\ln(1-p) + 3\ln(2) + \ln(x + u\,x) + 3\ln(1-p)) + (1-p)\,(3\,p\ln(p) - 3\,p\ln(1-p) + 3\ln(2) + \ln(x - u\,x) + 3\ln(1-p))$$

$$deriv_4 := \frac{p\,x}{x + u\,x} - \frac{(1-p)\,x}{x - u\,x}$$

$$\mu_4 := 2\,p - 1$$

$$V_4 := x \to 4\,p\ln(p) - 4\,p\ln(1-p) + 4\ln(2) + \ln(x\sim) + 4\ln(1-p)$$

Using induction, it can then be shown that when $p > \frac{1}{2}$ the optimal policy is to bet $\mu_n = p - q$ fraction of the current wealth at any stage of the game. This gives rise to a value function that is in the form $V_n(x) = nC(p) + \log(x)$ where $C(p) = q\ln(q) + p\ln(p) + \ln(2)$.

One of the most interesting features of this problem is the nature of its solution: The optimal strategy is myopic (invariant) in the sense that regardless of the number of bets left to place (n) and the current wealth (x), the optimal (nonnegative) fraction to bet in each period is the same, i.e., $u = p - q$ when $p > \frac{1}{2}$ and $u = 0$ when $p \leq \frac{1}{2}$. Optimal myopic policies of this type are usually difficult to obtain, but they exist for some models. For example, in some periodic-review inventory problems with random demand it has been shown that provided that some terminal cost is appended to the objective function, the optimal order quantity is the same for all periods; i.e., the policy is myopic (Veinott [Vei65], Heyman and Sobel [HS84, Chapter 3]). In some dynamic portfolio problems (Mossin [Mos68], Bertsekas [Ber95, pp. 152–157]) myopic policy is also optimal provided that the amount invested is not constrained and the investor's utility function satisfies certain conditions.

In a recent paper, Çetinkaya and Parlar [ÇP97] argue that the assumption of the simple logarithmic function in the Kelly-Bellman-Kalaba model is somewhat unrealistic and that the problem should ideally be solved with a more general logarithmic utility function. They thus assume that, in general, the gambler's terminal utility is given as $V_0(x) = \log(b + x)$ where b is a positive constant. This generalization results in the disappearance of the myopic nature of the solution and the optimal strategy assumes a form that *depends* on the stage (n) and the state (x) of the gambling process.

1.7 Optimal Stopping Problems

Let us suppose that a professor has N days to leave a country where he has spent the past year on sabbatical. Before leaving the country he wants to sell his car and hence places an advertisement in the local newspaper.

After the publication of the advertisement each day he receives random offers $w_0, w_1, \ldots, w_{N-1}$, which are independent and identically distributed. If the professor accepts an offer, then the process ends; however if he rejects the offer, he must wait until the next day to evaluate the next offer. We assume that offers that are rejected in the past are not retained—the potential buyer goes elsewhere. (For the case where past offers are retained, see Bertsekas [Ber95, p. 173].) The problem is to find a policy that will maximize the professor's revenue when he leaves the country.

Problems of this type are known as *stopping-rule* problems and their origin can be traced back to Wald's research on sequential analysis in the 1940s [Wal47]. For later surveys of stopping-rule problems see Breiman [Bre64] and Leonardz [Leo74]. The second part of Howard's delightful review of dynamic program-

ming [How66] discusses an elementary stopping-rule problem (called an "action-timing" problem by Howard). For a rigorous exposition of the theory of stopping rules, Chow, Robbins and Siegmund [CRS71] can be consulted. Applications in economics of job search are discussed in Lippman and McCall [LM76a], [LM76b].

We let $x_t = w_{t-1}$ be the state variable at the beginning of period t (or, equivalently, at the end of period $t-1$). Defining $V_t(x_t)$ as the value function, the DP functional equation is obtained as

$$V_t(x_t) = \max \begin{cases} x_t & \text{if the decision is to accept the offer of } w_{t-1} \\ E[V_{t+1}(w_t)] & \text{if the decision is to reject the offer of } w_{t-1}. \end{cases}$$

Note that if the last offer w_{t-1} is accepted, then at the start of period t, the professor will have a revenue of x_t and the process will "stop." If the offer is rejected, then $V_t(x_t)$ is equal to the expected value of continuing optimally in periods $t+1, t+2, \ldots, N$, i.e., $E[V_{t+1}(w_t)] = \int_0^\infty V_{t+1}(w)\, dF(w)$ where $F(w)$ is the distribution function of the offer w.

