13

Nonlinear Programming

The fundamental role of linear programming in OR is accurately reflected by the fact that it is the focus of a *third* of this book. A key assumption of linear programming is that *all its functions* (objective function and constraint functions) are linear. Although this assumption essentially holds for numerous practical problems, it frequently does not hold. In fact, many economists have found that some degree of nonlinearity is the rule and not the exception in economic planning problems.[1] Therefore, it often is necessary to deal directly with nonlinear programming problems, so we turn our attention to this important area.

In one general form,[2] the *nonlinear programming problem* is to find $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ so as to

$$\text{Maximize} \qquad f(\mathbf{x}),$$

subject to

$$g_i(\mathbf{x}) \le b_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

and

$$\mathbf{x} \ge \mathbf{0},$$

where $f(\mathbf{x})$ and the $g_i(\mathbf{x})$ are given functions of the n decision variables.[3]

No algorithm that will solve *every* specific problem fitting this format is available. However, substantial progress has been made for some important special cases of this problem by making various assumptions about these functions, and research is continuing very actively. This area is a large one, and we do not have the space to survey it completely. However, we do present a few sample applications and then introduce some of the basic ideas for solving certain important types of nonlinear programming problems.

Both Appendixes 2 and 3 provide useful background for this chapter, and we recommend that you review these appendixes as you study the next few sections.

[1] For example, see W. J. Baumol and R. C. Bushnell, “Error Produced by Linearization in Mathematical Programming,” *Econometrica,* **35:** 447–471, 1967.

[2] The other *legitimate forms* correspond to those for *linear programming* listed in Sec. 3.2. Section 4.6 describes how to convert these other forms to the form given here.

[3] For simplicity, we assume throughout the chapter that *all* these functions either are *differentiable* everywhere or are *piecewise linear functions* (discussed in Secs. 13.1 and 13.8).

13.1 SAMPLE APPLICATIONS

The following examples illustrate a few of the many important types of problems to which nonlinear programming has been applied.

The Product-Mix Problem with Price Elasticity

In *product-mix* problems, such as the Wyndor Glass Co. problem of Sec. 3.1, the goal is to determine the optimal mix of production levels for a firm's products, given limitations on the resources needed to produce those products, in order to maximize the firm's total profit. In some cases, there is a fixed unit profit associated with each product, so the resulting objective function will be linear. However, in many product-mix problems, certain factors introduce *nonlinearities* into the objective function.

For example, a large manufacturer may encounter *price elasticity,* whereby the amount of a product that can be sold has an inverse relationship to the price charged. Thus, the *price-demand curve* for a typical product might look like the one shown in Fig. 13.1, where $p(x)$ is the price required in order to be able to sell x units. The firm's profit from producing and selling x units of the product then would be the sales revenue, $xp(x)$, minus the production and distribution costs. Therefore, if the unit cost for producing and distributing the product is fixed at c (see the dashed line in Fig. 13.1), the firm's profit from producing and selling x units is given by the nonlinear function

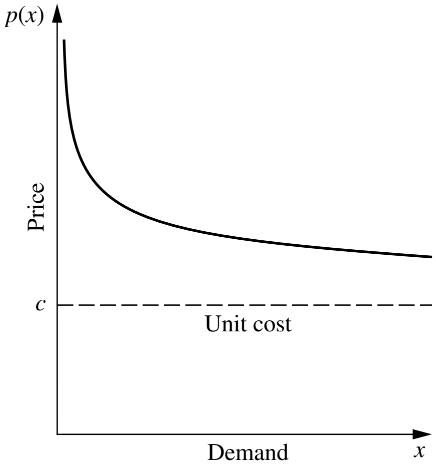
$$P(x) = xp(x) - cx,$$

as plotted in Fig. 13.2. If *each* of the firm's n products has a similar profit function, say, $P_j(x_j)$ for producing and selling x_j units of product j ($j = 1, 2, \dots, n$), then the overall objective function is

$$f(\mathbf{x}) = \sum_{j=1}^{n} P_j(x_j),$$

a sum of nonlinear functions.

FIGURE 13.1
Price-demand curve.



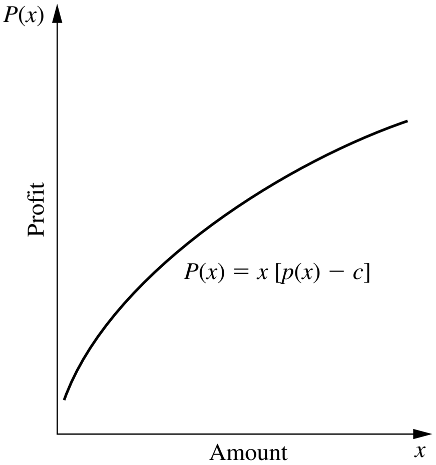


FIGURE 13.2
Profit function.

Another reason that nonlinearities can arise in the objective function is the fact that the *marginal cost* of producing another unit of a given product varies with the production level. For example, the marginal cost may decrease when the production level is increased because of a *learning-curve effect* (more efficient production with more experience). On the other hand, it may increase instead, because special measures such as overtime or more expensive production facilities may be needed to increase production further.

Nonlinearities also may arise in the $g_i(\mathbf{x})$ constraint functions in a similar fashion. For example, if there is a budget constraint on total production cost, the cost function will be nonlinear if the marginal cost of production varies as just described. For constraints on the other kinds of resources, $g_i(\mathbf{x})$ will be nonlinear whenever the use of the corresponding resource is not strictly proportional to the production levels of the respective products.

The Transportation Problem with Volume Discounts on Shipping Costs

As illustrated by the P & T Company example in Sec. 8.1, a typical application of the transportation problem is to determine an optimal plan for shipping goods from various sources to various destinations, given supply and demand constraints, in order to minimize total shipping cost. It was assumed in Chap. 8 that the *cost per unit shipped* from a given source to a given destination is *fixed,* regardless of the amount shipped. In actuality, this cost may not be fixed. *Volume discounts* sometimes are available for large shipments, so that the *marginal cost* of shipping one more unit might follow a pattern like the one shown in Fig. 13.3. The resulting cost of shipping x units then is given by a *nonlinear* function $C(x)$, which is a *piecewise linear function* with slope equal to the marginal cost, like the one shown in Fig. 13.4. [The function in Fig. 13.4 consists of a line segment with slope 6.5 from (0, 0) to (0.6, 3.9), a second line segment with slope 5 from (0.6, 3.9) to (1.5, 8.4), a third line segment with slope 4 from (1.5, 8.4) to (2.7, 13.2), and

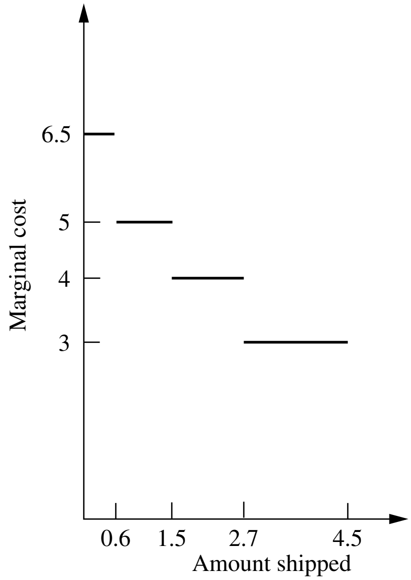
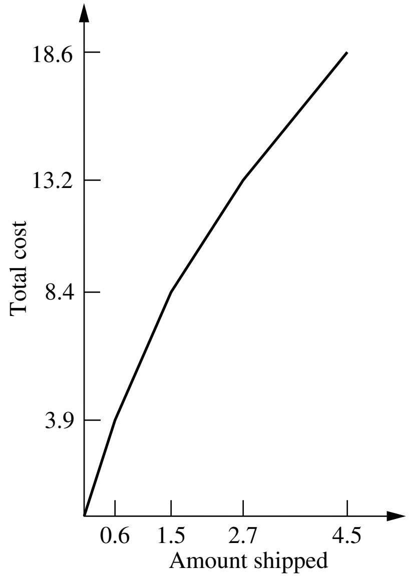


FIGURE 13.3
Marginal shipping cost.

FIGURE 13.4
Shipping cost function.



a fourth line segment with slope 3 from (2.7, 13.2) to (4.5, 18.6).] Consequently, if *each* combination of source and destination has a similar shipping cost function, so that the cost of shipping x_{ij} units from source i ($i = 1, 2, \dots, m$) to destination j ($j = 1, 2, \dots, n$) is given by a nonlinear function $C_{ij}(x_{ij})$, then the overall objective function to be *minimized* is

$$f(\mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij}(x_{ij}).$$

Even with this nonlinear objective function, the constraints normally are still the special linear constraints that fit the transportation problem model in Sec. 8.1.

Portfolio Selection with Risky Securities

It now is common practice for professional managers of large stock portfolios to use computer models based partially on nonlinear programming to guide them. Because investors are concerned about both the *expected return* (gain) and the *risk* associated with their investments, nonlinear programming is used to determine a portfolio that, under certain assumptions, provides an optimal trade-off between these two factors. This approach is based largely on path-breaking research done by Harry Markowitz and William Sharpe that helped them win the 1990 Nobel Prize in Economics.

A nonlinear programming model can be formulated for this problem as follows. Suppose that n stocks (securities) are being considered for inclusion in the portfolio, and let the decision variables x_j ($j = 1, 2, \dots, n$) be the number of shares of stock j to be included. Let μ_j and σ_{jj} be the (estimated) *mean* and *variance,* respectively, of the return on each share of stock j, where σ_{jj} measures the risk of this stock. For $i = 1, 2, \dots, n$ ($i \neq j$), let σ_{ij} be the *covariance* of the return on one share each of stock i and stock j. (Because it would be difficult to estimate all the σ_{ij} values, the usual approach is to make certain assumptions about market behavior that enable us to calculate σ_{ij} directly from σ_{ii} and σ_{jj}.) Then the expected value $R(\mathbf{x})$ and the variance $V(\mathbf{x})$ of the total return from the entire portfolio are

$$R(\mathbf{x}) = \sum_{j=1}^{n} \mu_j x_j$$

and

$$V(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j,$$

where $V(\mathbf{x})$ measures the risk associated with the portfolio. One way to consider the trade-off between these two factors is to use $V(\mathbf{x})$ as the objective function to be minimized and then impose the constraint that $R(\mathbf{x})$ must be no smaller than the minimum acceptable expected return. The complete nonlinear programming model then would be

$$\text{Minimize} \qquad V(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j,$$

subject to

$$\sum_{j=1}^{n} \mu_j x_j \geq L$$

$$\sum_{j=1}^{n} P_j x_j \leq B$$

and

$$x_j \geq 0, \qquad \text{for } j = 1, 2, \ldots, n,$$

where L is the minimum acceptable expected return, P_j is the price for each share of stock *j,* and B is the amount of money budgeted for the portfolio.

One drawback of this formulation is that it is relatively difficult to choose an appropriate value for L for obtaining the best trade-off between $R(\mathbf{x})$ and $V(\mathbf{x})$. Therefore, rather than stopping with one choice of L, it is common to use a *parametric* (nonlinear) programming approach to generate the optimal solution as a function of L over a wide range of values of L. The next step is to examine the values of $R(\mathbf{x})$ and $V(\mathbf{x})$ for these solutions that are optimal for some value of L and then to choose the solution that seems to give the best trade-off between these two quantities. This procedure often is referred to as generating the solutions on the *efficient frontier* of the two-dimensional graph of $(R(\mathbf{x}), V(\mathbf{x}))$ points for feasible $\mathbf{x}$. The reason is that the $(R(\mathbf{x}), V(\mathbf{x}))$ point for an optimal $\mathbf{x}$ (for some L) lies on the *frontier* (boundary) of the feasible points. Furthermore, each optimal $\mathbf{x}$ is *efficient* in the sense that no other feasible solution is at least equally good with one measure (R or V) and strictly better with the other measure (smaller V or larger R).

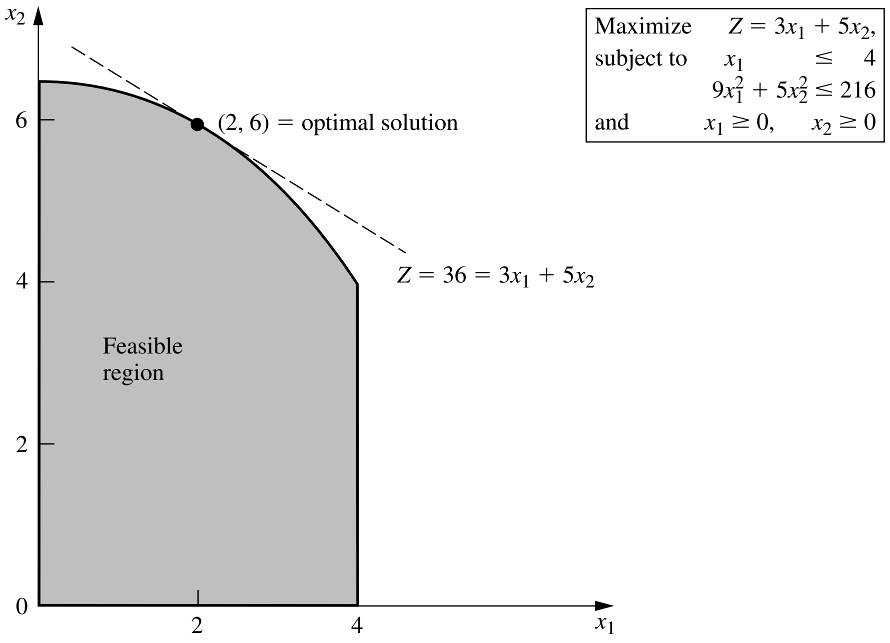
13.2 GRAPHICAL ILLUSTRATION OF NONLINEAR PROGRAMMING PROBLEMS

When a nonlinear programming problem has just one or two variables, it can be represented graphically much like the Wyndor Glass Co. example for linear programming in Sec. 3.1. Because such a graphical representation gives considerable insight into the properties of optimal solutions for linear and nonlinear programming, let us look at a few examples. To highlight the difference between linear and nonlinear programming, we shall use some *nonlinear* variations of the Wyndor Glass Co. problem.

Figure 13.5 shows what happens to this problem if the only changes in the model shown in Sec. 3.1 are that both the second and the third functional constraints are replaced by the single nonlinear constraint $9x_1^2 + 5x_2^2 \leq 216$. Compare Fig. 13.5 with Fig. 3.3. The optimal solution still happens to be $(x_1, x_2) = (2, 6)$. Furthermore, it still lies on the boundary of the feasible region. However, it is *not* a corner-point feasible (CPF) solution. The optimal solution could have been a CPF solution with a different objective function (check $Z = 3x_1 + x_2$), but the fact that it need not be one means that we no longer have the tremendous simplification used in linear programming of limiting the search for an optimal solution to just the CPF solutions.

Now suppose that the linear constraints of Sec. 3.1 are kept unchanged, but the objective function is made nonlinear. For example, if

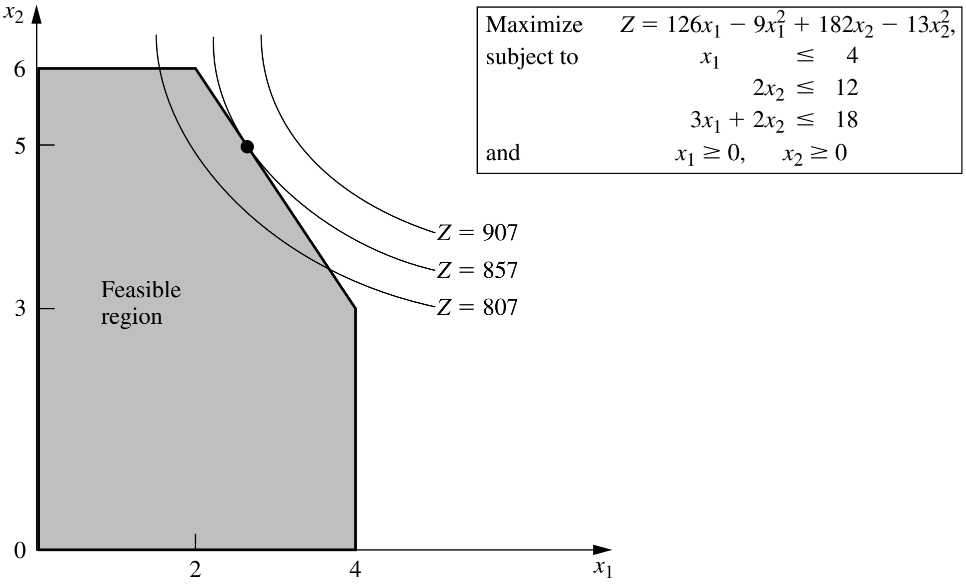
$$Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2,$$



$$
\begin{array}{lrl}
\text{Maximize} & Z = 3x_1 + 5x_2, & \\
\text{subject to} & x_1 & \leq 4 \\
 & 9x_1^2 + 5x_2^2 & \leq 216 \\
\text{and} & x_1 \geq 0, & x_2 \geq 0
\end{array}
$$

FIGURE 13.5
The Wyndor Glass Co. example with the nonlinear constraint $9x_1^2 + 5x_2^2 \leq 216$ replacing the original second and third functional constraints.

FIGURE 13.6
The Wyndor Glass Co. example with the original feasible region but with the nonlinear objective function $Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$ replacing the original objective function.



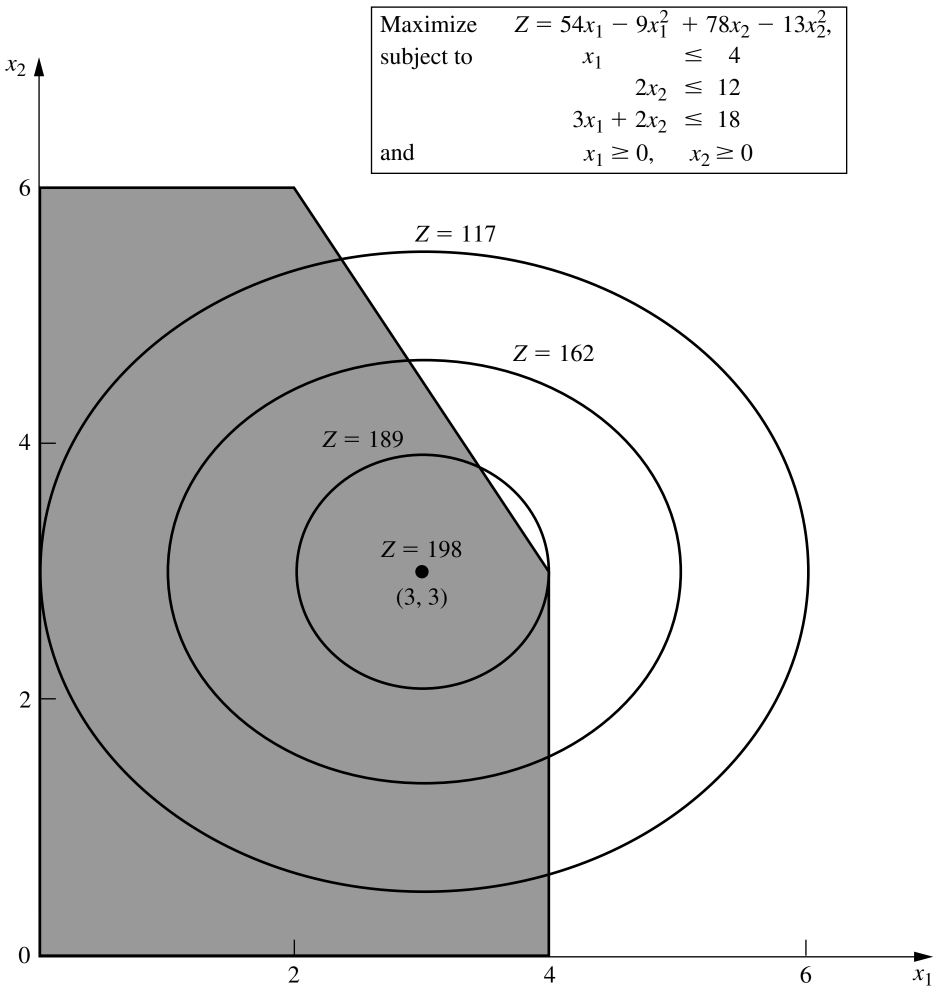
$$
\begin{array}{lrl}
\text{Maximize} & Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2, & \\
\text{subject to} & x_1 & \leq 4 \\
 & 2x_2 & \leq 12 \\
 & 3x_1 + 2x_2 & \leq 18 \\
\text{and} & x_1 \geq 0, & x_2 \geq 0
\end{array}
$$

then the graphical representation in Fig. 13.6 indicates that the optimal solution is $x_1 = \frac{8}{3}$, $x_2 = 5$, which again lies on the boundary of the feasible region. (The value of Z for this optimal solution is $Z = 857$, so Fig. 13.6 depicts the fact that the locus of all points with $Z = 857$ intersects the feasible region at just this one point, whereas the locus of points with any larger Z does not intersect the feasible region at all.) On the other hand, if

$$Z = 54x_1 - 9x_1^2 + 78x_2 - 13x_2^2,$$

then Fig. 13.7 illustrates that the optimal solution turns out to be $(x_1, x_2) = (3, 3)$, which lies *inside* the boundary of the feasible region. (You can check that this solution is optimal by using calculus to derive it as the unconstrained global maximum; because it also satisfies the constraints, it must be optimal for the constrained problem.) Therefore, a gen-

FIGURE 13.7
The Wyndor Glass Co. example with the original feasible region but with another nonlinear objective function, $Z = 54x_1 - 9x_1^2 + 78x_2 - 13x_2^2$, replacing the original objective function.



eral algorithm for solving similar problems needs to consider *all* solutions in the feasible region, not just those on the boundary.