Thus, the optimal policy is to accept the offer $x_t = w_{t-1}$ in period t if x_t exceeds a critical threshold level of $\theta_t = E[V_{t+1}(w_t)]$ and reject the offer if it falls below θ_t, i.e.,

$$\mu_t(x_t) = \begin{cases} \text{Accept} & \text{if } x_t > \theta_t \\ \text{Reject} & \text{if } x_t < \theta_t, \end{cases} \tag{1.6}$$

with either acceptance or rejection being optimal if $x_t = \theta_t$. Hence the value function is $V_t(x_t) = \max(x_t, \theta_t)$. Note that since $x_0 \equiv 0$, we have $V_0(0) = \max(0, \theta_0) = \theta_0$.

As the threshold value θ_t at time t is obtained in terms of $E[V_{t+1}(w_t)]$, we can develop a difference equation in terms of this value and compute θ_t, $t = 0, 1, \ldots, N$ recursively using Maple.

To obtain the difference equation, we write

$$\begin{aligned} \theta_t &= E[V_{t+1}(w_t)] = E[\max(x_{t+1}, \theta_{t+1})] = E[\max(w_t, \theta_{t+1})] \\ &= \theta_{t+1} \int_0^{\theta_{t+1}} dF(w) + \int_{\theta_{t+1}}^\infty w\, dF(w) \\ &= \theta_{t+1} F(\theta_{t+1}) + \int_{\theta_{t+1}}^\infty w\, dF(w) \end{aligned}$$

with the boundary condition being $\theta_N = 0$ (since the professor would be willing to accept *any* offer on his last day). Thus, the threshold levels are found as the solution of the nonlinear difference equation

$$\theta_t = \theta_{t+1} F(\theta_{t+1}) + A(\theta_{t+1}), \qquad t = 0, 1, \ldots, N-1$$

with $A(\theta_{t+1}) = \int_{\theta_{t+1}}^\infty w\, dF(w)$ and $\theta_N = 0$. We now discuss some special cases where the distributions of the random variables assume specific forms.

As a simple example, assume first that the professor has $N = 5$ days to leave the country and his departure is scheduled for next Friday morning. Thus the process of receiving offers starts on day $t = 0$ (Sunday) and the first decision is made on Monday morning. If we assume that the offers are distributed uniformly between 0 and 1, then their density is $f(w) = 1$ for $0 \le w \le 1$. With these data we find $F(\theta_{t+1}) = \theta_{t+1}$ and $A(\theta_{t+1}) = \frac{1}{2} - \frac{1}{2}\theta_{t+1}^2$ which gives rise to the following difference equation:

$$\theta_t = \frac{1}{2}(1 + \theta_{t+1}^2), \quad t = 0, 1, \ldots, 4, \quad \theta_N = 0.$$

Unfortunately, Maple is unable to solve this nonlinear difference equation in closed form with the `rsolve()` function:

```
> restart: # UniformDP.mws
> rsolve({theta(t) = (1/2)*(1+(theta(t+1)^2)),
  theta(5)=0},
  theta);
```

$$\text{rsolve}(\{\theta(t) = \frac{1}{2} + \frac{1}{2}\theta(t+1)^2,\ \theta(5) = 0\},\ \theta)$$

However, the θ_t values for $t = 0, 1, \ldots, N-1$ can easily be computed numerically as follows.

```
> restart: # UniformDP.mws (Part 2)
> Digits:=4:
> f:=w->1; # Uniform w
```

$$f := 1$$

```
> N:=5; theta[N]:=0;
```

$$N := 5$$

$$\theta_5 := 0$$

```
> for t from N-1 by -1 to 0 do:
> F[t+1]:=int(f,w=0..theta[t+1]):
> A[t+1]:=int(w*f,w=theta[t+1]..1):
> theta[t]:=evalf(theta[t+1]*F[t+1]+A[t+1]):
> od:
> seq(theta[t],t=0..N-1);
```

$$.7751,\ .7417,\ .6953,\ .6250,\ .5000$$

Thus, we see that at the start of the process, the professor expects to gain 0.7751 monetary units if he uses the optimal policy $\mu_t(x_t)$ that is prescribed in (1.6).

1.7.1 The Infinite-Stage Problem

What happens if the horizon N is large? Does the sequence θ_t approach a limit for large N? We now examine these issues.

First note that for large N, we must discount the future rewards, or the threshold levels would approach the maximum possible value attainable by the random variable representing the offer. For example, for the uniform example we would have $\theta_t \to 1$ as N gets large. This would be a strange result since in that case the optimal policy would *not* allow accepting any offers.

Now using discounting and assuming that $1/(1+r)$ is the discount factor per period, we can write $V_t(x_t) = \max\{x_t, (1+r)^{-1}E[V_{t+1}(w_t)]\}$, $t = 0, 1, \ldots, N-1$, with $V_N(x_N) = x_N$, which gives $\theta_t = (1+r)^{-1}E[V_{t+1}(w_t)]$. It can then be shown using induction that [Ber95, p. 161]

$$V_t(x) \geq V_{t+1}(x) \quad \text{for all } x \geq 0 \text{ and } t.$$

Since $\theta_t = (1+r)^{-1}E[V_{t+1}(w)]$, we obtain $\theta_t \geq \theta_{t+1}$. This means that as the end of horizon gets one period closer, the professor would be willing to accept a lower offer since he has one less chance of getting an improved offer. For the discounted case, the nonlinear difference equation assumes the form

$$\theta_t = (1+r)^{-1}F(\theta_{t+1}) + (1+r)^{-1}A(\theta_{t+1}), \quad t = 0, 1, \ldots, N-1.$$

Now, $0 \leq F(\theta) \leq 1$ for all $\theta \geq 0$ and $0 \leq \int_{\theta_{t+1}}^{\infty} w\, dF(w) \leq E(\text{offer}) < \infty$ for all t. Thus, using the property that $\theta_t \geq \theta_{t+1}$, we obtain an algebraic equation in terms of a constant $\bar{\theta}$ whose solution would give us the limiting value of θ:

$$(1+r)\bar{\theta} = \bar{\theta}F(\bar{\theta}) + \int_{\bar{\theta}}^{\infty} w\, dF(w).$$

For the uniformly distributed offers with a discount rate of $r = 0.10$, the limiting result is obtained simply as the solution of a quadratic equation:

```
> restart: # UniformDP.mws (Part 2)
> r:=0.10;
```

$$r := .10$$

```
> F:=theta;
```

$$F := \theta$$

```
> A:=int(w,w=theta..1);
```

$$A := \frac{1}{2} - \frac{1}{2}\theta^2$$

```
> Asy:=(1+r)*theta=theta*F+A;
```

$$Asy := 1.10\,\theta = \frac{1}{2}\theta^2 + \frac{1}{2}$$

```
> solve(Asy,theta);
```

$$.6417424305,\ 1.558257570$$

Thus, we obtain $\bar{\theta} = 0.641$ as the number of opportunities (periods) to receive offers approaches infinity.

Let us now consider a more general case where the offers have a beta distribution so that the density $f(w)$ assumes the form

$$f(w) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} w^{a-1}(1-w)^{b-1}, \quad \text{for } 0 \le w \le 1$$

where $\Gamma(y)$ is the gamma function defined by $\Gamma(y) = \int_0^\infty u^{y-1}e^{-u}\,du$. It is well known that when y is a positive integer, we obtain $\Gamma(y) = (y-1)!$, and that when $a = b = 1$, the beta density reduces to the uniform so that $f(w) = 1$ for $0 \le w \le 1$.

The Maple output for $a = 5$ and $b = 2$ follows where for $N = 5$, we compute the threshold levels θ_t, $t = 0, 1, \ldots, 4$.

```
> restart: # BetaDP.mws
> Digits:=3;
```

$$Digits := 3$$

```
> a:=5; b:=2;
```

$$a := 5$$

$$b := 2$$

```
> Mean:=a/(a+b);
```

$$Mean := \frac{5}{7}$$

```
> f:=GAMMA(a+b)/(GAMMA(a)*GAMMA(b))*
  w^(a-1)*(1-w)^(b-1);
```