Another complication that arises in nonlinear programming is that a *local* maximum need not be a *global* maximum (the overall optimal solution). For example, consider the function of a single variable plotted in Fig. 13.8. Over the interval $0 \leq x \leq 5$, this function has three local maxima—$x = 0$, $x = 2$, and $x = 4$—but only one of these—$x = 4$—is a *global maximum.* (Similarly, there are local minima at $x = 1$, 3, and 5, but only $x = 5$ is a *global minimum.*)

Nonlinear programming algorithms generally are unable to distinguish between a local maximum and a global maximum (except by finding another *better* local maximum). Therefore, it becomes crucial to know the conditions under which any local maximum is *guaranteed* to be a global maximum over the feasible region. You may recall from calculus that when we maximize an ordinary (doubly differentiable) function of a single variable $f(x)$ without any constraints, this guarantee can be given when

$$\frac{d^2f}{dx^2} \leq 0 \qquad \text{for all } x.$$

Such a function that is always "curving downward" (or not curving at all) is called a **concave** function.[1] Similarly, if $\leq$ is replaced by $\geq$, so that the function is always "curving upward" (or not curving at all), it is called a **convex** function.[2] (Thus, a *linear* function is both concave and convex.) See Fig. 13.9 for examples. Then note that Fig. 13.8 illustrates a function that is neither concave nor convex because it alternates between curving upward and curving downward.

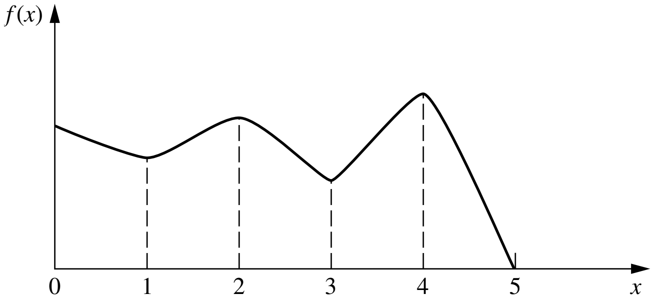
Functions of multiple variables also can be characterized as concave or convex if they always curve downward or curve upward. These intuitive definitions are restated in precise terms, along with further elaboration on these concepts, in Appendix 2. Appendix 2 also provides a convenient test for checking whether a function of two variables is concave, convex, or neither.

Here is a convenient way of checking this for a function of more than two variables when the function consists of a *sum* of smaller functions of just one or two variables each.

[1] Concave functions sometimes are referred to as *concave downward.*

[2] Convex functions sometimes are referred to as *concave upward.*

FIGURE 13.8
A function with several local maxima ($x = 0, 2, 4$), but only $x = 4$ is a global maximum.



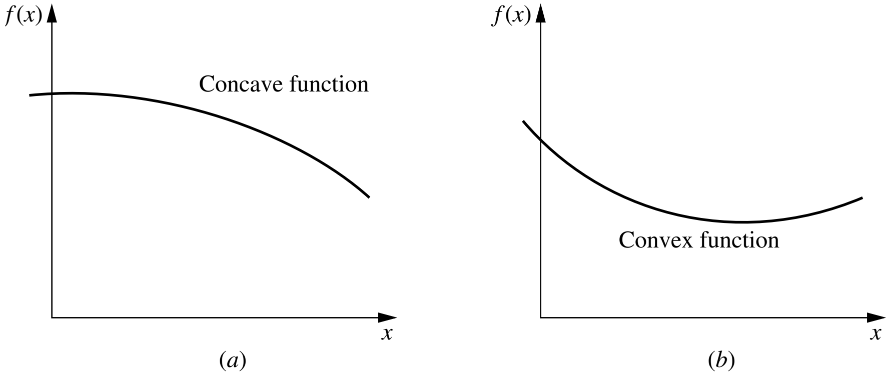


FIGURE 13.9
Examples of (*a*) a concave function and (*b*) a convex function.

If each smaller function is concave, then the overall function is concave. Similarly, the overall function is convex if each smaller function is convex.

To illustrate, consider the function

$$\begin{aligned} f(x_1, x_2, x_3) &= 4x_1 - x_1^2 - (x_2 - x_3)^2 \\ &= [4x_1 - x_1^2] + [-(x_2 - x_3)^2], \end{aligned}$$

which is the sum of the two smaller functions given in square brackets. The first smaller function $4x_1 - x_1^2$ is a function of the single variable x_1, so it can be found to be concave by noting that its second derivative is negative. The second smaller function $-(x_2 - x_3)^2$ is a function of just x_2 and x_3, so the test for functions of two variables given in Appendix 2 is applicable. In fact, Appendix 2 uses this particular function to illustrate the test and finds that the function is concave. Because both smaller functions are concave, the overall function $f(x_1, x_2, x_3)$ must be concave.

If a nonlinear programming problem has no constraints, the objective function being *concave* guarantees that a local maximum is a *global maximum.* (Similarly, the objective function being *convex* ensures that a local minimum is a *global minimum.*) If there are constraints, then one more condition will provide this guarantee, namely, that the *feasible region* is a **convex set.** As discussed in Appendix 2, a convex set is simply a set of points such that, for each pair of points in the collection, the entire line segment joining these two points is also in the collection. Thus, the feasible region for the original Wyndor Glass Co. problem (see Fig. 13.6 or 13.7) is a convex set. In fact, the feasible region for *any* linear programming problem is a convex set. Similarly, the feasible region in Fig. 13.5 is a convex set.

In general, the feasible region for a nonlinear programming problem is a convex set whenever all the $g_i(\mathbf{x})$ [for the constraints $g_i(\mathbf{x}) \le b_i$] are convex functions. For the example of Fig. 13.5, both of its $g_i(\mathbf{x})$ are convex functions, since $g_1(\mathbf{x}) = x_1$ (a linear function is automatically both concave and convex) and $g_2(\mathbf{x}) = 9x_1^2 + 5x_2^2$ (both $9x_1^2$ and $5x_2^2$ are convex functions so their sum is a convex function). These two convex $g_i(\mathbf{x})$ lead to the feasible region of Fig. 13.5 being a convex set.

Now let's see what happens when just one of these $g_i(\mathbf{x})$ is a concave function instead. In particular, suppose that the only change made in the example of Fig. 13.5 is that its nonlinear constraint is replaced by $8x_1 - x_1^2 + 14x_2 - x_2^2 \le 49$. Therefore, the new

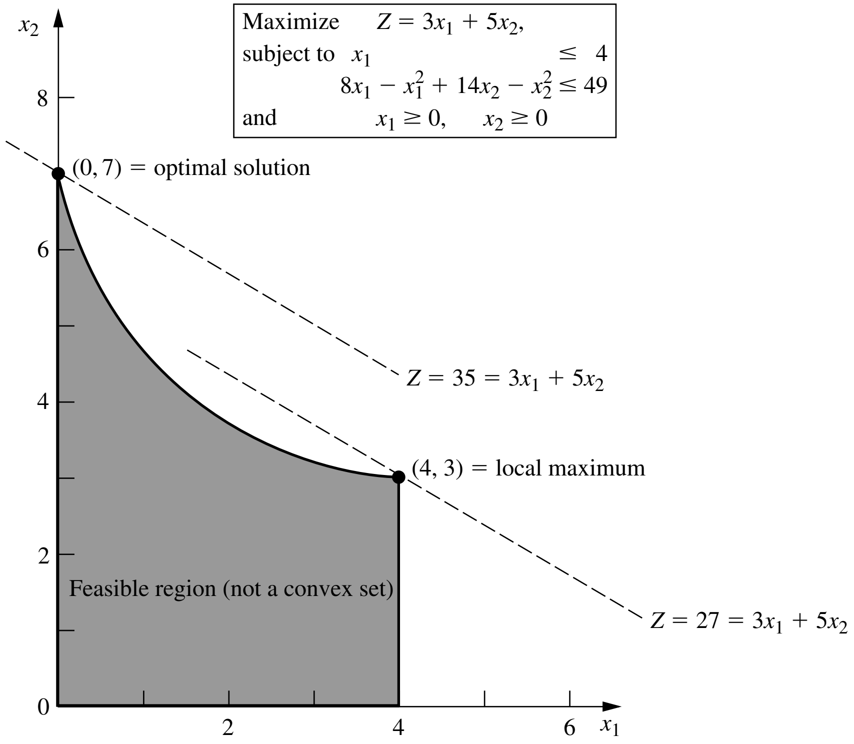


FIGURE 13.10
The Wyndor Glass Co. example with another nonlinear constraint, $8x_1 - x_1^2 + 14x_2 - x_2^2 \leq 49$, replacing the original second and third functional constraints.

$g_2(\mathbf{x}) = 8x_1 - x_1^2 + 14x_2 - x_2^2$, which is a concave function since both $8x_1 - x_1^2$ and $14x_2 - x_2^2$ are concave functions. The new feasible region shown in Fig. 13.10 is *not* a convex set. Why? Because this feasible region contains pairs of points, for example, (0, 7) and (4, 3), such that part of the line segment joining these two points is not in the feasible region. Consequently, we cannot guarantee that a local maximum is a global maximum. In fact, this example has two local maxima, (0, 7) and (4, 3), but only (0, 7) is a global maximum.

Therefore, to guarantee that a local maximum is a global maximum for a nonlinear programming problem with constraints $g_i(\mathbf{x}) \leq b_i$ $(i = 1, 2, \ldots, m)$ and $\mathbf{x} \geq \mathbf{0}$, the objective function $f(\mathbf{x})$ must be a *concave* function and each $g_i(\mathbf{x})$ must be a *convex* function. Such a problem is called a *convex programming problem,* which is one of the key types of nonlinear programming problems discussed in the next section.

13.3 TYPES OF NONLINEAR PROGRAMMING PROBLEMS

Nonlinear programming problems come in many different shapes and forms. Unlike the simplex method for linear programming, no single algorithm can solve all these different types of problems. Instead, algorithms have been developed for various individual *classes* (special types) of nonlinear programming problems. The most important classes are introduced briefly in this section. The subsequent sections then describe how some problems of these types can be solved.

Unconstrained Optimization

Unconstrained optimization problems have *no* constraints, so the objective is simply to

$$\text{Maximize} \qquad f(\mathbf{x})$$

over *all* values of $\mathbf{x} = (x_1, x_2, \dots, x_n)$. As reviewed in Appendix 3, the *necessary* condition that a particular solution $\mathbf{x} = \mathbf{x}^*$ be optimal when $f(\mathbf{x})$ is a differentiable function is

$$\frac{\partial f}{\partial x_j} = 0 \qquad \text{at } \mathbf{x} = \mathbf{x}^*, \text{ for } j = 1, 2, \dots, n.$$

When $f(\mathbf{x})$ is a *concave* function, this condition also is *sufficient,* so then solving for $\mathbf{x}^*$ reduces to solving the system of n equations obtained by setting the n partial derivatives equal to zero. Unfortunately, for *nonlinear* functions $f(\mathbf{x})$, these equations often are going to be *nonlinear* as well, in which case you are unlikely to be able to solve analytically for their simultaneous solution. What then? Sections 13.4 and 13.5 describe *algorithmic search procedures* for finding $\mathbf{x}^*$, first for $n = 1$ and then for $n > 1$. These procedures also play an important role in solving many of the problem types described next, where there are constraints. The reason is that many algorithms for *constrained* problems are designed so that they can focus on an *unconstrained* version of the problem during a portion of each iteration.

When a variable x_j does have a nonnegativity constraint $x_j \geq 0$, the preceding necessary and (perhaps) sufficient condition changes slightly to

$$\frac{\partial f}{\partial x_j} \begin{cases} \leq 0 & \text{at } \mathbf{x} = \mathbf{x}^*, \qquad \text{if } x_j^* = 0 \\ = 0 & \text{at } \mathbf{x} = \mathbf{x}^*, \qquad \text{if } x_j^* > 0 \end{cases}$$

for each such j. This condition is illustrated in Fig. 13.11, where the optimal solution for a problem with a single variable is at $x = 0$ even though the derivative there is negative rather than zero. Because this example has a concave function to be maximized subject to a nonnegativity constraint, having the derivative less than or equal to 0 at $x = 0$ is both a necessary and sufficient condition for $x = 0$ to be optimal.

A problem that has some nonnegativity constraints but no functional constraints is one special case ($m = 0$) of the next class of problems.

Linearly Constrained Optimization

Linearly constrained optimization problems are characterized by constraints that completely fit linear programming, so that *all* the $g_i(\mathbf{x})$ constraint functions are linear, but the objective function $f(\mathbf{x})$ is nonlinear. The problem is considerably simplified by having just one nonlinear function to take into account, along with a linear programming feasible region. A number of special algorithms based upon *extending* the simplex method to consider the nonlinear objective function have been developed.

One important special case, which we consider next, is quadratic programming.

Quadratic Programming

Quadratic programming problems again have linear constraints, but now the objective function $f(\mathbf{x})$ must be *quadratic.* Thus, the only difference between such a problem and a linear programming problem is that some of the terms in the objective function involve the *square* of a variable or the *product* of two variables.

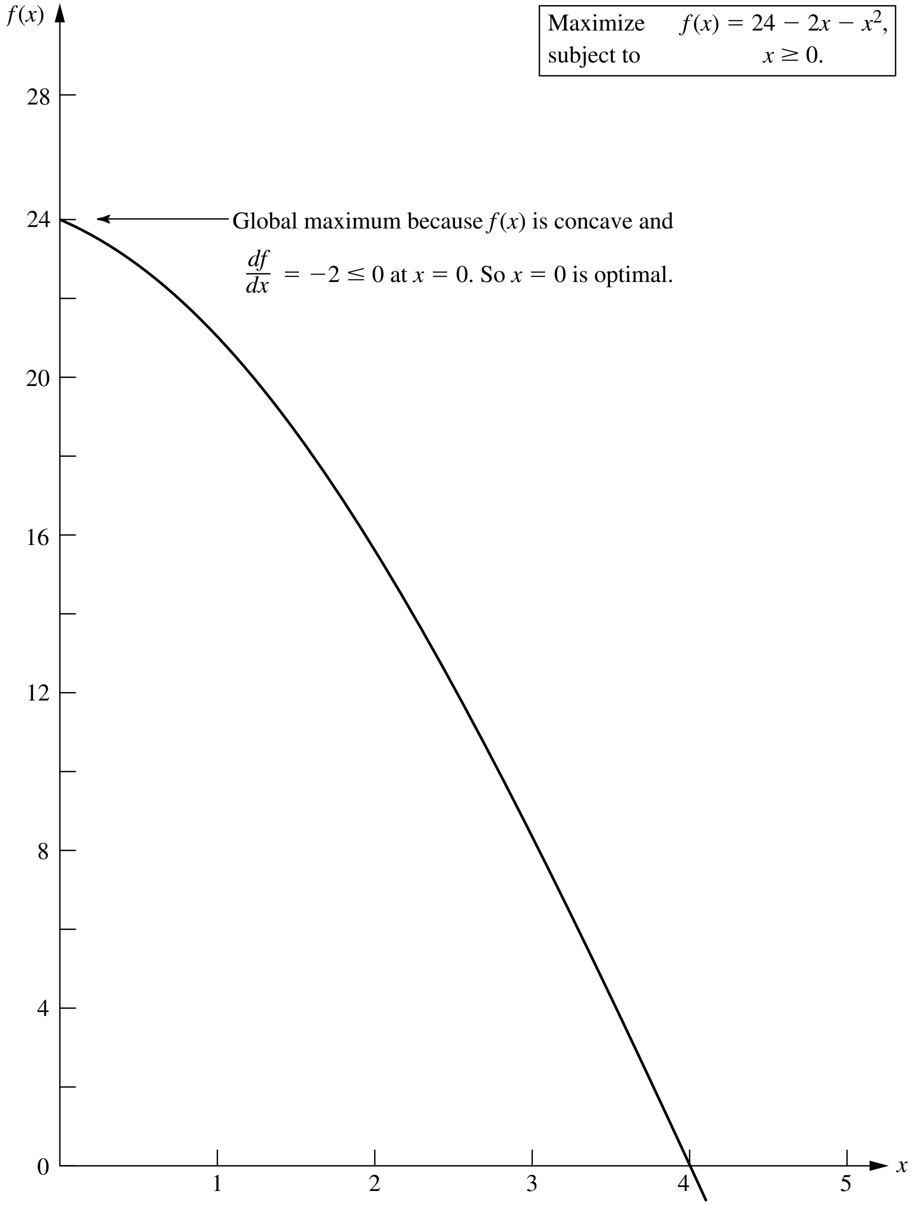


FIGURE 13.11
An example that illustrates how an optimal solution can lie at a point where a derivative is negative instead of zero, because that point lies at the boundary of a nonnegativity constraint.

Many algorithms have been developed for this case under the additional assumption that $f(\mathbf{x})$ is a concave function. Section 13.7 presents an algorithm that involves a direct extension of the simplex method.

Quadratic programming is very important, partially because such formulations arise naturally in many applications. For example, the problem of portfolio selection with risky securities described in Sec. 13.1 fits into this format. However, another major reason for its importance is that a common approach to solving general linearly constrained optimization problems is to solve a sequence of quadratic programming approximations.

Convex Programming

Convex programming covers a broad class of problems that actually encompasses as special cases all the preceding types when $f(\mathbf{x})$ is a concave function. The assumptions are that

1. $f(\mathbf{x})$ is a concave function.
2. Each $g_i(\mathbf{x})$ is a convex function.

As discussed at the end of Sec. 13.2, these assumptions are enough to ensure that a local maximum is a global maximum. You will see in Sec. 13.6 that the necessary and sufficient conditions for such an optimal solution are a natural generalization of the conditions just given for *unconstrained optimization* and its extension to include *nonnegativity constraints.* Section 13.9 then describes algorithmic approaches to solving convex programming problems.

Separable Programming

Separable programming is a special case of convex programming, where the one additional assumption is that

3. All the $f(\mathbf{x})$ and $g_i(\mathbf{x})$ functions are separable functions.

A **separable function** is a function where *each term* involves just a *single variable,* so that the function is separable into a sum of functions of individual variables. For example, if $f(\mathbf{x})$ is a separable function, it can be expressed as

$$f(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j),$$

where each $f_j(x_j)$ function includes only the terms involving just x_j. In the terminology of linear programming (see Sec. 3.3), separable programming problems satisfy the assumption of additivity but not the assumption of proportionality (for nonlinear functions).

To illustrate, the objective function considered in Fig. 13.6,

$$f(x_1, x_2) = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$$

is a separable function because it can be expressed as

$$f(x_1, x_2) = f_1(x_1) + f_2(x_2)$$

where $f_1(x_1) = 126x_1 - 9x_1^2$ and $f_2(x_2) = 182x_2 - 13x_2^2$ are each a function of a single variable—x_1 and x_2, respectively. By the same reasoning, you can verify that the objective function considered in Fig. 13.7 also is a separable function.

It is important to distinguish separable programming problems from other convex programming problems, because any such problem can be closely approximated by a linear programming problem so that the extremely efficient simplex method can be used. This approach is described in Sec. 13.8. (For simplicity, we focus there on the *linearly constrained* case where the special approach is needed only on the objective function.)

Nonconvex Programming

Nonconvex programming encompasses all nonlinear programming problems that do not satisfy the assumptions of convex programming. Now, even if you are successful in finding a *local maximum,* there is no assurance that it also will be a *global maximum.* Therefore, there is no algorithm that will guarantee finding an optimal solution for all such problems. However, there do exist some algorithms that are relatively well suited for finding local maxima, especially when the forms of the nonlinear functions do not deviate too strongly from those assumed for convex programming. One such algorithm is presented in Sec. 13.10.