$$f := 30\,w^4\,(1-w)$$

```
> N:=5; theta[N]:=0;
```

$$N := 5$$

$$\theta_5 := 0$$

```
> for t from N-1 by -1 to 0 do;
> F[t+1]:=int(f,w=0..theta[t+1]);
> A[t+1]:=int(w*f,w=theta[t+1]..1);
> theta[t]:=evalf(theta[t+1]*F[t+1]+A[t+1]);
> od:
> seq(theta[t],t=0..N);
```

$$.838,\ .830,\ .806,\ .765,\ .714,\ 0$$

We find the limiting value $\bar{\theta} = 0.704$ (with $r = 0.10$) as the unique real valued root of a polynomial of degree 5:

```
> restart: # BetaDP.mws (Part 2)
> Digits:=3;
```

$$Digits := 3$$

```
> r:=0.10;
```

$$r := .10$$

```
> a:=5;b:=2;
```

$$a := 5$$

$$b := 2$$

```
> f:=GAMMA(a+b)/(GAMMA(a)*GAMMA(b))
  *w^(a-1)*(1-w)^(b-1);
```

$$f := 30\,w^4\,(1-w)$$

```
> F:=int(f,w=0..theta);
```

$$F := -5\,\theta^6 + 6\,\theta^5$$

```
> A:=int(w*f,w=theta..1);
```

$$A := \frac{5}{7} + \frac{30}{7}\,\theta^7 - 5\,\theta^6$$

```
> Asy:=(1+r)*theta=theta*F+A;
```

$$Asy := 1.10\,\theta = \theta\,(-5\,\theta^6 + 6\,\theta^5) + \frac{5}{7} + \frac{30}{7}\,\theta^7 - 5\,\theta^6$$

```
> fsolve(Asy,theta);
```

$$.704$$

Compared to the limiting case $\bar{\theta} = 0.641$ obtained for uniformly distributed offers (with mean 1/2), the beta-distributed offers with a mean of $a/(a+b) = 5/7$ have higher threshold levels. This should be expected since with the higher mean of the latter case, the decision maker can afford to be picky and reject offers that are not high enough.

1.8 Summary

Dynamic programming (DP) is useful in solving different types of sequential decision problems with Bellman's principle of optimality. We started this chapter by introducing a simple discrete-time, discrete-state sequential decision problem known as the stagecoach problem. Sequential decision problems with a quadratic performance criterion and a linear system normally give rise to closed-form solutions that may require an iterative process of solving a sequence of recursive equations. Using Maple's symbolic manipulation capabilities, we were able to develop closed-form solutions for a large class of such problems. A finite horizon

workforce planning problem with constraints was solved with the help of Maple's `piecewise()` function. A gambling problem modeled using stochastic DP was solved that gave rise to a myopic policy. Finally, optimal stopping problems were considered where the stopping boundary was computed by solving a nonlinear difference equation numerically.

1.9 Exercises

1. Consider the following road network where a driver can travel only on the one-way streets as indicated by arrows. The driver starts at the intersection labelled "A" and must reach the final destination at the intersection labelled "H". The travel distances between any two intersections are indicated next to the arrows. Define the states and decisions and find the optimal route for the driver.

$$\begin{array}{ccccccc} A & \longrightarrow 5 & D & \longrightarrow 3 & E & \longrightarrow 7 & H \\ \downarrow 3 & & \uparrow 6 & & \downarrow 1 & & \uparrow 2 \\ B & \longrightarrow 8 & C & \longrightarrow 2 & F & \longrightarrow 3 & G \end{array}$$

2. Suppose we are given a positive quantity x. Use dynamic programming to divide x into n parts in such a way that the product of the n parts is maximized.

3. Consider the following optimization problem:

$$\min J = \sum_{i=0}^{N} (qx_i^2 + ru_i^2) + qx_3^2$$

subject to the state dynamics

$$x_{i+1} = ax_i + bu_i, i = 0, \ldots, N-1$$

where $N = 2, q = 2, r = 4, a = 1$ and $b = 0.5$.

   (a) Use dynamic programming to solve this problem.

   (b) The same problem appeared as a nonlinear programming problem in Exercise **??** of Chapter **??**; see page **??**. Comment on the nature of the solution produced by the two approaches.