However, certain specific types of nonconvex programming problems can be solved without great difficulty by special methods. Two especially important such types are discussed briefly next.

Geometric Programming

When we apply nonlinear programming to engineering design problems, the objective function and the constraint functions frequently take the form

$$g(\mathbf{x}) = \sum_{i=1}^{N} c_i P_i(\mathbf{x}),$$

where

$$P_i(\mathbf{x}) = x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}, \qquad \text{for } i = 1, 2, \ldots, N.$$

In such cases, the c_i and a_{ij} typically represent physical constants, and the x_j are design variables. These functions generally are neither convex nor concave, so the techniques of convex programming cannot be applied directly to these *geometric programming* problems. However, there is one important case where the problem can be transformed to an equivalent convex programming problem. This case is where *all* the c_i coefficients in each function are strictly positive, so that the functions are *generalized positive polynomials*—(now called **posynomials**)—and the objective function is to be minimized. The equivalent convex programming problem with decision variables $y_1, y_2, \ldots, y_n$ is then obtained by setting

$$x_j = e^{y_j}, \qquad \text{for } j = 1, 2, \ldots, n$$

throughout the original model, so now a convex programming algorithm can be applied. Alternative solution procedures also have been developed for solving these *posynomial programming* problems, as well as for geometric programming problems of other types.[1]

Fractional Programming

Suppose that the objective function is in the form of a *fraction,* i.e., the ratio of two functions,

$$\text{Maximize} \qquad f(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}.$$

[1] R. J. Duffin, E. L. Peterson, and C. M. Zehner, *Geometric Programming,* Wiley, New York, 1967; C. Beightler and D. T. Phillips, *Applied Geometric Programming,* Wiley, New York, 1976.

Such *fractional programming* problems arise, e.g., when one is maximizing the ratio of output to person-hours expended (productivity), or profit to capital expended (rate of return), or expected value to standard deviation of some measure of performance for an investment portfolio (return/risk). Some special solution procedures[1] have been developed for certain forms of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$.

When it can be done, the most straightforward approach to solving a fractional programming problem is to transform it to an equivalent problem of a standard type for which effective solution procedures already are available. To illustrate, suppose that $f(\mathbf{x})$ is of the *linear fractional programming* form

$$f(\mathbf{x}) = \frac{\mathbf{cx} + c_0}{\mathbf{dx} + d_0},$$

where $\mathbf{c}$ and $\mathbf{d}$ are row vectors, $\mathbf{x}$ is a column vector, and c_0 and d_0 are scalars. Also assume that the constraint functions $g_i(\mathbf{x})$ are linear, so that the constraints in matrix form are $\mathbf{Ax} \le \mathbf{b}$ and $\mathbf{x} \ge \mathbf{0}$.

Under mild additional assumptions, we can transform the problem to an equivalent *linear programming* problem by letting

$$\mathbf{y} = \frac{\mathbf{x}}{\mathbf{dx} + d_0} \qquad \text{and} \qquad t = \frac{1}{\mathbf{dx} + d_0},$$

so that $\mathbf{x} = \mathbf{y}/t$. This result yields

$$\text{Maximize} \qquad Z = \mathbf{cy} + c_0 t,$$

subject to

$$\begin{aligned} \mathbf{Ay} - \mathbf{b}t &\le \mathbf{0}, \\ \mathbf{dy} + d_0 t &= 1, \end{aligned}$$

and

$$\mathbf{y} \ge \mathbf{0}, \qquad t \ge 0,$$

which can be solved by the simplex method. More generally, the same kind of transformation can be used to convert a fractional programming problem with concave $f_1(\mathbf{x})$, convex $f_2(\mathbf{x})$, and convex $g_i(\mathbf{x})$ to an equivalent convex programming problem.

The Complementarity Problem

When we deal with quadratic programming in Sec. 13.7, you will see one example of how solving certain nonlinear programming problems can be reduced to solving the complementarity problem. Given variables $w_1, w_2, \ldots, w_p$ and $z_1, z_2, \ldots, z_p$, the **complementarity problem** is to find a *feasible* solution for the set of constraints

$$\mathbf{w} = F(\mathbf{z}), \qquad \mathbf{w} \ge \mathbf{0}, \qquad \mathbf{z} \ge \mathbf{0}$$

[1] The pioneering work on fractional programming was done by A. Charnes and W. W. Cooper, "'Programming with Linear Fractional Functionals," *Naval Research Logistics Quarterly,* **9:** 181–186, 1962. Also see S. Schaible, "A Survey of Fractional Programming," in S. Schaible and W. T. Ziemba (eds.), *Generalized Concavity in Optimization and Economics*, Academic Press, New York, 1981, pp. 417–440.

that also satisfies the **complementarity constraint**

$$\mathbf{w}^T\mathbf{z} = 0.$$

Here, **w** and **z** are column vectors, F is a given vector-valued function, and the superscript T denotes the transpose (see Appendix 4). The problem has no objective function, so technically it is not a full-fledged nonlinear programming problem. It is called the complementarity problem because of the complementary relationships that either

$$w_i = 0 \qquad \text{or} \qquad z_i = 0 \qquad \text{(or both)} \qquad \text{for each } i = 1, 2, \ldots, p.$$

An important special case is the **linear complementarity problem,** where

$$F(\mathbf{z}) = \mathbf{q} + \mathbf{M}\mathbf{z},$$

where **q** is a given column vector and **M** is a given $p \times p$ matrix. Efficient algorithms have been developed for solving this problem under suitable assumptions[1] about the properties of the matrix **M**. One type involves pivoting from one basic feasible (BF) solution to the next, much like the simplex method for linear programming.

In addition to having applications in nonlinear programming, complementarity problems have applications in game theory, economic equilibrium problems, and engineering equilibrium problems.

13.4 ONE-VARIABLE UNCONSTRAINED OPTIMIZATION

We now begin discussing how to solve some of the types of problems just described by considering the simplest case—*unconstrained optimization* with just a single variable x ($n = 1$), where the differentiable function $f(x)$ to be maximized is *concave.*[2] Thus, the *necessary and sufficient condition* for a particular solution $x = x^*$ to be optimal (a global maximum) is

$$\frac{df}{dx} = 0 \qquad \text{at } x = x^*,$$

as depicted in Fig. 13.12. If this equation can be solved directly for x^*, you are done. However, if $f(x)$ is not a particularly simple function, so the derivative is not just a linear or quadratic function, you may not be able to solve the equation *analytically.* If not, the *one-dimensional search procedure* provides a straightforward way of solving the problem *numerically.*

The One-Dimensional Search Procedure

Like other search procedures in nonlinear programming, the *one-dimensional* search procedure finds a sequence of *trial solutions* that leads toward an optimal solution. At each iteration, you begin at the current trial solution to conduct a systematic search that culminates by identifying a new *improved* trial solution.

[1] See R. W. Cottle and G. B. Dantzig, "Complementary Pivot Theory of Mathematical Programming," *Linear Algebra and Its Applications,* **1:** 103–125, 1966; and R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem,* Academic Press, Boston, 1992.

[2] See the beginning of Appendix 3 for a review of the corresponding case when $f(x)$ is not concave.

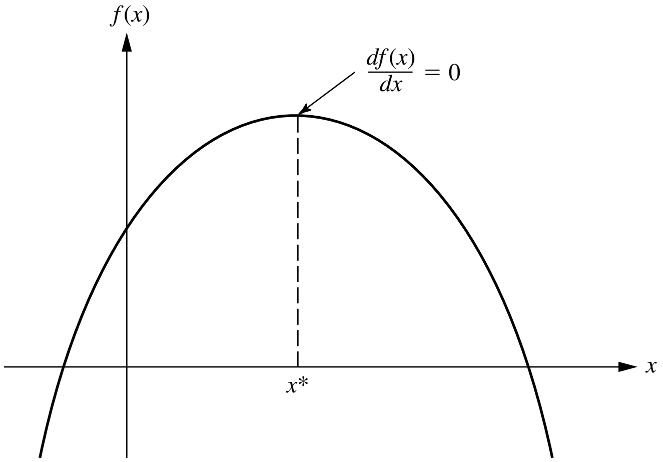


FIGURE 13.12
The one-variable unconstrained programming problem when the function is concave.

The idea behind the one-dimensional search procedure is a very intuitive one, namely, that whether the slope (derivative) is positive or negative at a trial solution definitely indicates whether improvement lies immediately to the right or left, respectively. Thus, if the derivative evaluated at a particular value of x is *positive,* then x^* must be larger than this x (see Fig. 13.12), so this x becomes a *lower bound* on the trial solutions that need to be considered thereafter. Conversely, if the derivative is *negative,* then x^* must be *smaller* than this x, so x would become an *upper bound.* Therefore, after both types of bounds have been identified, each new trial solution selected between the current bounds provides a new tighter bound of one type, thereby narrowing the search further. As long as a reasonable rule is used to select each trial solution in this way, the resulting *sequence* of trial solutions must *converge* to x^*. In practice, this means continuing the sequence until the distance between the bounds is sufficiently small that the next trial solution must be within a prespecified *error tolerance* of x^*.

This entire process is summarized next, given the notation

x' = current trial solution,

$\underline{x}$ = current lower bound on x^*,

$\overline{x}$ = current upper bound on x^*,

ϵ = error tolerance for x^*.

Although there are several reasonable rules for selecting each new trial solution, the one used in the following procedure is the **midpoint rule** (traditionally called the *Bolzano search plan*), which says simply to select the midpoint between the two current bounds.

Summary of the One-Dimensional Search Procedure.

Initialization: Select ϵ. Find an initial $\underline{x}$ and $\overline{x}$ by inspection (or by respectively finding any value of x at which the derivative is positive and then negative). Select an initial trial solution

$$x' = \frac{\underline{x} + \overline{x}}{2}.$$

Iteration:

1. Evaluate $\dfrac{df(x)}{dx}$ at $x = x'$.
2. If $\dfrac{df(x)}{dx} \geq 0$, reset $\underline{x} = x'$.
3. If $\dfrac{df(x)}{dx} \leq 0$, reset $\bar{x} = x'$.
4. Select a new $x' = \dfrac{\underline{x} + \bar{x}}{2}$.

Stopping rule: If $\bar{x} - \underline{x} \leq 2\epsilon$, so that the new x' must be within ϵ of x^*, stop. Otherwise, perform another iteration.

We shall now illustrate this procedure by applying it to the following example.

Example. Suppose that the function to be maximized is

$$f(x) = 12x - 3x^4 - 2x^6,$$

as plotted in Fig. 13.13. Its first two derivatives are

$$\frac{df(x)}{dx} = 12(1 - x^3 - x^5),$$

$$\frac{d^2f(x)}{dx^2} = -12(3x^2 + 5x^4).$$

FIGURE 13.13
Example for the one-dimensional search procedure.

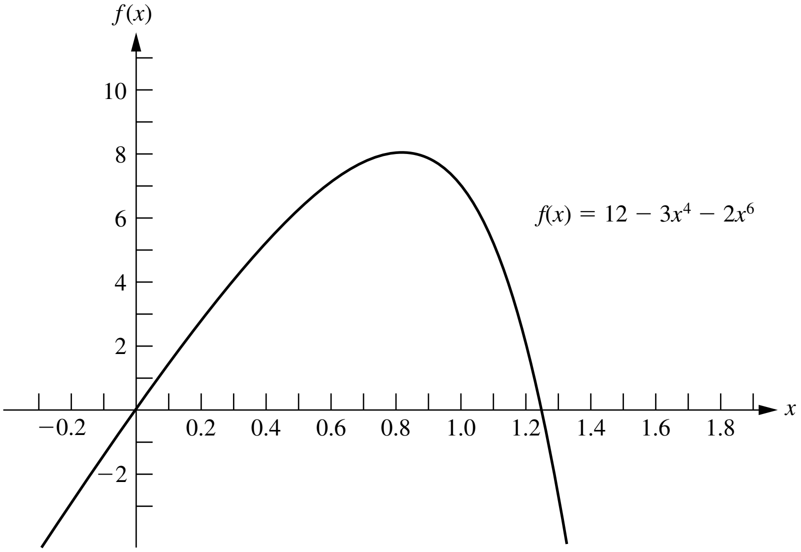


TABLE 13.1 Application of the one-dimensional search procedure to the example

Iteration	$\frac{df(x)}{dx}$	$\underline{x}$	$\overline{x}$	New x'	$f(x')$
0		0	2	1	7.0000
1	−12	0	1	0.5	5.7812
2	+10.12	0.5	1	0.75	7.6948
3	+4.09	0.75	1	0.875	7.8439
4	−2.19	0.75	0.875	0.8125	7.8672
5	+1.31	0.8125	0.875	0.84375	7.8829
6	−0.34	0.8125	0.84375	0.828125	7.8815
7	+0.51	0.828125	0.84375	0.8359375	7.8839
Stop					

Because the second derivative is nonpositive everywhere, $f(x)$ is a concave function, so the one-dimensional search procedure can be applied safely to find its global maximum (assuming a global maximum exists).

A quick inspection of this function (without even constructing its graph as shown in Fig. 13.13) indicates that $f(x)$ is positive for small positive values of x, but it is negative for $x < 0$ or $x > 2$. Therefore, $\underline{x} = 0$ and $\overline{x} = 2$ can be used as the initial bounds, with their midpoint, $x' = 1$, as the initial trial solution. Let $\epsilon = 0.01$ be the error tolerance for x^* in the stopping rule, so the final $(\overline{x} - \underline{x}) \leq 0.02$ with the final x' at the midpoint.

Applying the one-dimensional search procedure then yields the sequence of results shown in Table 13.1. [This table includes both the function and derivative values for your information, where the derivative is evaluated at the trial solution generated at the *preceding* iteration. However, note that the algorithm actually doesn't need to calculate $f(x')$ at all and that it only needs to calculate the derivative far enough to determine its sign.] The conclusion is that

$$x^* \approx 0.836,$$
$$0.828125 < x^* < 0.84375.$$

Your OR Courseware includes an interactive routine for executing the one-dimensional search procedure.

13.5 MULTIVARIABLE UNCONSTRAINED OPTIMIZATION

Now consider the problem of maximizing a *concave* function $f(\mathbf{x})$ of *multiple* variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ when there are no constraints on the feasible values. Suppose again that the necessary and sufficient condition for optimality, given by the system of equations obtained by setting the respective partial derivatives equal to zero (see Sec. 13.3), cannot be solved analytically, so that a numerical search procedure must be used. How can the preceding one-dimensional search procedure be extended to this multidimensional problem?

In Sec. 13.4, the value of the ordinary derivative was used to select one of just two possible directions (increase x or decrease x) in which to move from the current trial solution to the next one. The goal was to reach a point eventually where this derivative is (essentially) 0. Now, there are *innumerable* possible directions in which to move; they correspond to the possible *proportional rates* at which the respective variables can be changed. The goal is to reach a point eventually where all the partial derivatives are (essentially) 0. Therefore, extending the one-dimensional search procedure requires using the values of the *partial* derivatives to select the specific direction in which to move. This selection involves using the gradient of the objective function, as described next.

Because the objective function $f(\mathbf{x})$ is assumed to be differentiable, it possesses a gradient, denoted by $\nabla f(\mathbf{x})$, at each point $\mathbf{x}$. In particular, the **gradient** at a specific point $\mathbf{x} = \mathbf{x}'$ is the *vector* whose elements are the respective *partial derivatives* evaluated at $\mathbf{x} = \mathbf{x}'$, so that

$$\nabla f(\mathbf{x}') = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right) \qquad \text{at } \mathbf{x} = \mathbf{x}'.$$

The significance of the gradient is that the (infinitesimal) change in $\mathbf{x}$ that *maximizes* the rate at which $f(\mathbf{x})$ increases is the change that is *proportional* to $\nabla f(\mathbf{x})$. To express this idea geometrically, the "direction" of the gradient $\nabla f(\mathbf{x}')$ is interpreted as the *direction* of the directed line segment (arrow) from the origin $(0, 0, \dots, 0)$ to the point $(\partial f/\partial x_1, \partial f/\partial x_2, \dots, \partial f/\partial x_n)$, where $\partial f/\partial x_j$ is evaluated at $x_j = x_j'$. Therefore, it may be said that the rate at which $f(\mathbf{x})$ increases is maximized if (infinitesimal) changes in $\mathbf{x}$ are in the *direction* of the gradient $\nabla f(\mathbf{x})$. Because the objective is to find the feasible solution maximizing $f(\mathbf{x})$, it would seem expedient to attempt to move in the direction of the gradient as much as possible.

OR TUTOR

The Gradient Search Procedure

Because the current problem has no constraints, this interpretation of the gradient suggests that an efficient search procedure should keep moving in the direction of the gradient until it (essentially) reaches an optimal solution $\mathbf{x}^*$, where $\nabla f(\mathbf{x}^*) = \mathbf{0}$. However, normally it would not be practical to change $\mathbf{x}$ *continuously* in the direction of $\nabla f(\mathbf{x})$, because this series of changes would require continuously *reevaluating* the $\partial f/\partial x_j$ and changing the direction of the path. Therefore, a better approach is to keep moving in a *fixed* direction from the current trial solution, not stopping until $f(\mathbf{x})$ stops increasing. This stopping point would be the next trial solution, so the gradient then would be recalculated to determine the new direction in which to move. With this approach, each iteration involves changing the current trial solution $\mathbf{x}'$ as follows:

$$\text{Reset} \qquad \mathbf{x}' = \mathbf{x}' + t^* \, \nabla f(\mathbf{x}'),$$

where t^* is the positive value of t that *maximizes* $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$; that is,

$$f(\mathbf{x}' + t^*\,\nabla f(\mathbf{x}')) = \max_{t \geq 0} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')).$$

[Note that $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ is simply $f(\mathbf{x})$ where

$$x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}'}, \qquad \text{for } j = 1, 2, \dots, n,$$

and that these expressions for the x_j involve only constants and t, so $f(\mathbf{x})$ becomes a function of just the single variable t.] The iterations of this gradient search procedure continue until $\nabla f(\mathbf{x}) = 0$ within a small tolerance ϵ, that is, until

$$\left|\frac{\partial f}{\partial x_j}\right| \leq \epsilon \qquad \text{for } j = 1, 2, \ldots, n.$$[1]

An analogy may help to clarify this procedure. Suppose that you need to climb to the top of a hill. You are nearsighted, so you cannot see the top of the hill in order to walk directly in that direction. However, when you stand still, you can see the ground around your feet well enough to determine the direction in which the hill is sloping upward most sharply. You are able to walk in a straight line. While walking, you also are able to tell when you stop climbing (zero slope in your direction). Assuming that the hill is *concave,* you now can use the *gradient search procedure* for climbing to the top efficiently. This problem is a *two-variable problem*, where (x_1, x_2) represents the coordinates (ignoring height) of your current location. The function $f(x_1, x_2)$ gives the height of the hill at (x_1, x_2). You start each iteration at your current location (current trial solution) by determining the direction [in the (x_1, x_2) coordinate system] in which the hill is sloping upward most sharply (the direction of the gradient) at this point. You then begin walking in this fixed direction and continue as long as you still are climbing. You eventually stop at a new trial location (solution) when the hill becomes level in your direction, at which point you prepare to do another iteration in another direction. You continue these iterations, following a zigzag path up the hill, until you reach a trial location where the slope is essentially zero in all directions. Under the assumption that the hill $[f(x_1, x_2)]$ is concave, you must then be essentially at the top of the hill.

The most difficult part of the gradient search procedure usually is to find t^*, the value of t that maximizes f in the direction of the gradient, at each iteration. Because $\mathbf{x}$ and $\nabla f(\mathbf{x})$ have fixed values for the maximization, and because $f(\mathbf{x})$ is concave, this problem should be viewed as maximizing a *concave* function of a *single variable t.* Therefore, it can be solved by the one-dimensional search procedure of Sec. 13.4 (where the initial lower bound on t must be nonnegative because of the $t \geq 0$ constraint). Alternatively, if f is a simple function, it may be possible to obtain an analytical solution by setting the derivative with respect to t equal to zero and solving.

Summary of the Gradient Search Procedure.

Initialization: Select ϵ and any initial trial solution x'. Go first to the stopping rule.

Iteration:

1. Express $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ as a function of t by setting

$$x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}'}, \qquad \text{for } j = 1, 2, \ldots, n,$$

and then substituting these expressions into $f(\mathbf{x})$.

[1] This stopping rule generally will provide a solution $\mathbf{x}$ that is close to an optimal solution $\mathbf{x}^*$, with a value of $f(\mathbf{x})$ that is very close to $f(\mathbf{x}^*)$. However, this cannot be guaranteed, since it is possible that the function maintains a very small positive slope ($\leq \epsilon$) over a great distance from $\mathbf{x}$ to $\mathbf{x}^*$.

2. Use the one-dimensional search procedure (or calculus) to find $t = t^*$ that maximizes $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ over $t \geq 0$.

3. Reset $\mathbf{x}' = \mathbf{x}' + t^*\,\nabla f(\mathbf{x}')$. Then go to the stopping rule.

Stopping rule: Evaluate $\nabla f(\mathbf{x}')$ at $\mathbf{x} = \mathbf{x}'$. Check if

$$\left|\frac{\partial f}{\partial x_j}\right| \leq \epsilon \qquad \text{for all } j = 1, 2, \ldots, n.$$

If so, stop with the current $\mathbf{x}'$ as the desired approximation of an optimal solution $\mathbf{x}^*$. Otherwise, perform another iteration.

Now let us illustrate this procedure.

Example. Consider the following two-variable problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2.$$

Thus,

$$\frac{\partial f}{\partial x_1} = 2x_2 - 2x_1,$$

$$\frac{\partial f}{\partial x_2} = 2x_1 + 2 - 4x_2.$$

We also can verify (see Appendix 2) that $f(\mathbf{x})$ is concave. To begin the gradient search procedure, suppose that $\mathbf{x} = (0, 0)$ is selected as the initial trial solution. Because the respective partial derivatives are 0 and 2 at this point, the gradient is

$$\nabla f(0, 0) = (0, 2).$$

Therefore, to begin the first iteration, set

$$x_1 = 0 + t(0) = 0,$$
$$x_2 = 0 + t(2) = 2t,$$

and then substitute these expressions into $f(\mathbf{x})$ to obtain

$$\begin{aligned} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) &= f(0, 2t) \\ &= 2(0)(2t) + 2(2t) - 0^2 - 2(2t)^2 \\ &= 4t - 8t^2. \end{aligned}$$

Because

$$f(0, 2t^*) = \max_{t \geq 0} f(0, 2t) = \max_{t \geq 0} \{4t - 8t^2\}$$

and

$$\frac{d}{dt}(4t - 8t^2) = 4 - 16t = 0,$$

it follows that

$$t^* = \frac{1}{4},$$

so

$$\text{Reset} \qquad \mathbf{x}' = (0, 0) + \frac{1}{4}(0, 2) = \left(0, \frac{1}{2}\right).$$

For this new trial solution, the gradient is

$$\nabla f\left(0, \frac{1}{2}\right) = (1, 0).$$

Thus, for the second iteration, set

$$\mathbf{x} = \left(0, \frac{1}{2}\right) + t(1, 0) = \left(t, \frac{1}{2}\right),$$

so

$$\begin{aligned} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) &= f\left(0 + t, \frac{1}{2} + 0t\right) = f\left(t, \frac{1}{2}\right) \\ &= (2t)\left(\frac{1}{2}\right) + 2\left(\frac{1}{2}\right) - t^2 - 2\left(\frac{1}{2}\right)^2 \\ &= t - t^2 + \frac{1}{2}. \end{aligned}$$

Because

$$f\left(t^*, \frac{1}{2}\right) = \max_{t \geq 0} f\left(t, \frac{1}{2}\right) = \max_{t \geq 0} \left\{t - t^2 + \frac{1}{2}\right\}$$

and

$$\frac{d}{dt}\left(t - t^2 + \frac{1}{2}\right) = 1 - 2t = 0,$$

then

$$t^* = \frac{1}{2},$$

so

$$\text{Reset} \qquad \mathbf{x}' = \left(0, \frac{1}{2}\right) + \frac{1}{2}(1, 0) = \left(\frac{1}{2}, \frac{1}{2}\right).$$

A nice way of organizing this work is to write out a table such as Table 13.2 which summarizes the preceding two iterations. At each iteration, the second column shows the current trial solution, and the rightmost column shows the eventual new trial solution, which then is carried down into the second column for the next iteration. The fourth column gives the expressions for the x_j in terms of t that need to be substituted into $f(\mathbf{x})$ to give the fifth column.

By continuing in this fashion, the subsequent trial solutions would be $(\frac{1}{2}, \frac{3}{4})$, $(\frac{3}{4}, \frac{3}{4})$, $(\frac{3}{4}, \frac{7}{8})$, $(\frac{7}{8}, \frac{7}{8})$, . . . , as shown in Fig. 13.14. Because these points are converging to $\mathbf{x}^* = (1, 1)$, this solution is the optimal solution, as verified by the fact that

$$\nabla f(1, 1) = (0, 0).$$

TABLE 13.2 Application of the gradient search procedure to the example

Iteration	$\mathbf{x}'$	$\nabla f(\mathbf{x}')$	$\mathbf{x}' + t\,\nabla f(\mathbf{x}')$	$f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$	t^*	$\mathbf{x}' + t^*\,\nabla f(\mathbf{x}')$
1	(0, 0)	(0, 2)	(0, 2t)	$4t - 8t^2$	$\frac{1}{4}$	$\left(0, \frac{1}{2}\right)$
2	$\left(0, \frac{1}{2}\right)$	(1, 0)	$\left(t, \frac{1}{2}\right)$	$t - t^2 + \frac{1}{2}$	$\frac{1}{2}$	$\left(\frac{1}{2}, \frac{1}{2}\right)$

However, because this converging sequence of trial solutions never reaches its limit, the procedure actually will stop somewhere (depending on ϵ) slightly below (1, 1) as its final approximation of $\mathbf{x}^*$.

As Fig. 13.14 suggests, the gradient search procedure zigzags to the optimal solution rather than moving in a straight line. Some modifications of the procedure have been developed that *accelerate* movement toward the optimal solution by taking this zigzag behavior into account.

If $f(\mathbf{x})$ were *not* a concave function, the gradient search procedure still would converge to a *local* maximum. The only change in the description of the procedure for this case is that t^* now would correspond to the *first local maximum* of $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ as t is increased from 0.

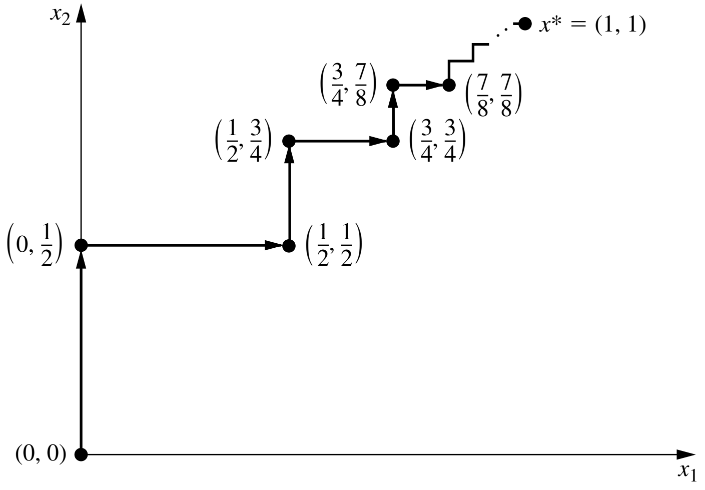
If the objective were to *minimize* $f(\mathbf{x})$ instead, one change in the procedure would be to move in the *opposite* direction of the gradient at each iteration. In other words, the rule for obtaining the next point would be

$$\text{Reset} \qquad \mathbf{x}' = \mathbf{x}' - t^*\,\nabla f(\mathbf{x}').$$

The only other change is that t^* now would be the nonnegative value of t that *minimizes* $f(\mathbf{x}' - t\,\nabla f(\mathbf{x}'))$; that is,

$$f(\mathbf{x}' - t^*\,\nabla f(\mathbf{x}')) = \min_{t \geq 0} f(\mathbf{x}' - t\,\nabla f(\mathbf{x}')).$$

FIGURE 13.14 Illustration of the gradient search procedure when $f(x_1, x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$.


x_2
$x^* = (1, 1)$
$\left(\frac{3}{4}, \frac{7}{8}\right)$
$\left(\frac{7}{8}, \frac{7}{8}\right)$
$\left(\frac{1}{2}, \frac{3}{4}\right)$
$\left(\frac{3}{4}, \frac{3}{4}\right)$
$\left(0, \frac{1}{2}\right)$
$\left(\frac{1}{2}, \frac{1}{2}\right)$
(0, 0)
x_1

Another example of an application of the gradient search procedure is included in your OR Tutor. The OR Courseware includes both an interactive routine and an automatic routine for applying this algorithm.

13.6 THE KARUSH-KUHN-TUCKER (KKT) CONDITIONS FOR CONSTRAINED OPTIMIZATION

We now focus on the question of how to recognize an *optimal solution* for a nonlinear programming problem (with differentiable functions). What are the necessary and (perhaps) sufficient conditions that such a solution must satisfy?

In the preceding sections we already noted these conditions for *unconstrained optimization,* as summarized in the first two rows of Table 13.3. Early in Sec. 13.3 we also gave these conditions for the slight *extension* of unconstrained optimization where the *only* constraints are nonnegativity constraints. These conditions are shown in the third row of Table 13.3. As indicated in the last row of the table, the conditions for the general case are called the **Karush-Kuhn-Tucker conditions** (or **KKT conditions**), because they were derived independently by Karush[1] and by Kuhn and Tucker.[2] Their basic result is embodied in the following theorem.

Theorem. Assume that $f(\mathbf{x})$, $g_1(\mathbf{x})$, $g_2(\mathbf{x})$, . . . , $g_m(\mathbf{x})$ are *differentiable* functions satisfying certain regularity conditions.[3] Then

$$\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$$

[1] W. Karush, "Minima of Functions of Several Variables with Inequalities as Side Conditions," M.S. thesis, Department of Mathematics, University of Chicago, 1939.

[2] H. W. Kuhn and A. W. Tucker, "Nonlinear Programming," in Jerzy Neyman (ed.), *Proceedings of the Second Berkeley Symposium,* University of California Press, Berkeley, 1951, pp. 481–492.

[3] Ibid., p. 483.

TABLE 13.3 Necessary and sufficient conditions for optimality

Problem	Necessary Conditions for Optimality	Also Sufficient if:
One-variable unconstrained	$\frac{df}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$	$f(\mathbf{x})$ concave
Constrained, nonnegativity constraints only	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$ (or ≤ 0 if $x_j = 0$)	$f(\mathbf{x})$ concave
General constrained problem	Karush-Kuhn-Tucker conditions	$f(\mathbf{x})$ concave and $g_i(\mathbf{x})$ convex $(i = 1, 2, \ldots, m)$

can be an *optimal solution* for the nonlinear programming problem only if there exist m numbers $u_1, u_2, \ldots, u_m$ such that *all* the following *KKT conditions* are satisfied:

$$\left.\begin{aligned} &\textbf{1. } \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \le 0 \\ &\textbf{2. } x_j^* \left(\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \right) = 0 \end{aligned}\right\} \quad \text{at } \mathbf{x} = \mathbf{x}^*, \text{ for } j = 1, 2, \ldots, n.$$

$$\left.\begin{aligned} &\textbf{3. } g_i(\mathbf{x}^*) - b_i \le 0 \\ &\textbf{4. } u_i[g_i(\mathbf{x}^*) - b_i] = 0 \end{aligned}\right\} \quad \text{for } i = 1, 2, \ldots, m.$$

$$\textbf{5. } x_j^* \ge 0, \qquad \text{for } j = 1, 2, \ldots, n.$$

$$\textbf{6. } u_i \ge 0, \qquad \text{for } i = 1, 2, \ldots, m.$$

Note that both conditions 2 and 4 require that the product of two quantities be zero. Therefore, each of these conditions really is saying that at least one of the two quantities must be zero. Consequently, condition 4 can be combined with condition 3 to express them in another equivalent form as

$$(3, 4) \qquad g_i(\mathbf{x}^*) - b_i = 0$$
$$(\text{or} \le 0 \quad \text{if } u_i = 0), \qquad \text{for } i = 1, 2, \ldots, m.$$

Similarly, condition 2 can be combined with condition 1 as

$$(1, 2) \qquad \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} = 0$$
$$(\text{or} \le 0 \quad \text{if } x_j^* = 0), \qquad \text{for } j = 1, 2, \ldots, n.$$

When $m = 0$ (no functional constraints), this summation drops out and the combined condition (1, 2) reduces to the condition given in the third row of Table 13.3. Thus, for $m > 0$, each term in the summation modifies the $m = 0$ condition to incorporate the effect of the corresponding functional constraint.

In conditions 1, 2, 4, and 6, the u_i correspond to the *dual variables* of linear programming (we expand on this correspondence at the end of the section), and they have a comparable economic interpretation. However, the u_i actually arose in the mathematical derivation as *Lagrange multipliers* (discussed in Appendix 3). Conditions 3 and 5 do nothing more than ensure the feasibility of the solution. The other conditions eliminate most of the feasible solutions as possible candidates for an optimal solution.

However, note that satisfying these conditions does not guarantee that the solution is optimal. As summarized in the rightmost column of Table 13.3, certain additional *convexity* assumptions are needed to obtain this guarantee. These assumptions are spelled out in the following extension of the theorem.

Corollary. Assume that $f(\mathbf{x})$ is a *concave* function and that $g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})$ are *convex* functions (i.e., this problem is a convex programming problem), where all these functions satisfy the regularity conditions. Then $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is an *optimal solution* if and only if all the conditions of the theorem are satisfied.

Example. To illustrate the formulation and application of the *KKT conditions,* we consider the following two-variable nonlinear programming problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = \ln(x_1 + 1) + x_2,$$

subject to

$$2x_1 + x_2 \leq 3$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0,$$

where ln denotes the natural logarithm. Thus, $m = 1$ (one functional constraint) and $g_1(\mathbf{x}) = 2x_1 + x_2$, so $g_1(\mathbf{x})$ is convex. Furthermore, it can be easily verified (see Appendix 2) that $f(\mathbf{x})$ is concave. Hence, the corollary applies, so any solution that satisfies the KKT conditions will definitely be an optimal solution. Applying the formulas given in the theorem yields the following KKT conditions for this example:

1(*j* = 1). $\dfrac{1}{x_1 + 1} - 2u_1 \leq 0.$

2(*j* = 1). $x_1\left(\dfrac{1}{x_1 + 1} - 2u_1\right) = 0.$

1(*j* = 2). $1 - u_1 \leq 0.$

2(*j* = 2). $x_2(1 - u_1) = 0.$

3. $2x_1 + x_2 - 3 \leq 0.$

4. $u_1(2x_1 + x_2 - 3) = 0.$

5. $x_1 \geq 0, x_2 \geq 0.$

6. $u_1 \geq 0.$

The steps in solving the KKT conditions for this particular example are outlined below.

1. $u_1 \geq 1$, from condition $1(j = 2)$.
   $x_1 \geq 0$, from condition 5.
2. Therefore, $\dfrac{1}{x_1 + 1} - 2u_1 < 0.$
3. Therefore, $x_1 = 0$, from condition $2(j = 1)$.
4. $u_1 \neq 0$ implies that $2x_1 + x_2 - 3 = 0$, from condition 4.
5. Steps 3 and 4 imply that $x_2 = 3$.
6. $x_2 \neq 0$ implies that $u_1 = 1$, from condition $2(j = 2)$.
7. No conditions are violated by $x_1 = 0$, $x_2 = 3$, $u_1 = 1$.

Therefore, there exists a number $u_1 = 1$ such that $x_1 = 0$, $x_2 = 3$, and $u_1 = 1$ satisfy all the conditions. Consequently, $\mathbf{x}^* = (0, 3)$ is an optimal solution for this problem.

The particular progression of steps needed to solve the KKT conditions will differ from one problem to the next. When the logic is not apparent, it is sometimes helpful to consider separately the different cases where each x_j and u_i are specified to be either equal to or greater than 0 and then trying each case until one leads to a solution. In the example, there are eight such cases corresponding to the eight combinations of $x_1 = 0$ versus $x_1 > 0$, $x_2 = 0$ versus $x_2 > 0$, and $u_1 = 0$ versus $u_1 > 0$. Each case leads to a simpler statement and analysis of the conditions. To illustrate, consider first the case shown next, where $x_1 = 0$, $x_2 = 0$, and $u_1 = 0$.

KKT Conditions for the Case $x_1 = 0, x_2 = 0, u_1 = 0$

1(j = 1). $\dfrac{1}{0+1} \leq 0.$ Contradiction.
1(j = 2). $1 - 0 \leq 0.$ Contradiction.
3. $0 + 0 \leq 3.$
(All the other conditions are redundant.)

As listed below, the other three cases where $u_1 = 0$ also give immediate contradictions in a similar way, so no solution is available.

Case $x_1 = 0$, $x_2 > 0$, $u_1 = 0$ contradicts conditions 1($j = 1$), 1($j = 2$), and 2($j = 2$).
Case $x_1 > 0$, $x_2 = 0$, $u_1 = 0$ contradicts conditions 1($j = 1$), 2($j = 1$), and 1($j = 2$).
Case $x_1 > 0$, $x_2 > 0$, $u_1 = 0$ contradicts conditions 1($j = 1$), 2($j = 1$), 1($j = 2$), and 2($j = 2$).

The case $x_1 > 0$, $x_2 > 0$, $u_1 > 0$ enables one to delete these nonzero multipliers from conditions 2($j = 1$), 2($j = 2$), and 4, which then enables deletion of conditions 1($j = 1$), 1($j = 2$), and 3 as redundant, as summarized next.

KKT Conditions for the Case $x_1 > 0, x_2 > 0, u_1 > 0$

1(j = 1). $\dfrac{1}{x_1+1} - 2u_1 = 0.$
2(j = 2). $1 - u_1 = 0.$
4. $2x_1 + x_2 - 3 = 0.$
(All the other conditions are redundant.)

Therefore, $u_1 = 1$, so $x_1 = -\frac{1}{2}$, which contradicts $x_1 > 0$.

Now suppose that the case $x_1 = 0$, $x_2 > 0$, $u_1 > 0$ is tried next.

KKT Conditions for the Case $x_1 = 0, x_2 > 0, u_1 > 0$

1(j = 1). $\dfrac{1}{0+1} - 2u_1 = 0.$
2(j = 2). $1 - u_1 = 0.$
4. $0 + x_2 - 3 = 0.$
(All the other conditions are redundant.)

Therefore, $x_1 = 0$, $x_2 = 3$, $u_1 = 1$. Having found a solution, we know that no additional cases need be considered.

For problems more complicated than this example, it may be difficult, if not essentially impossible, to derive an optimal solution *directly* from the KKT conditions. Nevertheless, these conditions still provide valuable clues as to the identity of an optimal solution, and they also permit us to check whether a proposed solution may be optimal.

There also are many valuable *indirect* applications of the KKT conditions. One of these applications arises in the *duality theory* that has been developed for nonlinear programming to parallel the duality theory for linear programming presented in Chap. 6. In particular, for any given constrained maximization problem (call it the *primal problem*), the KKT conditions can be used to define a closely associated dual problem that is a constrained minimization problem. The variables in the dual problem[1] consist of both the Lagrange multipliers u_i $(i = 1, 2, \dots, m)$ and the primal variables x_j $(j = 1, 2, \dots, n)$. In the special case where the primal problem is a linear programming problem, the x_j variables drop out of the dual problem and it becomes the familiar dual problem of linear programming (where the u_i variables here correspond to the y_i variables in Chap. 6). When the primal problem is a convex programming problem, it is possible to establish relationships between the primal problem and the dual problem that are similar to those for linear programming. For example, the *strong duality property* of Sec. 6.1, which states that the optimal objective function values of the two problems are equal, also holds here. Furthermore, the values of the u_i variables in an optimal solution for the dual problem can again be interpreted as *shadow prices* (see Secs. 4.7 and 6.2); i.e., they give the rate at which the optimal objective function value for the primal problem could be increased by (slightly) increasing the right-hand side of the corresponding constraint. Because duality theory for nonlinear programming is a relatively advanced topic, the interested reader is referred elsewhere for further information.[2]

You will see another indirect application of the KKT conditions in the next section.