4. Consider the problem of minimizing $J = \sum_{t=0}^{\infty}[u_t^2 + (x_t - u_t)^2]$ subject to the state dynamics $x_{t+1} = a(x_t - u_t)$ with $0 < a < 1$ and $0 \leq u_n \leq x_n$, $n = 0, 1, 2, \ldots$. Find the optimal policy and the value function for this infinite-stage problem.

5. Use the Hamilton-Jacobi-Bellman equation to minimize the performance criterion $J = \frac{1}{4}x^2(T) + \int_0^T \frac{1}{4}u^2(t)\,dt$ subject to the state dynamics $x'(t) = x(t) + u(t)$ where $T < \infty$.

6. Suppose we need to purchase a particular commodity within the next N days. Successive prices w_t for this commodity are random and i.i.d. with density $f(w_t)$. Determine an optimal policy to purchase this commodity in order to minimize the expected cost assuming that w_t are distributed uniformly between 0 and 1.

7. Consider a machine whose probability of failure depends on the number of items it has produced. Assume that if the machine fails while producing an item, there is a repair cost of c_f. The machine can be overhauled before it fails at a cost of $c_o < c_f$. If the machine is repaired or overhauled, then it is considered good as new. After the production of each unit we may decide to overhaul the machine or attempt to continue with the production of the next unit.

   Define $V_n(x)$ to be the minimum expected cost of producing n more units when the machine has already produced x parts. Let $p(x)$ be the probability of the machine failing while producing the $(x+1)$st unit *given* that it already has produced x units. Develop the DP functional equations for $V_n(x)$ and use the following data to compute $V_1(x), V_2(x), \ldots, V_6(x)$ for $x = 0, 1, 2, \ldots, 5$.

$p(0)$	$p(1)$	$p(2)$	$p(3)$	$p(4)$	$p(5)$	c_o	c_f
0	0.2	0.3	0.5	0.8	1.0	1	2

8. A set of items is called "group-testable" if for any of its subsets it is possible to perform a simultaneous (group) test on the subset with an outcome of "success" or "failure"; see Bar-Lev, Parlar and Perry [BLPP95]. The "success" outcome indicates that all the tested units are good, and the "failure" outcome indicates that at least one item in the tested subset is defective without knowing which (or how many) are defective.

   In industrial problems that involve, e.g., the identification of good electronic chips in a large batch that contain good and bad ones, chips of 100% quality cost much more than the chips of $100q\%$ quality where q is a positive constant that is usually greater than 0.9 but strictly less than 1.

   Let K be the set-up cost for each test, u be the group size decision variable, D be the requirement (demand) for 100% quality items, N be the number of $100q\%$ units purchased. Define the state variables x as the number of units of demand that is yet to be satisfied, y be the number of untested units. Let q be the probability that a given unit in the group is "good" and π be the unit cost of having a shortage of 100% quality items after the testing ends.

   Define $V(x, y)$ as the minimum expected cost (until the process terminates) when we are currently at state (x, y) and an optimal policy is followed until

termination. Write down the DP functional equation for the value function $V(x, y)$ and solve the problem using the following data:

D	N	K	π	q
3	5	20	100	0.95

References

[Arr71] K. J. Arrow. *Essays in the Theory of Risk-Bearing*. Markham, Chicago, 1971.

[BD58] R. Bellman and S. Dreyfus. Dynamic programming and the reliability of multicomponent devices. *Operations Research*, 6:200–206, 1958.

[Bec68] M. J. Beckmann. *Dynamic Programming of Economic Decisions*. Springer-Verlag, Berlin, 1968.

[Bel52] R. Bellman. On the theory of dynamic programming. *Proceedings of National Academy of Sciences*, 38:716–719, 1952.

[Bel53] R. Bellman. Dynamic progamming and a new formalism in the calculus of variations. *Proceedings of the National Academy of Sciences*, 39:1077–1082, 1953.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, N.J., 1957.

[Bel61] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, N.J., 1961.

[Ber95] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmont, Massachusetts, 1995.

[BH75] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Halstead, New York, 1975.

[BHN74] A. Bensoussan, E. G. Hurst, and B. Näslund. *Management Applications of Modern Control Theory*. North-Holland, Amsterdam, 1974.

[BK57a] R. Bellman and R. E. Kalaba. Dynamic programming and statistical communication theory. *Proceedings of the National Academy of Sciences, USA*, 43:749–751, 1957.