13.7 QUADRATIC PROGRAMMING

As indicated in Sec. 13.3, the quadratic programming problem differs from the linear programming problem only in that the objective function also includes x_j^2 and $x_i x_j$ $(i \neq j)$ terms. Thus, if we use matrix notation like that introduced at the beginning of Sec. 5.2, the problem is to find $\mathbf{x}$ so as to

$$\text{Maximize} \qquad f(\mathbf{x}) = \mathbf{c}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x},$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \qquad \text{and} \qquad \mathbf{x} \geq \mathbf{0},$$

where $\mathbf{c}$ is a row vector, $\mathbf{x}$ and $\mathbf{b}$ are column vectors, $\mathbf{Q}$ and $\mathbf{A}$ are matrices, and the superscript T denotes the transpose (see Appendix 4). The q_{ij} (elements of Q) are given constants such that $q_{ij} = q_{ji}$ (which is the reason for the factor of $\frac{1}{2}$ in the objective function). By performing the indicated vector and matrix multiplications, the objective function then is expressed in terms of these q_{ij}, the c_j (elements of $\mathbf{c}$), and the variables as follows:

$$f(\mathbf{x}) = \mathbf{c}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} = \sum_{j=1}^{n} c_j x_j - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j.$$

For each term where $i = j$ in this double summation, $x_i x_j = x_j^2$, so $-\frac{1}{2}q_{jj}$ is the coefficient of x_j^2. When $i \neq j$, then $-\frac{1}{2}(q_{ij}x_i x_j + q_{ji}x_j x_i) = -q_{ij}x_i x_j$, so $-q_{ij}$ is the total coefficient for the product of x_i and x_j.

[1] For details on this formulation, see O. T. Mangasarian, *Nonlinear Programming,* McGraw-Hill, New York, 1969, chap 8. For a unified survey of various approaches to duality in nonlinear programming, see A. M. Geoffrion, "Duality in Nonlinear Programming: A Simplified Applications-Oriented Development," *SIAM Review,* **13:** 1–37, 1971.

[2] Ibid.

To illustrate this notation, consider the following example of a quadratic programming problem.

$$\text{Maximize} \qquad f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2,$$

subject to

$$x_1 + 2x_2 \le 30$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

In this case,

$$\mathbf{c} = [15 \quad 30], \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix},$$

$$\mathbf{A} = [1 \quad 2], \qquad \mathbf{b} = [30].$$

Note that

$$\begin{aligned} \mathbf{x}^T\mathbf{Q}\mathbf{x} &= [x_1 \quad x_2] \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= [(4x_1 - 4x_2) \quad (-4x_1 + 8x_2)] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 4x_1^2 - 4x_2x_1 - 4x_1x_2 + 8x_2^2 \\ &= q_{11}x_1^2 + q_{21}x_2x_1 + q_{12}x_1x_2 + q_{22}x_2^2. \end{aligned}$$

Multiplying through by $-\frac{1}{2}$ gives

$$-\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} = -2x_1^2 + 4x_1x_2 - 4x_2^2,$$

which is the nonlinear portion of the objective function for this example. Since $q_{11} = 4$ and $q_{22} = 8$, the example illustrates that $-\frac{1}{2}q_{jj}$ is the coefficient of x_j^2 in the objective function. The fact that $q_{12} = q_{21} = -4$ illustrates that both $-q_{ij}$ and $-q_{ji}$ give the total coefficient of the product of x_i and x_j.

Several algorithms have been developed for the special case of the quadratic programming problem where the objective function is a *concave* function. (A way to verify that the objective function is concave is to verify the equivalent condition that

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} \ge \mathbf{0}$$

for all $\mathbf{x}$, that is, $\mathbf{Q}$ is a *positive semidefinite* matrix.) We shall describe one[1] of these algorithms, the *modified simplex method,* that has been quite popular because it requires using only the simplex method with a slight modification. The key to this approach is to construct the KKT conditions from the preceding section and then to reexpress these conditions in a convenient form that closely resembles linear programming. Therefore, before describing the algorithm, we shall develop this convenient form.

[1] P. Wolfe, "The Simplex Method for Quadratic Programming," *Econometrics,* **27:** 382–398, 1959. This paper develops both a short form and a long form of the algorithm. We present a version of the *short form,* which assumes further that *either* $\mathbf{c} = \mathbf{0}$ *or* the objective function is *strictly* concave.

The KKT Conditions for Quadratic Programming

For concreteness, let us first consider the above example. Starting with the form given in the preceding section, its KKT conditions are the following.

1(*j* = 1). $15 + 4\mathrm{x}_2 - 4x_1 - u_1 \le 0.$
2(*j* = 1). $x_1(15 + 4x_2 - 4x_1 - u_1) = 0.$
1(*j* = 2). $30 + 4x_1 - 8x_2 - 2u_1 \le 0.$
2(*j* = 2). $x_2(30 + 4x_1 - 8x_2 - 2u_1) = 0.$
3. $x_1 + 2x_2 - 30 \le 0.$
4. $u_1(\mathrm{x}_1 + 2x_2 - 30) = 0.$
5. $x_1 \ge 0, \quad x_2 \ge 0.$
6. $u_1 \ge 0.$

To begin reexpressing these conditions in a more convenient form, we move the constants in conditions 1(j = 1), 1(j = 2), and 3 to the right-hand side and then introduce nonnegative *slack variables* (denoted by y_1, y_2, and v_1, respectively) to convert these inequalities to equations.

$$\begin{array}{lrrrrrrl}
\textbf{1}(\boldsymbol{j}=\textbf{1}). & -4x_1 & +\,4x_2 & -\;\;u_1 & +\,y_1 & & & = -15 \\
\textbf{1}(\boldsymbol{j}=\textbf{2}). & 4x_1 & -\,8x_2 & -\,2u_1 & & +\,y_2 & & = -30 \\
\textbf{3}. & x_1 & +\,2x_2 & & & & +\,v_1 & = \;\;\;30
\end{array}$$

Note that condition 2(j = 1) can now be reexpressed as simply requiring that either $x_1 = 0$ or $y_1 = 0$; that is,

2(*j* = 1). $x_1y_1 = 0.$

In just the same way, conditions 2(j = 2) and 4 can be replaced by

2(*j* = 2). $x_2y_2 = 0,$
4. $u_1v_1 = 0.$

For each of these three pairs—(x_1, y_1), (x_2, y_2), (u_1, v_1)—the two variables are called **complementary variables,** because only one of the two variables can be nonzero. These new forms of conditions 2(j = 1), 2(j = 2), and 4 can be combined into one constraint,

$$x_1y_1 + x_2y_2 + u_1v_1 = 0,$$

called the **complementarity constraint.**

After multiplying through the equations for conditions 1(j = 1) and 1(j = 2) by −1 to obtain nonnegative right-hand sides, we now have the desired convenient form for the entire set of conditions shown here:

$$\begin{array}{l}
\begin{array}{rrrrrrl}
4x_1 & -\,4x_2 & +\;\;u_1 & -\,y_1 & & & = 15 \\
-4x_1 & +\,8x_2 & +\,2u_1 & & -\,y_2 & & = 30 \\
x_1 & +\,2x_2 & & & & +\,v_1 & = 30
\end{array} \\
x_1 \ge 0, \quad x_2 \ge 0, \quad u_1 \ge 0, \quad y_1 \ge 0, \quad y_2 \ge 0, \quad v_1 \ge 0 \\
x_1y_1 + x_2y_2 + u_1v_1 = 0
\end{array}$$

This form is particularly convenient because, except for the complementarity constraint, these conditions are *linear programming constraints.*

For *any* quadratic programming problem, its KKT conditions can be reduced to this same convenient form containing just linear programming constraints plus one complementarity constraint. In matrix notation again, this general form is

$$\begin{aligned}&\mathbf{Qx} + \mathbf{A}^T\mathbf{u} - \mathbf{y} = \mathbf{c}^T,\\ &\qquad \mathbf{Ax} + \mathbf{v} = \mathbf{b},\\ &\mathbf{x} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0},\\ &\mathbf{x}^T\mathbf{y} + \mathbf{u}^T\mathbf{v} = \mathbf{0},\end{aligned}$$

where the elements of the column vector **u** are the u_i of the preceding section and the elements of the column vectors **y** and **v** are slack variables.

Because the objective function of the original problem is assumed to be concave and because the constraint functions are linear and therefore convex, the corollary to the theorem of Sec. 13.6 applies. Thus, **x** is *optimal* if and only if there exist values of **y**, **u**, and **v** such that all four vectors together satisfy all these conditions. The original problem is thereby reduced to the equivalent problem of finding a *feasible solution* to these *constraints.*

It is of interest to note that this equivalent problem is one example of the *linear complementarity problem* introduced in Sec. 13.3 (see Prob. 13.3-6), and that a key constraint for the linear complementarity problem is its *complementarity constraint.*

The Modified Simplex Method

The *modified simplex method* exploits the key fact that, with the exception of the complementarity constraint, the KKT conditions in the convenient form obtained above are nothing more than linear programming constraints. Furthermore, the complementarity constraint simply implies that it is not permissible for *both* complementary variables of any pair to be (nondegenerate) basic variables (the only variables > 0) when (nondegenerate) BF solutions are considered. Therefore, the problem reduces to finding an initial BF solution to any linear programming problem that has these constraints, subject to this additional restriction on the identity of the basic variables. (This initial BF solution may be the only feasible solution in this case.)

As we discussed in Sec. 4.6, finding such an initial BF solution is relatively straightforward. In the simple case where $\mathbf{c}^T \leq \mathbf{0}$ (unlikely) and $\mathbf{b} \geq \mathbf{0}$, the initial basic variables are the elements of **y** and **v** (multiply through the first set of equations by -1), so that the desired solution is $\mathbf{x} = \mathbf{0}$, $\mathbf{u} = \mathbf{0}$, $\mathbf{y} = -\mathbf{c}^T$, $\mathbf{v} = \mathbf{b}$. Otherwise, you need to revise the problem by introducing an *artificial variable* into each of the equations where $c_j > 0$ (add the variable on the left) or $b_i < 0$ (subtract the variable on the left and then multiply through by -1) in order to use these artificial variables (call them z_1, z_2, and so on) as initial basic variables for the revised problem. (Note that this choice of initial basic variables satisfies the complementarity constraint, because as nonbasic variables $\mathbf{x} = \mathbf{0}$ and $\mathbf{u} = \mathbf{0}$ automatically.)

Next, use phase 1 of the *two-phase method* (see Sec. 4.6) to find a BF solution for the real problem; i.e., apply the simplex method (with one modification) to the following linear programming problem

$$\text{Minimize} \qquad Z = \sum_j z_j,$$

subject to the linear programming constraints obtained from the KKT conditions, but with these artificial variables included.

The one modification in the simplex method is the following change in the procedure for selecting an entering basic variable.

> **Restricted-Entry Rule:** When you are choosing an entering basic variable, exclude from consideration any nonbasic variable whose *complementary variable* already is a basic variable; the choice should be made from the other nonbasic variables according to the usual criterion for the simplex method.

This rule keeps the complementarity constraint satisfied throughout the course of the algorithm. When an optimal solution

$$\mathbf{x}^*, \mathbf{u}^*, \mathbf{y}^*, \mathbf{v}^*, z_1 = 0, \ldots, z_n = 0$$

is obtained for the phase 1 problem, $\mathbf{x}^*$ is the desired optimal solution for the original quadratic programming problem. Phase 2 of the two-phase method is not needed.

Example. We shall now illustrate this approach on the example given at the beginning of the section. As can be verified from the results in Appendix 2 (see Prob. 13.7-1*a*), $f(x_1, x_2)$ is *strictly concave;* i.e.,

$$\mathbf{Q} = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}$$

is positive definite, so the algorithm can be applied.

The starting point for solving this example is its KKT conditions in the convenient form obtained earlier in the section. After the needed artificial variables are introduced, the linear programming problem to be addressed explicitly by the modified simplex method then is

$$\text{Minimize} \qquad Z = z_1 + z_2,$$

subject to

$$\begin{aligned} 4x_1 - 4x_2 + u_1 - y_1 \qquad\qquad + z_1 \qquad &= 15 \\ -4x_1 + 8x_2 + 2u_1 \qquad - y_2 \qquad\qquad + z_2 &= 30 \\ x_1 + 2x_2 \qquad\qquad\qquad + v_1 \qquad\qquad &= 30 \end{aligned}$$

and

$$\begin{aligned} &x_1 \ge 0, \quad x_2 \ge 0, \quad u_1 \ge 0, \quad y_1 \ge 0, \quad y_2 \ge 0, \quad v_1 \ge 0, \\ &z_1 \ge 0, \quad z_2 \ge 0. \end{aligned}$$

The additional complementarity constraint

$$x_1y_1 + x_2y_2 + u_1v_1 = 0,$$

is not included explicitly, because the algorithm automatically enforces this constraint because of the *restricted-entry rule.* In particular, for each of the three pairs of complementary variables—(x_1, y_1), (x_2, y_2), (u_1,v_1)—whenever one of the two variables already is a basic variable, the other variable is *excluded* as a candidate for the entering basic variable. Remember that the only *nonzero* variables are basic variables. Because the initial set of basic variables for the linear programming problem—z_1, z_2, v_1—gives an initial BF solution that satisfies the complementarity constraint, there is no way that this constraint can be violated by any subsequent BF solution.

Table 13.4 shows the results of applying the modified simplex method to this problem. The first simplex tableau exhibits the initial system of equations *after* converting from minimizing Z to maximizing $-Z$ *and* algebraically eliminating the initial basic variables from Eq. (0), just as was done for the radiation therapy example in Sec. 4.6. The three iterations proceed just as for the regular simplex method, *except* for eliminating certain candidates for the entering basic variable because of the restricted-entry rule. In the first tableau, u_1 is eliminated as a candidate because its complementary variable (v_1) already is a basic variable (but x_2 would have been chosen anyway because $-4 < -3$). In the second tableau, both u_1 and y_2 are eliminated as candidates (because v_1 and x_2 are basic variables), so x_1 automatically is chosen as the only candidate with a negative coeffi-

TABLE 13.4 Application of the modified simplex method to the quadratic programming example

Iteration	Basic Variable	Eq.	Z	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	Right Side
0	Z	(0)	-1	0	-4	-3	1	1	0	0	0	-45
	z_1	(1)	0	4	-4	1	-1	0	0	1	0	15
	z_2	(2)	0	-4	8	2	0	-1	0	0	1	30
	v_1	(3)	0	1	2	0	0	0	1	0	0	30
1	Z	(0)	-1	-2	0	-2	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	-30
	z_1	(1)	0	2	0	2	-1	$-\frac{1}{2}$	0	1	$\frac{1}{2}$	30
	x_2	(2)	0	$-\frac{1}{2}$	1	$\frac{1}{4}$	0	$-\frac{1}{8}$	0	0	$\frac{1}{8}$	$3\frac{3}{4}$
	v_1	(3)	0	2	0	$-\frac{1}{2}$	0	$\frac{1}{4}$	1	0	$-\frac{1}{4}$	$22\frac{1}{2}$
2	Z	(0)	-1	0	0	$-\frac{5}{2}$	1	$\frac{3}{4}$	1	0	$\frac{1}{4}$	$-7\frac{1}{2}$
	z_1	(1)	0	0	0	$\frac{5}{2}$	-1	$-\frac{3}{4}$	-1	1	$\frac{3}{4}$	$7\frac{1}{2}$
	x_2	(2)	0	0	1	$\frac{1}{8}$	0	$-\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$9\frac{3}{8}$
	x_1	(3)	0	1	0	$-\frac{1}{4}$	0	$\frac{1}{8}$	$\frac{1}{2}$	0	$-\frac{1}{8}$	$11\frac{1}{4}$
3	Z	(0)	-1	0	0	0	0	0	0	1	1	0
	u_1	(1)	0	0	0	1	$-\frac{2}{5}$	$-\frac{3}{10}$	$-\frac{2}{5}$	$\frac{2}{5}$	$\frac{3}{10}$	3
	x_2	(2)	0	0	1	0	$\frac{1}{20}$	$-\frac{1}{40}$	$\frac{3}{10}$	$-\frac{1}{20}$	$\frac{1}{40}$	9
	x_1	(3)	0	1	0	0	$-\frac{1}{10}$	$\frac{1}{20}$	$\frac{2}{5}$	$\frac{1}{10}$	$-\frac{1}{20}$	12

cient in row 0 (whereas the *regular* simplex method would have permitted choosing *either* x_1 or u_1 because they are tied for having the largest negative coefficient). In the third tableau, both y_1 and y_2 are eliminated (because x_1 and x_2 are basic variables). However, u_1 is *not* eliminated because v_1 no longer is a basic variable, so u_1 is chosen as the entering basic variable in the usual way.

The resulting optimal solution for this phase 1 problem is $x_1 = 12$, $x_2 = 9$, $u_1 = 3$, with the rest of the variables zero. (Problem 13.7-1*c* asks you to verify that this solution is optimal by showing that $x_1 = 12$, $x_2 = 9$, $u_1 = 3$ satisfy the KKT conditions for the original problem when they are written in the form given in Sec. 13.6.) Therefore, the optimal solution for the quadratic programming problem (which includes only the x_1 and x_2 variables) is $(x_1, x_2) = (12, 9)$.

Some Software Options

Your OR Tutor includes an interactive routine for the modified simplex method to help you learn this algorithm efficiently. In addition, Excel, LINGO, LINDO, and MPL/CPLEX all can solve quadratic programming problems.

The procedure for using Excel is almost the same as with linear programming. The one crucial difference is that the equation entered for the cell that contains the value of the objective function now needs to be a quadratic equation. To illustrate, consider again the example introduced at the beginning of the section, which has the objective function

$$f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2.$$

Suppose that the values of x_1 and x_2 are in cells B4 and C4 of the Excel spreadsheet, and that the value of the objective function is in cell F4. Then the equation for cell F4 needs to be

F4 = 15*B4 + 30*C4 + 4*B4*C4 − 2*(B4^2) − 4*(C4^2),

where the symbol ^2 indicates an exponent of 2. Before solving the model, you should click on the Option button and make sure that the *Assume Linear Model* option is *not* selected (since this is not a *linear* programming model).

When using MPL/CPLEX, you should select the Quadratic Models option from the MPL Language option dialogue box and the Barrier method from the CPLEX Simplex options dialogue box. Otherwise, the procedure is the same as with linear programming except that the expression for the objective function now is a quadratic function. Thus, for the example, the objective function would be expressed as

15x1 + 30x2 + 4x1*x2 − 2(x1^2) − 4(x2^2).

Nothing more needs to be done when calling CPLEX, since it will automatically recognize the model as being a quadratic programming problem.

This objective function would be expressed in this same way for a LINGO model. LINGO then will automatically call its nonlinear solver to solve the model. When using LINDO instead, the procedure is somewhat more involved, since it requires converting the model to an equivalent linear form in terms of the KKT conditions. The LINDO file for this chapter illustrates how this is done for the example.

In fact, the Excel, MPL/CPLEX, and LINGO/LINDO files for this chapter in your OR Courseware all demonstrate their procedures by showing the details for how these software packages set up and solve this example.

Some of these software packages also can be applied to more complicated kinds of nonlinear programming problems than quadratic programming. Although CPLEX cannot, the professional version of MPL does support some other solvers that can. The student version of MPL on the CD-ROM includes one such solver called CONOPT (a product of ARKI Consulting) that should be used instead of CPLEX after selecting Nonlinear Models for the Default Model Type entry in the MPL Language option dialogue box. Both Excel and LINGO include versatile nonlinear solvers. However, be aware that these solvers are not guaranteed to find an optimal solution for complicated problems, especially nonconvex programming problems.

13.8 SEPARABLE PROGRAMMING

The preceding section showed how one class of nonlinear programming problems can be solved by an extension of the simplex method. We now consider another class, called *separable programming,* that actually can be solved by the simplex method itself, because any such problem can be approximated as closely as desired by a linear programming problem with a larger number of variables.

As indicated in Sec. 13.3, in separable programming it is assumed that the objective function $f(\mathbf{x})$ is concave, that each of the constraint functions $g_i(\mathbf{x})$ is convex, and that all these functions are separable functions (functions where each term involves just a single variable). However, to simplify the discussion, we focus here on the special case where the convex and separable $g_i(\mathbf{x})$ are, in fact, *linear functions,* just as for linear programming. Thus, only the objective function requires special treatment.

Under the preceding assumptions, the objective function can be expressed as a sum of concave functions of individual variables

$$f(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j),$$

so that each $f_j(x_j)$ has a shape[1] such as the one shown in Fig. 13.15 (either case) over the feasible range of values of x_j. Because $f(\mathbf{x})$ represents the measure of performance (say, profit) for all the activities together, $f_j(x_j)$ represents the *contribution to profit* from activity j when it is conducted at level x_j. The condition of $f(\mathbf{x})$ being separable simply implies additivity (see Sec. 3.3); i.e., there are no interactions between the activities (no cross-product terms) that affect total profit beyond their independent contributions. The assumption that each $f_j(x_j)$ is concave says that the *marginal profitability* (slope of the profit curve) either stays the same or decreases (*never* increases) as x_j is increased.