[BK57b] R. Bellman and R. E. Kalaba. On the role of dynamic programming in statistical communication theory. *IRE Transactions on Information Theory*, IT-3:197–203, 1957.

[BLPP95] S. K. Bar-Lev, M. Parlar, and D. Perry. Optimal sequential decisions for incomplete identification of group-testable items. *Sequential Analysis*, 14(1):41–57, 1995.

[Bre64] L. Breiman. Stopping-rule problems. In E. F. Beckenbach, editor, *Applied Combinatorial Mathematics*, pages 284–319. John Wiley, New York, 1964.

[CC81] L. Cooper and M. W. Cooper. *Introduction to Dynamic Programming*. Pergamon Press, New York, 1981.

[ÇP97] S. Çetinkaya and M. Parlar. Optimal nonmyopic gambling strategy for the generalized Kelly criterion. *Naval Research Logistics*, 44(4):639–654, 1997.

[CRS71] Y. S. Chow, H. Robbins, and D. Siegmund. *The Theory of Optimal Stopping*. Houghton Mifflin, Boston, 1971.

[Dan75] S. Danø. *Nonlinear and Dynamic Programming*. Springer-Verlag, New York, 1975.

[Edw62] W. Edwards. Dynamic decision theory and probabilistic information processing. *Human Factors*, 4:59–73, 1962.

[HL86] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. Holden-Day, Oakland, Calif., 4th edition, 1986.

[HMMS60] C. C. Holt, F. Modigliani, J. F. Muth, and H. A. Simon. *Planning Production, Inventories, and Work Force*. Prentice-Hall, Englewood Cliffs, N.J., 1960.

[How66] R. A. Howard. Dynamic programming. *Management Science*, 12:317–348, 1966.

[HS84] D. P. Heyman and M. J. Sobel. *Stochastic Models in Operations Research, Volume II: Stochastic Optimization*. McGraw-Hill, New York, 1984.

[Kel56] J. L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35:917–926, 1956.

[Kir70] D. E. Kirk. *Optimal Control Theory*. Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[KS81] M. I. Kamien and N. L. Schwartz. *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*. North-Holland, New York, 1981.

[LC78] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming. Part I: Basic Analytic and Computational Methods*. Marcel Dekker, New York, 1978.

[Leo74] B. Leonardz. *To Stop or Not to Stop: Some Elementary Optimal Stopping Problems with Economic Interpretations*. Almqvist and Wiksell, Stockholm, 1974.

[LM76a] S. A. Lippman and J. J. McCall. The economics of job search: A survey. Part I. *Economic Inquiry*, 14:155–189, 1976.

[LM76b] S. A. Lippman and J. J. McCall. The economics of job search: A survey. Part II. *Economic Inquiry*, 14:347–368, 1976.

[Mos68] J. Mossin. Optimal multi-period portfolio policies. *Journal of Business*, 41:215–229, 1968.

[Nem66] G. L. Nemhauser. *Introduction to Dynamic Programming*. John Wiley, New York, 1966.

[Par82] M. Parlar. A decomposition technique for an optimal control problem with "PQDZ" cost and bounded controls. *IEEE Transactions on Automatic Control*, AC-27(4):947–951, 1982.

[PG89] M. Parlar and Y. Gerchak. Control of a production system with variable yield and random demand. *Computers and Operations Research*, 16(4):315–324, 1989.

[PR92] M. Parlar and R. Rempała. Stochastic inventory problem with piecewise quadratic holding cost function containing a cost-free interval. *Journal of Optimization Theory and Applications*, 75(1):133–153, 1992.

[Ros83] S. M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

[Sca60] H. Scarf. The optimality of (S, s) policies in the dynamic inventory problem. In K. J. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences*. Stanford University Press, Stanford, Calif., 1960.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[SW77] A. P. Sage and C. C. White. *Optimum Systems Control*. Prentice Hall, Englewood Cliffs, N.J., 2nd edition, 1977.

[SYF59] M. Sasieni, A. Yaspan, and L. Friedman. *Operations Research: Methods and Problems*. John Wiley, New York, 1959.

[Vei65] A. F. Veinott. Optimal policy for a multi-product, dynamic non-stationary inventory problem. *Management Science*, 12:206–222, 1965.

[Wal47] A. Wald. *Sequential Analysis*. John Wiley, New York, 1947. Republished by Dover in 1973.

Index