Concave profit curves occur quite frequently. For example, it may be possible to sell a limited amount of some product at a certain price, then a further amount at a lower price, and perhaps finally a further amount at a still lower price. Similarly, it may be necessary to purchase raw materials from increasingly expensive sources. In another common situ-

[1] $f(\mathbf{x})$ is concave if and only if *every* $f_j(x_j)$ is concave.

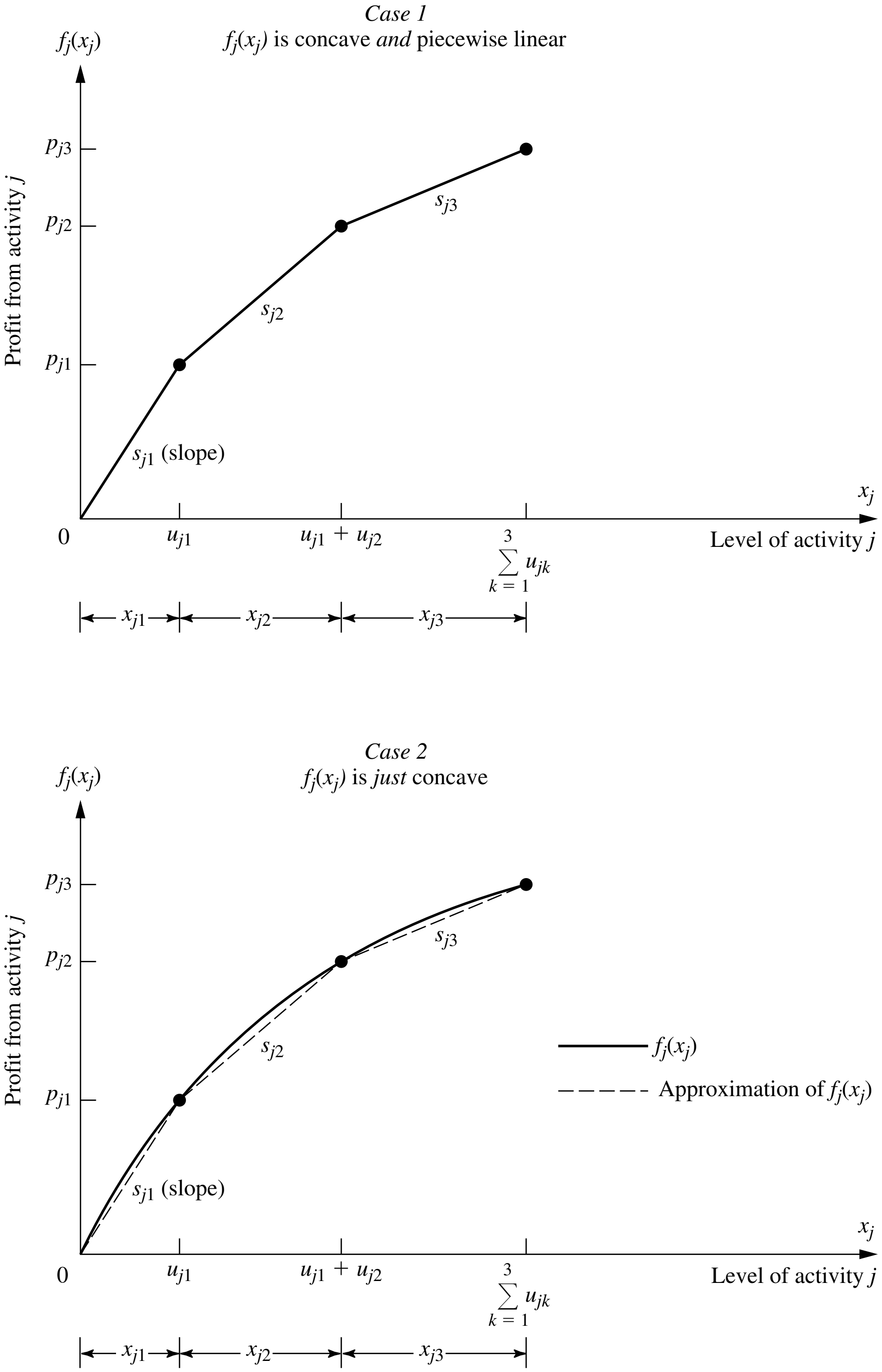


FIGURE 13.15
Shape of profit curves for separable programming.

ation, a more expensive production process must be used (e.g., overtime rather than regular-time work) to increase the production rate beyond a certain point.

These kinds of situations can lead to either type of profit curve shown in Fig. 13.15. In case 1, the slope decreases only at certain *breakpoints,* so that $f_j(x_j)$ is a *piecewise linear function* (a sequence of connected line segments). For case 2, the slope may decrease continuously as x_j increases, so that $f_j(x_j)$ is a general concave function. Any such function can be approximated as closely as desired by a piecewise linear function, and this kind of approximation is used as needed for separable programming problems. (Figure 13.15 shows an approximating function that consists of just three line segments, but the approximation can be made even better just by introducing additional breakpoints.) This approximation is very convenient because a piecewise linear function of a single variable can be rewritten as a *linear function* of several variables, with one special restriction on the values of these variables, as described next.

Reformulation as a Linear Programming Problem

The key to rewriting a piecewise linear function as a linear function is to use a separate variable for each line segment. To illustrate, consider the piecewise linear function $f_j(x_j)$ shown in Fig. 13.15, case 1 (or the approximating piecewise linear function for case 2), which has three line segments over the feasible range of values of x_j. Introduce the three new variables x_{j1}, x_{j2}, and x_{j3} and set

$$x_j = x_{j1} + x_{j2} + x_{j3},$$

where

$$0 \le x_{j1} \le u_{j1}, \qquad 0 \le x_{j2} \le u_{j2}, \qquad 0 \le x_{j3} \le u_{j3}.$$

Then use the slopes s_{j1}, s_{j2}, and s_{j3} to rewrite $f_j(x_j)$ as

$$f_j(x_j) = s_{j1}x_{j1} + s_{j2}x_{j2} + s_{j3}x_{j3},$$

with the *special restriction* that

$$\begin{aligned} x_{j2} &= 0 \qquad \text{whenever} \qquad x_{j1} < u_{j1}, \\ x_{j3} &= 0 \qquad \text{whenever} \qquad x_{j2} < u_{j2}. \end{aligned}$$

To see why this special restriction is required, suppose that $x_j = 1$, where $u_{jk} > 1$ ($k = 1, 2, 3$), so that $f_j(1) = s_{j1}$. Note that

$$x_{j1} + x_{j2} + x_{j3} = 1$$

permits

$$\begin{aligned} x_{j1} = 1, \quad x_{j2} = 0, \quad x_{j3} = 0 \quad &\Rightarrow \quad f_j(1) = s_{j1}, \\ x_{j1} = 0, \quad x_{j2} = 1, \quad x_{j3} = 0 \quad &\Rightarrow \quad f_j(1) = s_{j2}, \\ x_{j1} = 0, \quad x_{j2} = 0, \quad x_{j3} = 1 \quad &\Rightarrow \quad f_j(1) = s_{j3}, \end{aligned}$$

and so on, where

$$s_{j1} > s_{j2} > s_{j3}.$$

However, the special restriction permits only the first possibility, which is the only one giving the correct value for $f_j(1)$.

Unfortunately, the special restriction does not fit into the required format for linear programming constraints, so *some* piecewise linear functions cannot be rewritten in a linear programming format. However, *our* $f_j(x_j)$ are assumed to be concave, so $s_{j1} > s_{j2} > \cdots$, so that an algorithm for maximizing $f(\mathbf{x})$ *automatically* gives the highest priority to using x_{j1} when (in effect) increasing x_j from zero, the next highest priority to using x_{j2}, and so on, without even including the special restriction explicitly in the model. This observation leads to the following key property.

Key Property of Separable Programming. When $f(\mathbf{x})$ and the $g_i(\mathbf{x})$ satisfy the assumptions of separable programming, and when the resulting piecewise linear functions are rewritten as linear functions, deleting the *special restriction* gives a *linear programming model* whose optimal solution automatically satisfies the special restriction.

We shall elaborate further on the logic behind this key property later in this section in the context of a specific example. (Also see Prob. 13.8-8*a*).

To write down the complete linear programming model in the above notation, let n_j be the number of line segments in $f_j(x_j)$ (or the piecewise linear function approximating it), so that

$$x_j = \sum_{k=1}^{n_j} x_{jk}$$

would be substituted throughout the original model and

$$f_j(x_j) = \sum_{k=1}^{n_j} s_{jk} x_{jk}$$

would be substituted[1] into the objective function for $j = 1, 2, \ldots, n$. The resulting model is

$$\text{Maximize} \qquad Z = \sum_{j=1}^{n} \left(\sum_{k=1}^{n_j} s_{jk} x_{jk} \right),$$

subject to

$$\sum_{j=1}^{n} a_{ij} \left(\sum_{k=1}^{n_j} x_{jk} \right) \le b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

$$x_{jk} \le u_{jk}, \qquad \text{for } k = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, n$$

and

$$x_{jk} \ge 0, \qquad \text{for} \qquad k = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, n.$$

(The $\Sigma_{k=1}^{n_j} x_{jk} \ge 0$ constraints are deleted because they are ensured by the $x_{jk} \ge 0$ constraints.) If some original variable x_j has no upper bound, then $u_{jn_j} = \infty$, so the constraint involving this quantity will be deleted.

[1] If one or more of the $f_j(x_j)$ already are *linear* functions $f_j(x_j) = c_j x_j$, then $n_j = 1$ so neither of these substitutions will be made for j.

An efficient way of solving this model[1] is to use the streamlined version of the simplex method for dealing with upper bound constraints (described in Sec. 7.3). After obtaining an optimal solution for this model, you then would calculate

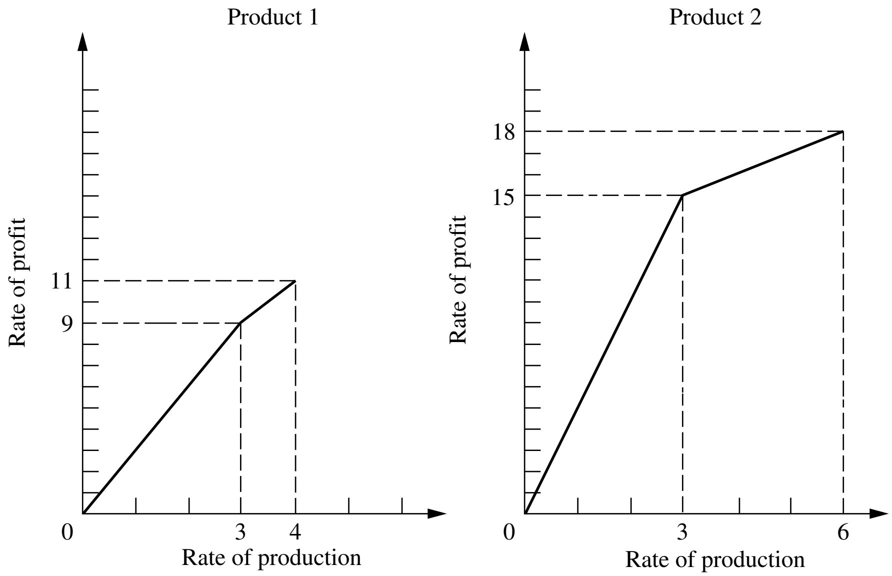
$$x_j = \sum_{k=1}^{n_j} x_{jk},$$

for $j = 1, 2, \ldots, n$ in order to identify an optimal solution for the original separable programming program (or its piecewise linear approximation).

Example. The Wyndor Glass Co. (see Sec. 3.1) has received a special order for handcrafted goods to be made in Plants 1 and 2 throughout the next 4 months. Filling this order will require borrowing certain employees from the work crews for the regular products, so the remaining workers will need to work overtime to utilize the full production capacity of the plant's machinery and equipment for these regular products. In particular, for the two new regular products discussed in Sec. 3.1, overtime will be required to utilize the last 25 percent of the production capacity available in Plant 1 for product 1 and for the last 50 percent of the capacity available in Plant 2 for product 2. The additional cost of using overtime work will reduce the profit for each unit involved from $3 to $2 for product 1 and from $5 to $1 for product 2, giving the *profit curves* of Fig. 13.16, both of which fit the form for case 1 of Fig. 13.15.

[1] For a specialized algorithm for solving this model very efficiently, see R. Fourer, "A Specialized Algorithm for Piecewise-Linear Programming III: Computational Analysis and Applications," *Mathematical Programming,* **53:** 213–235, 1992. Also see A. M. Geoffrion, "Objective Function Approximations in Mathematical Programming," *Mathematical Programming,* **13:** 23–37, 1977.

FIGURE 13.16
Profit data during the next 4 months for the Wyndor Glass Co.



Management has decided to go ahead and use overtime work rather than hire additional workers during this temporary situation. However, it does insist that the work crew for each product be fully utilized on regular time before any overtime is used. Furthermore, it feels that the current production rates ($x_1 = 2$ for product 1 and $x_2 = 6$ for product 2) should be changed temporarily if this would improve overall profitability. Therefore, it has instructed the OR team to review products 1 and 2 again to determine the most profitable product mix during the next 4 months.

Formulation. To refresh your memory, the linear programming model for the original Wyndor Glass Co. problem in Sec. 3.1 is

$$\text{Maximize} \qquad Z = 3x_1 + 5x_2,$$

subject to

$$\begin{aligned} x_1 \qquad\quad &\le 4 \\ 2x_2 &\le 12 \\ 3x_1 + 2x_2 &\le 18 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

We now need to modify this model to fit the new situation described above. For this purpose, let the production rate for product 1 be $x_1 = x_{1R} + x_{1O}$, where x_{1R} is the production rate achieved on regular time and x_{1O} is the incremental production rate from using overtime. Define $x_2 = x_{2R} + x_{2O}$ in the same way for product 2. Thus, in the notation of the general linear programming model for separable programming given just before this example, $n = 2$, $n_1 = 2$, and $n_2 = 2$. Plugging the data given in Fig. 13.16 (including maximum rates of production on regular time and on overtime) into this general model gives the specific model for this application. In particular, the new linear programming problem is to determine the values of x_{1R}, x_{1O}, x_{2R}, and x_{2O} so as to

$$\text{Maximize} \qquad Z = 3x_{1R} + 2x_{1O} + 5x_{2R} + x_{2O},$$

subject to

$$\begin{aligned} x_{1R} + x_{1O} \qquad\qquad\qquad\qquad &\le 4 \\ 2(x_{2R} + x_{2O}) &\le 12 \\ 3(x_{1R} + x_{1O}) + 2(x_{2R} + x_{2O}) &\le 18 \\ x_{1R} \le 3, \qquad x_{1O} \le 1, \qquad x_{2R} \le 3, \qquad x_{2O} &\le 3 \end{aligned}$$

and

$$x_{1R} \ge 0, \qquad x_{1O} \ge 0, \qquad x_{2R} \ge 0, \qquad x_{2O} \ge 0.$$

(Note that the upper bound constraints in the next-to-last row of the model make the first two functional constraints *redundant,* so these two functional constraints can be deleted.)

However, there is one important factor that is not taken into account explicitly in this formulation. Specifically, there is nothing in the model that requires all available regular time for a product to be fully utilized before any overtime is used for that product. In other words, it may be feasible to have $x_{1O} > 0$ even when $x_{1R} < 3$ and to have $x_{2O} > 0$

even when $x_{2R} < 3$. Such solutions would not, however, be acceptable to management. (Prohibiting such solutions is the *special restriction* discussed earlier in this section.)

Now we come to the *key property of separable programming.* Even though the model does not take this factor into account explicitly, the model does take it into account implicitly! Despite the model's having excess "feasible" solutions that actually are unacceptable, any *optimal* solution for the model is *guaranteed* to be a legitimate one that does not replace any available regular-time work with overtime work. (The reasoning here is analogous to that for the Big M method discussed in Sec. 4.6, where excess feasible but *nonoptimal* solutions also were allowed in the model as a matter of convenience.) Therefore, the simplex method can be safely applied to this model to find the most profitable acceptable product mix. The reasons are twofold. First, the two decision variables for each product *always* appear together as a *sum,* $x_{1R} + x_{1O}$ or $x_{2R} + x_{2O}$, in *each* functional constraint other than the upper bound constraints on individual variables. Therefore, it *always* is possible to convert an unacceptable feasible solution to an acceptable one having the same total production rates, $x_1 = x_{1R} + x_{1O}$ and $x_2 = x_{2R} + x_{2O}$, merely by replacing overtime production by regular-time production as much as possible. Second, overtime production is less profitable than regular-time production (i.e., the slope of each profit curve in Fig. 13.16 is a monotonic *decreasing* function of the rate of production), so converting an unacceptable feasible solution to an acceptable one in this way *must* increase the total rate of profit Z. Consequently, any feasible solution that uses overtime production for a product when regular-time production is still available *cannot* be optimal with respect to the model.

For example, consider the unacceptable feasible solution $x_{1R} = 1$, $x_{1O} = 1$, $x_{2R} = 1$, $x_{2O} = 3$, which yields a total rate of profit $Z = 13$. The acceptable way of achieving the same total production rates $x_1 = 2$ and $x_2 = 4$ is $x_{1R} = 2$, $x_{1O} = 0$, $x_{2R} = 3$, $x_{2O} = 1$. This latter solution is still feasible, but it also increases Z by $(3 - 2)(1) + (5 - 1)(2) = 9$ to a total rate of profit $Z = 22$.

Similarly, the optimal solution for this model turns out to be $x_{1R} = 3$, $x_{1O} = 1$, $x_{2R} = 3$, $x_{2O} = 0$, which is an acceptable feasible solution.

Extensions

Thus far we have focused on the special case of separable programming where the only nonlinear function is the objective function $f(\mathbf{x})$. Now consider briefly the general case where the constraint functions $g_i(\mathbf{x})$ need not be linear but are convex and separable, so that each $g_i(\mathbf{x})$ can be expressed as a sum of functions of individual variables

$$g_i(\mathbf{x}) = \sum_{j=1}^{n} g_{ij}(x_j),$$

where each $g_{ij}(x_j)$ is a *convex* function. Once again, each of these new functions may be approximated as closely as desired by a *piecewise linear* function (if it is not already in that form). The one new restriction is that for each variable x_j ($j = 1, 2, \ldots, n$), all the piecewise linear approximations of the functions of this variable $[f_j(x_j), g_{1j}(x_j), \ldots, g_{mj}(x_j)]$ must have the *same* breakpoints so that the same new variables $(x_{j1}, x_{j2}, \ldots, x_{jn_j})$ can be used for all these piecewise linear functions. This formulation leads to a linear programming model just like the one given for the special case except that for each i and j, the x_{jk} variables now have different coefficients in constraint i [where these coefficients are the corresponding slopes of the piecewise linear function approximating $g_{ij}(x_j)$]. Because the $g_{ij}(x_j)$ are required to be convex, essentially the same logic as before implies that the key property of separable programming still must hold. (See Prob. 13.8-8*b*.)

One drawback of approximating functions by piecewise linear functions as described in this section is that achieving a close approximation requires a large number of line segments (variables), whereas such a fine grid for the breakpoints is needed only in the immediate neighborhood of an optimal solution. Therefore, more sophisticated approaches that use a succession of *two-segment* piecewise linear functions have been developed[1] to obtain *successively closer approximations* within this immediate neighborhood. This kind of approach tends to be both faster and more accurate in closely approximating an optimal solution.

13.9 CONVEX PROGRAMMING

We already have discussed some special cases of convex programming in Secs. 13.4 and 13.5 (unconstrained problems), 13.7 (quadratic objective function with linear constraints), and 13.8 (separable functions). You also have seen some theory for the general case (necessary and sufficient conditions for optimality) in Sec. 13.6. In this section, we briefly discuss some types of approaches used to solve the general convex programming problem [where the objective function $f(\mathbf{x})$ to be maximized is concave and the $g_i(\mathbf{x})$ constraint functions are convex], and then we present one example of an algorithm for convex programming.

There is no single standard algorithm that always is used to solve convex programming problems. Many different algorithms have been developed, each with its own advantages and disadvantages, and research continues to be active in this area. Roughly speaking, most of these algorithms fall into one of the following three categories.

The first category is **gradient algorithms,** where the gradient search procedure of Sec. 13.5 is modified in some way to keep the search path from penetrating any constraint boundary. For example, one popular gradient method is the *generalized reduced gradient* (GRG) method.

The second category—**sequential unconstrained algorithms**—includes *penalty function* and *barrier function* methods. These algorithms convert the original constrained optimization problem to a sequence of *unconstrained optimization* problems whose optimal solutions converge to the optimal solution for the original problem. Each of these unconstrained optimization problems can be solved by the gradient search procedure of Sec. 13.5. This conversion is accomplished by incorporating the constraints into a penalty function (or barrier function) that is subtracted from the objective function in order to impose large penalties for violating constraints (or even being near constraint boundaries). You will see one example of this category of algorithms in the next section.

The third category—**sequential-approximation algorithms**—includes *linear approximation* and *quadratic approximation* methods. These algorithms replace the nonlinear objective function by a succession of linear or quadratic approximations. For linearly

[1] R. R. Meyer, "Two-Segment Separable Programming," *Management Science,* **25:** 385–395, 1979.

constrained optimization problems, these approximations allow repeated application of linear or quadratic programming algorithms. This work is accompanied by other analysis that yields a sequence of solutions that converges to an optimal solution for the original problem. Although these algorithms are particularly suitable for linearly constrained optimization problems, some also can be extended to problems with nonlinear constraint functions by the use of appropriate linear approximations.

As one example of a *sequential-approximation* algorithm, we present here the **Frank-Wolfe algorithm**[1] for the case of *linearly constrained* convex programming (so the constraints are $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ in matrix form). This procedure is particularly straightforward; it combines *linear* approximations of the objective function (enabling us to use the simplex method) with the one-dimensional search procedure of Sec. 13.4.

OR TUTOR

A Sequential Linear Approximation Algorithm (Frank-Wolfe)

Given a feasible trial solution $\mathbf{x}'$, the linear approximation used for the objective function $f(\mathbf{x})$ is the first-order Taylor series expansion of $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}'$, namely,

$$f(\mathbf{x}') \approx f(\mathbf{x}') + \sum_{j=1}^{n} \frac{\partial f(\mathbf{x}')}{\partial x_j}(x_j - x_j') = f(\mathbf{x}') + \nabla f(\mathbf{x}')(\mathbf{x} - \mathbf{x}'),$$

where these partial derivatives are evaluated at $\mathbf{x} = \mathbf{x}'$. Because $f(\mathbf{x}')$ and $\nabla f(\mathbf{x}')\mathbf{x}'$ have fixed values, they can be dropped to give an equivalent linear objective function

$$g(\mathbf{x}) = \nabla f(\mathbf{x}')\mathbf{x} = \sum_{j=1}^{n} c_j x_j, \qquad \text{where } c_j = \frac{\partial f(\mathbf{x})}{\partial x_j} \qquad \text{at } \mathbf{x} = \mathbf{x}'.$$

The simplex method (or the graphical procedure if $n = 2$) then is applied to the resulting linear programming problem [maximize $g(\mathbf{x})$ subject to the original constraints, $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$] to find *its* optimal solution x_{LP}. Note that the linear objective function necessarily increases steadily as one moves along the line segment from $\mathbf{x}'$ to $\mathbf{x}_{\text{LP}}$ (which is on the boundary of the feasible region). However, the linear approximation may not be a particularly close one for $\mathbf{x}$ far from $\mathbf{x}'$, so the *nonlinear* objective function may not continue to increase all the way from $\mathbf{x}'$ to $\mathbf{x}_{\text{LP}}$. Therefore, rather than just accepting $\mathbf{x}_{\text{LP}}$ as the next trial solution, we choose the point that maximizes the nonlinear objective function along this line segment. This point may be found by conducting the *one-dimensional search procedure* of Sec. 13.4, where the one variable for purposes of this search is the fraction t of the total distance from $\mathbf{x}'$ to $\mathbf{x}_{\text{LP}}$. This point then becomes the new trial solution for initiating the next iteration of the algorithm, as just described. The sequence of trial solutions generated by repeated iterations converges to an optimal solution for the original problem, so the algorithm stops as soon as the successive trial solutions are close enough together to have essentially reached this optimal solution.

[1] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly,* **3:** 95–110, 1956. Although originally designed for quadratic programming, this algorithm is easily adapted to the case of a general concave objective function considered here.

Summary of the Frank-Wolfe Algorithm.

Initialization: Find a feasible initial trial solution $\mathbf{x}^{(0)}$, for example, by applying linear programming procedures to find an initial BF solution. Set $k = 1$.

Iteration:

1. For $j = 1, 2, \ldots, n$, evaluate

$$\frac{\partial f(\mathbf{x})}{\partial x_j} \qquad \text{at } \mathbf{x} = \mathbf{x}^{(k-1)}$$

and set c_j equal to this value.

2. Find an optimal solution $\mathbf{x}_{\text{LP}}^{(k)}$ for the following linear programming problem.

$$\text{Maximize} \qquad g(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j,$$

subject to

$$\mathbf{Ax} \le \mathbf{b} \qquad \text{and} \qquad \mathbf{x} \ge \mathbf{0}.$$

3. For the variable t $(0 \le t \le 1)$, set

$$h(t) = f(\mathbf{x}) \qquad \text{for } \mathbf{x} = \mathbf{x}^{(k-1)} + t(\mathbf{x}_{\text{LP}}^{(k)} - \mathbf{x}^{(k-1)}),$$

so that $h(t)$ gives the value of $f(\mathbf{x})$ on the line segment between $\mathbf{x}^{(k-1)}$ (where $t = 0$) and $\mathbf{x}_{\text{LP}}^{(k)}$ (where $t = 1$). Use some procedure such as the one-dimensional search procedure (see Sec. 13.4) to maximize $h(t)$ over $0 \le t \le 1$, and set $\mathbf{x}^{(k)}$ equal to the corresponding $\mathbf{x}$. Go to the stopping rule.

Stopping rule: If $\mathbf{x}^{(k-1)}$ and $\mathbf{x}^{(k)}$ are sufficiently close, stop and use $\mathbf{x}^{(k)}$ (or some extrapolation of $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k-1)}, \mathbf{x}^{(k)}$) as your estimate of an optimal solution. Otherwise, reset $k = k + 1$ and perform another iteration.

Now let us illustrate this procedure.

Example. Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = 5x_1 - x_1^2 + 8x_2 - 2x_2^2,$$

subject to

$$3x_1 + 2x_2 \le 6$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

Note that

$$\frac{\partial f}{\partial x_1} = 5 - 2x_1, \qquad \frac{\partial f}{\partial x_2} = 8 - 4x_2,$$

so that the *unconstrained* maximum $\mathbf{x} = (\frac{5}{2}, 2)$ violates the functional constraint. Thus, more work is needed to find the *constrained* maximum.

Because $\mathbf{x} = (0, 0)$ is clearly feasible (and corresponds to the initial BF solution for the linear programming constraints), let us choose it as the initial trial solution $\mathbf{x}^{(0)}$ for the Frank-Wolfe algorithm. Plugging $x_1 = 0$ and $x_2 = 0$ into the expressions for the partial derivatives gives $c_1 = 5$ and $c_2 = 8$, so that $g(\mathbf{x}) = 5x_1 + 8x_2$ is the initial linear approximation of the objective function. Graphically, solving this linear programming problem (see Fig. 13.17*a*) yields $\mathbf{x}_{\text{LP}}^{(1)} = (0, 3)$. For step 3 of the first iteration, the points on the line segment between (0, 0) and (0, 3) shown in Fig. 13.17*a* are expressed by

$$\begin{aligned}(x_1, x_2) &= (0, 0) + t[(0, 3) - (0, 0)] \qquad \text{for } 0 \le t \le 1\\ &= (0, 3t)\end{aligned}$$

as shown in the sixth column of Table 13.5. This expression then gives

$$\begin{aligned}h(t) &= f(0, 3t) = 8(3t) - 2(3t)^2\\ &= 24t - 18t^2,\end{aligned}$$

so that the value $t = t^*$ that maximizes $h(t)$ over $0 \le t \le 1$ may be obtained in this case by setting

$$\frac{dh(t)}{dt} = 24 - 36t = 0,$$

so that $t^* = \frac{2}{3}$. This result yields the next trial solution

$$\begin{aligned}\mathbf{x}^{(1)} &= (0, 0) + \frac{2}{3}[(0, 3) - (0, 0)]\\ &= (0, 2),\end{aligned}$$

which completes the first iteration.

FIGURE 13.17
Illustration of the Frank-Wolfe algorithm.

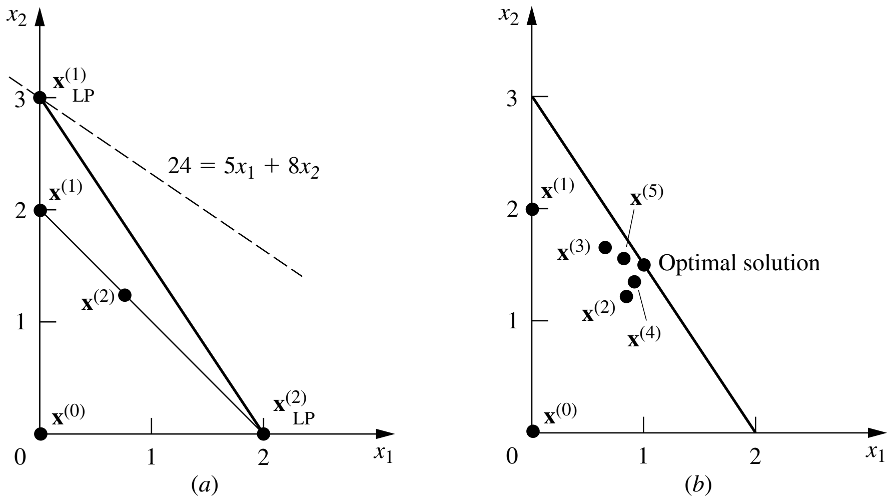


TABLE 13.5 Application of the Frank-Wolfe algorithm to the example

k	$\mathbf{x}^{(k-1)}$	c_1	c_2	$\mathbf{x}_{\text{LP}}^{(k)}$	$\mathbf{x}$ for $h(t)$	$h(t)$	t^*	$\mathbf{x}^{(k)}$
1	(0, 0)	5	8	(0, 3)	(0, 3t)	$24t - 18t^2$	$\frac{2}{3}$	(0, 2)
2	(0, 2)	5	0	(2, 0)	(2t, 2 − 2t)	$8 + 10t - 12t^2$	$\frac{5}{12}$	$\left(\frac{5}{6}, \frac{7}{6}\right)$

To sketch the calculations that lead to the results in the second row of Table 13.5, note that $\mathbf{x}^{(1)} = (0, 2)$ gives

$$\begin{aligned} c_1 &= 5 - 2(0) = 5, \\ c_2 &= 8 - 4(2) = 0. \end{aligned}$$

For the objective function $g(\mathbf{x}) = 5x_1$, graphically solving the problem over the feasible region in Fig. 13.17*a* gives $\mathbf{x}_{\text{LP}}^{(2)} = (2, 0)$. Therefore, the expression for the line segment between $\mathbf{x}^{(1)}$ and $\mathbf{x}_{\text{LP}}^{(2)}$ (see Fig. 13.17*a*) is

$$\begin{aligned} \mathbf{x} &= (0, 2) + t[(2, 0) - (0, 2)] \\ &= (2t, 2 - 2t), \end{aligned}$$

so that

$$\begin{aligned} h(t) &= f(2t, 2 - 2t) \\ &= 5(2t) - (2t)^2 + 8(2 - 2t) - 2(2 - 2t)^2 \\ &= 8 + 10t - 12t^2. \end{aligned}$$

Setting

$$\frac{dh(t)}{dt} = 10 - 24t = 0$$

yields $t^* = \frac{5}{12}$. Hence,

$$\begin{aligned} \mathbf{x}^{(2)} &= (0, 2) + \frac{5}{12}[(2, 0) - (0, 2)] \\ &= \left(\frac{5}{6}, \frac{7}{6}\right). \end{aligned}$$

You can see in Fig. 13.17*b* how the trial solutions keep alternating between two trajectories that appear to intersect at approximately the point $\mathbf{x} = (1, \frac{3}{2})$. This point is, in fact, the optimal solution, as can be verified by applying the KKT conditions from Sec. 13.6.

This example illustrates a common feature of the Frank-Wolfe algorithm, namely, that the trial solutions alternate between two (or more) trajectories. When they alternate in this way, we can extrapolate the trajectories to their approximate point of intersection to estimate an optimal solution. This estimate tends to be better than using the last trial solution generated. The reason is that the trial solutions tend to converge rather slowly toward an optimal solution, so the last trial solution may still be quite far from optimal.

In conclusion, we emphasize that the Frank-Wolfe algorithm is just one example of sequential-approximation algorithms. Many of these algorithms use *quadratic* instead of *linear* approximations at each iteration because quadratic approximations provide a considerably closer fit to the original problem and thus enable the sequence of solutions to converge considerably more rapidly toward an optimal solution than was the case in Fig. 13.17*b*. For this reason, even though sequential linear approximation methods such as the Frank-Wolfe algorithm are relatively straightforward to use, *sequential quadratic approximation methods*[1] now are generally preferred in actual applications. Popular among these are the *quasi-Newton* (or *variable metric*) methods, which compute a quadratic approximation to the curvature of a nonlinear function without explicitly calculating second (partial) derivatives. (For linearly constrained optimization problems, this nonlinear function is just the objective function; whereas with nonlinear constraints, it is the Lagrangian function described in Appendix 3.) Some quasi-Newton algorithms do not even explicitly form and solve an approximating quadratic programming problem at each iteration, but instead incorporate some of the basic ingredients of *gradient algorithms.*

For further information about convex programming algorithms, see Selected References 4 and 6.

Some Software Options

Another example illustrating the application of the Frank-Wolfe algorithm is provided in your OR Tutor. The OR Courseware also includes an interactive routine for this algorithm.

As indicated at the end of Sec. 13.7, both Excel and LINGO can solve convex programming problems, but LINDO and CPLEX cannot except for the special case of quadratic programming (which includes the example in this section). Details for this example are given in the Excel and LINGO/LINDO files for this chapter in your OR Courseware. The professional version of MPL supports a large number of solvers, including some that can handle convex programming. One of these, called CONOPT, is included with the student version of MPL that is on the CD-ROM. The convex programming examples that are formulated in this chapter's MPL file have been solved with this solver after selecting Nonlinear Models for the Default Model Type entry in the MPL Language option dialogue box.

13.10 NONCONVEX PROGRAMMING

The assumptions of convex programming are very convenient ones, because they ensure that any *local maximum* also is a *global maximum.* Unfortunately, the nonlinear programming problems that arise in practice frequently only come fairly close to satisfying these assumptions, but they have some relatively minor disparities. What kind of approach can be used to deal with such *nonconvex programming* problems?

A common approach is to apply an algorithmic *search procedure* that will stop when it finds a *local maximum* and then to restart it a number of times from a variety of initial trial solutions in order to find as many distinct local maxima as possible. The best of these

[1] For a survey of these methods, see M. J. D. Powell, "Variable Metric Methods for Constrained Optimization," in A. Bachem, M. Grotschel, and B. Korte (eds.), *Mathematical Programming: The State of the Art,* Springer-Verlag, Berlin, 1983, pp. 288–311.

local maxima is then chosen for implementation. Normally, the search procedure is one that has been designed to find a global maximum when all the assumptions of convex programming hold, but it also can operate to find a local maximum when they do not.

One such search procedure that has been widely used since its development in the 1960s is the *sequential unconstrained minimization technique* (or *SUMT* for short).[1] There actually are two main versions of SUMT, one of which is an *exterior-point* algorithm that deals with *infeasible* solutions while using a *penalty function* to force convergence to the feasible region. We shall describe the other version, which is an *interior-point* algorithm that deals directly with *feasible* solutions while using a *barrier function* to force staying inside the feasible region. Although SUMT was originally presented as a minimization technique, we shall convert it to a maximization technique in order to be consistent with the rest of the chapter. Therefore, we continue to assume that the problem is in the form given at the beginning of the chapter and that all the functions are differentiable.

OR TUTOR

Sequential Unconstrained Minimization Technique (SUMT)

As the name implies, SUMT replaces the original problem by a *sequence* of *unconstrained* optimization problems whose solutions *converge* to a solution (local maximum) of the original problem. This approach is very attractive because unconstrained optimization problems are much easier to solve (see the gradient search procedure in Sec. 13.5) than those with constraints. Each of the unconstrained problems in this sequence involves choosing a (successively smaller) strictly positive value of a scalar r and then solving for $\mathbf{x}$ so as to

$$\text{Maximize} \qquad P(\mathbf{x}; r) = f(\mathbf{x}) - rB(\mathbf{x}).$$

Here $B(\mathbf{x})$ is a **barrier function** that has the following properties (for $\mathbf{x}$ that are feasible for the original problem):

1. $B(\mathbf{x})$ is *small* when $\mathbf{x}$ is *far* from the boundary of the feasible region.
2. $B(\mathbf{x})$ is *large* when $\mathbf{x}$ is *close* to the boundary of the feasible region.
3. $B(\mathbf{x}) \to \infty$ as the distance from the (nearest) boundary of the feasible region $\to 0$.

Thus, by starting the search procedure with a *feasible* initial trial solution and then attempting to increase $P(\mathbf{x}; r)$, $B(\mathbf{x})$ provides a *barrier* that prevents the search from ever crossing (or even reaching) the boundary of the feasible region for the original problem.

The most common choice of $B(\mathbf{x})$ is

$$B(\mathbf{x}) = \sum_{i=1}^{m} \frac{1}{b_i - g_i(\mathbf{x})} + \sum_{j=1}^{n} \frac{1}{x_j}.$$

For feasible values of $\mathbf{x}$, note that the denominator of each term is proportional to the distance of $\mathbf{x}$ from the constraint boundary for the corresponding functional or nonnegativity constraint. Consequently, *each* term is a *boundary repulsion term* that has all the preceding three properties with respect to this particular constraint boundary. Another attractive feature of this $B(\mathbf{x})$ is that when all the assumptions of *convex programming* are satisfied, $P(\mathbf{x}; r)$ is a *concave* function.

[1] See Selected Reference 3.

Because $B(\mathbf{x})$ keeps the search away from the boundary of the feasible region, you probably are asking the very legitimate question: What happens if the desired solution lies there? This concern is the reason that SUMT involves solving a *sequence* of these unconstrained optimization problems for successively smaller values of r approaching zero (where the final trial solution from each one becomes the initial trial solution for the next). For example, each new r might be obtained from the preceding one by multiplying by a constant $\theta\,(0 < \theta < 1)$, where a typical value is $\theta = 0.01$. As r approaches 0, $P(\mathbf{x}; r)$ approaches $f(\mathbf{x})$, so the corresponding local maximum of $P(\mathbf{x}; r)$ converges to a local maximum of the original problem. Therefore, it is necessary to solve only enough unconstrained optimization problems to permit extrapolating their solutions to this limiting solution.

How many are enough to permit this extrapolation? When the original problem satisfies the assumptions of convex programming, useful information is available to guide us in this decision. In particular, if $\bar{x}$ is a global maximizer of $P(\mathbf{x}; r)$, then

$$f(\bar{\mathbf{x}}) \leq f(\mathbf{x}^*) \leq f(\bar{\mathbf{x}}) + rB(\bar{\mathbf{x}}),$$

where $\mathbf{x}^*$ is the (unknown) *optimal* solution for the original problem. Thus, $rB(\bar{\mathbf{x}})$ is the *maximum error* (in the value of the objective function) that can result by using $\bar{\mathbf{x}}$ to approximate $\mathbf{x}^*$, and extrapolating beyond $\bar{\mathbf{x}}$ to increase $f(\mathbf{x})$ further decreases this error. If an *error tolerance* is established in advance, then you can stop as soon as $rB(\bar{\mathbf{x}})$ is less than this quantity.

Unfortunately, no such guarantee for the maximum error can be given for nonconvex programming problems. However, $rB(\bar{\mathbf{x}})$ still is *likely* to exceed the actual error when $\bar{\mathbf{x}}$ and $\mathbf{x}^*$ now are corresponding *local maxima* of $P(\mathbf{x}; r)$ and the original problem, respectively.

Summary of SUMT.

Initialization: Identify a *feasible* initial trial solution $\mathbf{x}^{(0)}$ that is not on the boundary of the feasible region. Set $k = 1$ and choose appropriate strictly positive values for the initial r and for $\theta < 1$ (say, $r = 1$ and $\theta = 0.01$).[1]

Iteration: Starting from $\mathbf{x}^{(k-1)}$, apply the gradient search procedure described in Sec. 13.5 (or some similar method) to find a local maximum $\mathbf{x}^{(k)}$ of

$$P(\mathbf{x}; r) = f(\mathbf{x}) - r\left[\sum_{i=1}^{m} \frac{1}{b_i - g_i(\mathbf{x})} + \sum_{j=1}^{n} \frac{1}{x_j}\right].$$

Stopping rule: If the change from $\mathbf{x}^{(k-1)}$ to $\mathbf{x}^{(k)}$ is negligible, stop and use $\mathbf{x}^{(k)}$ (or an extrapolation of $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k-1)}, \mathbf{x}^{(k)}$) as your estimate of a *local maximum* of the original problem. Otherwise, reset $k = k + 1$ and $r = \theta r$ and perform another iteration.

When the assumptions of convex programming are not satisfied, this algorithm should be repeated a number of times by starting from a variety of feasible initial trial solutions. The best of the *local maxima* thereby obtained for the original problem should be used as the best available approximation of a *global maximum.*

[1] A reasonable criterion for choosing the initial r is one that makes $rB(\mathbf{x})$ about the same order of magnitude as $f(\mathbf{x})$ for feasible solutions $\mathbf{x}$ that are not particularly close to the boundary.

Finally, note that SUMT also can be extended to accommodate *equality* constraints $g_i(\mathbf{x}) = b_i$. One standard way is as follows. For each equality constraint,

$$\frac{-[b_i - g_i(\mathbf{x})]^2}{\sqrt{r}} \qquad \text{replaces} \qquad \frac{-r}{b_i - g_i(\mathbf{x})}$$

in the expression for $P(\mathbf{x}; r)$ given under "Summary of SUMT," and then the same procedure is used. The numerator $-[b_i - g_i(\mathbf{x})]^2$ imposes a large penalty for deviating substantially from satisfying the equality constraint, and then the denominator tremendously increases this penalty as r is decreased to a tiny amount, thereby forcing the sequence of trial solutions to converge toward a point that satisfies the constraint.

SUMT has been widely used because of its simplicity and versatility. However, numerical analysts have found that it is relatively prone to *numerical instability,* so considerable caution is advised. For further information on this issue as well as similar analyses for alternative algorithms, see Selected Reference 4.

Example. To illustrate SUMT, consider the following two-variable problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = x_1 x_2,$$

subject to

$$x_1^2 + x_2 \le 3$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

Even though $g_1(\mathbf{x}) = x_1^2 + x_2$ is convex (because each term is convex), this problem is a *nonconvex* programming problem because $f(\mathbf{x}) = x_1 x_2$ is *not* concave (see Appendix 2).

For the initialization, $(x_1, x_2) = (1, 1)$ is one obvious feasible solution that is not on the boundary of the feasible region, so we can set $\mathbf{x}^{(0)} = (1, 1)$. Reasonable choices for r and θ are $r = 1$ and $\theta = 0.01$.

For each iteration,

$$P(\mathbf{x}; r) = x_1 x_2 - r\left(\frac{1}{3 - x_1^2 - x_2} + \frac{1}{x_1} + \frac{1}{x_2}\right).$$

With $r = 1$, applying the gradient search procedure starting from (1, 1) to maximize this expression eventually leads to $\mathbf{x}^{(1)} = (0.90, 1.36)$. Resetting $r = 0.01$ and restarting the gradient search procedure from (0.90, 1.36) then lead to $\mathbf{x}^{(2)} = (0.983, 1.933)$. One more iteration with $r = 0.01(0.01) = 0.0001$ leads from $\mathbf{x}^{(2)}$ to $\mathbf{x}^{(3)} = (0.998, 1.994)$. This sequence of points, summarized in Table 13.6, quite clearly is converging to (1, 2). Applying the KKT conditions to this solution verifies that it does indeed satisfy the necessary condition for optimality. Graphical analysis demonstrates that $(x_1, x_2) = (1, 2)$ is, in fact, a global maximum (see Prob. 13.10-4*b*).

For this problem, there are no local maxima other than $(x_1, x_2) = (1, 2)$, so reapplying SUMT from various feasible initial trial solutions always leads to this same solution.[1]

[1] The technical reason is that $f(\mathbf{x})$ is a (strictly) *quasiconcave* function that shares the property of concave functions that a local maximum always is a global maximum. For further information, see M. Avriel, W. E. Diewert, S. Schaible, and I. Zang, *Generalized Concavity,* Plenum, New York, 1985.

TABLE 13.6 Illustration of SUMT

k	r	$x_1^{(k)}$	$x_2^{(k)}$
0		1	1
1	1	0.90	1.36
2	10^{-2}	0.983	1.933
3	10^{-4}	0.998	1.994
		↓	↓
		1	2

Your OR Tutor includes another example illustrating the application of SUMT. The OR Courseware includes an automatic routine for executing SUMT.

13.11 CONCLUSIONS

Practical optimization problems frequently involve *nonlinear* behavior that must be taken into account. It is sometimes possible to *reformulate* these nonlinearities to fit into a linear programming format, as can be done for *separable programming* problems. However, it is frequently necessary to use a *nonlinear programming* formulation.

In contrast to the case of the simplex method for linear programming, there is no efficient all-purpose algorithm that can be used to solve all nonlinear programming problems. In fact, some of these problems cannot be solved in a very satisfactory manner by any method. However, considerable progress has been made for some important classes of problems, including *quadratic programming, convex programming,* and certain special types of *nonconvex programming*. A variety of algorithms that frequently perform well are available for these cases. Some of these algorithms incorporate highly efficient procedures for *unconstrained optimization* for a portion of each iteration, and some use a succession of linear or quadratic approximations to the original problem.

There has been a strong emphasis in recent years on developing high-quality, reliable *software packages* for general use in applying the best of these algorithms. (See Selected Reference 7 for a comprehensive survey of the available software packages for nonlinear programming.) For example, several powerful software packages such as MINOS have been developed in the Systems Optimization Laboratory at Stanford University. These packages are widely used elsewhere for solving many of the types of problems discussed in this chapter (as well as linear programming problems). The steady improvements being made in both algorithmic techniques and software now are bringing some rather large problems into the range of computational feasibility.

Research in nonlinear programming remains very active.

SELECTED REFERENCES

1. Bazaraa, M. S., H. D. Sherali, and C. M. Shetty: *Nonlinear Programming: Theory and Algorithms,* 2d ed., Wiley, New York, 1993.
2. Bertsekas, D. P.: *Nonlinear Programming,* Athena Scientific, Belmont, MA, 1995.

3. Fiacco, A. V., and G. P. McCormick: *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* Classics in Applied Mathematics 4, Society for Industrial and Applied Mathematics, Philadelphia, 1990. (Reprint of a classic book published in 1968.)

4. Gill, P. E., W. Murray, and M. H. Wright: *Practical Optimization,* Academic Press, London, 1981.

5. Horst, R., and P. M. Pardalos (eds.): *Handbook of Global Optimization,* Kluwer Academic Publishers, Boston, 1995.

6. Murray, W., "Algorithms for Large Nonlinear Programming Problems," pp. 172–185 in J. R. Birge and K. G. Murty (eds.), *Mathematical Programming: State of the Art 1994,* 15th International Symposium on Mathematical Programming, University of Michigan, Ann Arbor, MI, 1994.

7. Nash, S. G.: "Software Survey: NLP," *OR/MS Today,* April 1995, pp. 60–71.

8. Nemhauser, G. L., A. H. G. Rinnooy Kan, and M. J. Todd (eds.): *Optimization,* Handbooks in Operations Research and Management Science, Volume 1, North-Holland, Amsterdam, 1989.

9. Nesterov, Y., and A. Nemirovskii: *Interior-Point Polynomial Algorithms in Convex Programming,* Studies in Applied Mathematics 13, Society for Industrial and Applied Mathematics, Philadelphia, 1993.

10. Nocedal, J.: "Recent Advances in Large-Scale Nonlinear Optimization," pp. 208–219 in J. R. Birge and K. G. Murty (eds.), *Mathematical Programming: State of the Art 1994,* 15th International Symposium on Mathematical Programming, University of Michigan, Ann Arbor, MI, 1994.

LEARNING AIDS FOR THIS CHAPTER IN YOUR OR COURSEWARE

Demonstration Examples in OR Tutor:

Gradient Search Procedure
Frank-Wolfe Algorithm
Sequential Unconstrained Minimization Technique—SUMT

Interactive Routines:

Interactive One-Dimensional Search Procedure
Interactive Gradient Search Procedure
Interactive Modified Simplex Method
Interactive Frank-Wolfe Algorithm

Automatic Routines:

Automatic Gradient Search Procedure
Sequential Unconstrained Minimization Technique—SUMT

An Excel Add-in:

Premium Solver

OR TUTOR

"Ch. 13—Nonlinear Programming" Files for Solving the Examples:

Excel File
LINGO/LINDO File
MPL/CPLEX/CONOPT File

See Appendix 1 for documentation of the software.

PROBLEMS

The symbols to the left of some of the problems (or their parts) have the following meaning:

D: The corresponding demonstration example listed above may be helpful.

I: We suggest that you use the corresponding interactive routine listed above (the printout records your work).

C: Use the computer with any of the software options available to you (or as instructed by your instructor) to solve the problem.

An asterisk on the problem number indicates that at least a partial answer is given in the back of the book.

13.1-1. Consider the *product mix* problem described in Prob. 3.1-11. Suppose that this manufacturing firm actually encounters *price elasticity* in selling the three products, so that the profits would be different from those stated in Chap. 3. In particular, suppose that the unit costs for producing products 1, 2, and 3 are \$25, \$10, and \$15, respectively, and that the prices required (in dollars) in order to be able to sell x_1, x_2, and x_3 units are $(35 + 100x_1^{-\frac{1}{3}})$, $(15 + 40x_2^{-\frac{1}{4}})$, and $(20 + 50x_3^{-\frac{1}{2}})$, respectively.

Formulate a nonlinear programming model for the problem of determining how many units of each product the firm should produce to maximize profit.

13.1-2. For the P & T Co. problem described in Sec. 8.1, suppose that there is a 10 percent discount in the shipping cost for all truckloads *beyond* the first 40 for each combination of cannery and warehouse. Draw figures like Figs. 13.3 and 13.4, showing the marginal cost and total cost for shipments of truckloads of peas from cannery 1 to warehouse 1. Then describe the overall nonlinear programming model for this problem.

13.1-3. A stockbroker, Richard Smith, has just received a call from his most important client, Ann Hardy. Ann has \$50,000 to invest, and wants to use it to purchase two stocks. Stock 1 is a solid blue-chip security with a respectable growth potential and little risk involved. Stock 2 is much more speculative. It is being touted in two investment newsletters as having outstanding growth potential, but also is considered very risky. Ann would like a large return on her investment, but also has considerable aversion to risk. Therefore, she has instructed Richard to analyze what mix of investments in the two stocks would be appropriate for her.

Ann is used to talking in units of thousands of dollars and 1,000-share blocks of stocks. Using these units, the price per block is 20 for stock 1 and 30 for stock 2. After doing some research, Richard has made the following estimates. The expected return per block is 5 for stock 1 and 10 for stock 2. The variance of the return on each block is 4 for stock 1 and 100 for stock 2. The covariance of the return on one block each of the two stocks is 5.

Without yet assigning a specific numerical value to the minimum acceptable expected return, formulate a nonlinear programming model for this problem.

13.2-1. Reconsider Prob. 13.1-1. Verify that this problem is a convex programming problem.

13.2-2. Reconsider Prob. 13.1-3. Show that the model formulated is a convex programming problem by using the test in Appendix 2 to show that the objective function being minimized is convex.

13.2-3. Consider the variation of the Wyndor Glass Co. example represented in Fig. 13.5, where the second and third functional constraints of the original problem (see Sec. 3.1) have been replaced by $9x_1^2 + 5x_2^2 \leq 216$. Demonstrate that $(x_1, x_2) = (2, 6)$ with $Z = 36$ is indeed optimal by showing that the objective function line $36 = 3x_1 + 5x_2$ is *tangent* to this constraint boundary at $(2, 6)$. (*Hint:* Express x_2 in terms of x_1 on this boundary, and then differentiate this expression with respect to x_1 to find the slope of the boundary.)

13.2-4. Consider the variation of the Wyndor Glass Co. problem represented in Fig. 13.6, where the original objective function (see Sec. 3.1) has been replaced by $Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$. Demonstrate that $(x_1, x_2) = (\frac{8}{3}, 5)$ with $Z = 857$ is indeed optimal by showing that the ellipse $857 = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$ is *tangent* to the constraint boundary $3x_1 + 2x_2 = 18$ at $(\frac{8}{3}, 5)$. (*Hint:* Solve for x_2 in terms of x_1 for the ellipse, and then differentiate this expression with respect to x_1 to find the slope of the ellipse.)

13.2-5. Consider the following function:

$$f(x) = 48x - 60x^2 + x^3.$$

(a) Use the first and second derivatives to find the local maxima and local minima of $f(x)$.

(b) Use the first and second derivatives to show that $f(x)$ has neither a global maximum nor a global minimum because it is unbounded in both directions.

13.2-6. For each of the following functions, show whether it is convex, concave, or neither.

(a) $f(x) = 10x - x^2$

(b) $f(x) = x^4 + 6x^2 + 12x$

(c) $f(x) = 2x^3 - 3x^2$

(d) $f(x) = x^4 + x^2$

(e) $f(x) = x^3 + x^4$

13.2-7.* For each of the following functions, use the test given in Appendix 2 to determine whether it is convex, concave, or neither.

(a) $f(\mathbf{x}) = x_1x_2 - x_1^2 - x_2^2$
(b) $f(\mathbf{x}) = 3x_1 + 2x_1^2 + 4x_2 + x_2^2 - 2x_1x_2$
(c) $f(\mathbf{x}) = x_1^2 + 3x_1x_2 + 2x_2^2$
(d) $f(\mathbf{x}) = 20x_1 + 10x_2$
(e) $f(\mathbf{x}) = x_1x_2$

13.2-8. Consider the following function:

$$f(\mathbf{x}) = 5x_1 + 2x_2^2 + x_3^2 - 3x_3x_4 + 4x_4^2 + 2x_5^4 + x_5^2 + 3x_5x_6 + 6x_6^2 + 3x_6x_7 + x_7^2.$$

Show that $f(\mathbf{x})$ is convex by expressing it as a sum of functions of one or two variables and then showing (see Appendix 2) that all these functions are convex.

13.2-9. Consider the following nonlinear programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = x_1 + x_2,$$

subject to

$$x_1^2 + x_2^2 \le 1$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Verify that this is a convex programming problem.
(b) Solve this problem graphically.

13.2-10. Consider the following nonlinear programming problem:

$$\text{Minimize} \quad Z = x_1^4 + 2x_2^2,$$

subject to

$$x_1^2 + x_2^2 \ge 2.$$

(No nonnegativity constraints.)

(a) Use geometric analysis to determine whether the feasible region is a convex set.
(b) Now use algebra and calculus to determine whether the feasible region is a convex set.

13.3-1. Reconsider Prob. 13.1-2. Show that this problem is a nonconvex programming problem.

13.3-2. Consider the following constrained optimization problem:

$$\text{Maximize} \quad f(x) = -6x + 3x^2 - 2x^3,$$

subject to

$$x \ge 0.$$

Use just the first and second derivatives of $f(x)$ to derive an optimal solution.

13.3-3. Consider the following nonlinear programming problem:

$$\text{Minimize} \quad Z = x_1^4 + 2x_1^2 + 2x_1x_2 + 4x_2^2,$$

subject to

$$\begin{aligned} 2x_1 + x_2 &\ge 10 \\ x_1 + 2x_2 &\ge 10 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Of the special types of nonlinear programming problems described in Sec. 13.3, to which type or types can this particular problem be fitted? Justify your answer.
(b) Now suppose that the problem is changed slightly by replacing the nonnegativity constraints by $x_1 \ge 1$ and $x_2 \ge 1$. Convert this new problem to an equivalent problem that has just two functional constraints, two variables, and two nonnegativity constraints.

13.3-4. Consider the following geometric programming problem:

$$\text{Minimize} \quad f(\mathbf{x}) = 2x_1^{-2}x_2^{-1} + x_2^{-2},$$

subject to

$$4x_1x_2 + x_1^2x_2^2 \le 12$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Transform this problem to an equivalent convex programming problem.
(b) Use the test given in Appendix 2 to verify that the model formulated in part *(a)* is indeed a convex programming problem.

13.3-5. Consider the following linear fractional programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = \frac{10x_1 + 20x_2 + 10}{3x_1 + 4x_2 + 20},$$

subject to

$$\begin{aligned} x_1 + 3x_2 &\le 50 \\ 3x_1 + 2x_2 &\le 80 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Transform this problem to an equivalent linear programming problem.
C **(b)** Use the computer to solve the model formulated in part *(a)*. What is the resulting optimal solution for the original problem?

13.3-6. Consider the expressions in matrix notation given in Sec. 13.7 for the general form of the KKT conditions for the quadratic programming problem. Show that the problem of finding a feasible solution for these conditions is a linear complementarity problem, as introduced in Sec. 13.3, by identifying **w**, **z**, **q**, and **M** in terms of the vectors and matrices in Sec. 13.7.

I **13.4-1.*** Use the one-dimensional search procedure to interactively solve (approximately) the following problem:

$$\text{Maximize} \quad f(x) = x^3 + 2x - 2x^2 - 0.25x^4.$$

Use an error tolerance $\epsilon = 0.04$ and initial bounds $\underline{x} = 0$, $\bar{x} = 2.4$.

I **13.4-2.** Use the one-dimensional search procedure with an error tolerance $\epsilon = 0.04$ and with the following initial bounds to interactively solve (approximately) each of the following problems.

(a) Maximize $f(x) = 6x - x^2$, with $\underline{x} = 0$, $\bar{x} = 4.8$.

(b) Minimize $f(x) = 6x + 7x^2 + 4x^3 + x^4$, with $\underline{x} = -4$, $\bar{x} = 1$.

I **13.4-3.** Use the one-dimensional search procedure to interactively solve (approximately) the following problem:

$$\text{Maximize} \quad f(x) = 48x^5 + 42x^3 + 3.5x - 16x^6 - 61x^4 - 16.5x^2.$$

Use an error tolerance $\epsilon = 0.08$ and initial bounds $\underline{x} = -1$, $\bar{x} = 4$.

I **13.4-4.** Use the one-dimensional search procedure to interactively solve (approximately) the following problem:

$$\text{Maximize} \quad f(x) = x^3 + 30x - x^6 - 2x^4 - 3x^2.$$

Use an error tolerance $\epsilon = 0.07$ and find appropriate initial bounds by inspection.

13.4-5. Consider the following convex programming problem:

$$\text{Minimize} \quad Z = x^4 + x^2 - 4x,$$

subject to

$$x \le 2 \quad \text{and} \quad x \ge 0.$$

(a) Use one simple calculation *just* to check whether the optimal solution lies in the interval $0 \le x \le 1$ or the interval $1 \le x \le 2$. (Do *not* actually solve for the optimal solution in order to determine in which interval it must lie.) Explain your logic.

I **(b)** Use the one-dimensional search procedure with initial bounds $\underline{x} = 0$, $\bar{x} = 2$ and with an error tolerance $\epsilon = 0.02$ to interactively solve (approximately) this problem.

13.4-6. Consider the problem of maximizing a differentiable function $f(x)$ of a single unconstrained variable x. Let $\underline{x}_0$ and $\bar{x}_0$, respectively, be a valid lower bound and upper bound on the same global maximum (if one exists). Prove the following general properties of the one-dimensional search procedure (as presented in Sec. 13.4) for attempting to solve such a problem.

(a) Given $\underline{x}_0$, $\bar{x}_0$, and $\epsilon = 0$, the sequence of trial solutions selected by the *midpoint rule* must *converge* to a limiting solution. [*Hint:* First show that $\lim_{n\to\infty}(\bar{x}_n - \underline{x}_n) = 0$, where $\bar{x}_n$ and $\underline{x}_n$ are the upper and lower bounds identified at iteration n.]

(b) If $f(x)$ is concave [so that $df(x)/dx$ is a monotone decreasing function of x], then the limiting solution in part (*a*) must be a global maximum.

(c) If $f(x)$ is not concave everywhere, but would be concave if its domain were restricted to the interval between $\underline{x}_0$ and $\bar{x}_0$, then the limiting solution in part (*a*) must be a global maximum.

(d) If $f(x)$ is not concave even over the interval between $\underline{x}_0$ and $\bar{x}_0$, then the limiting solution in part (*a*) need not be a global maximum. (Prove this by graphically constructing a counterexample.)

(e) If $df(x)/dx < 0$ for all x, then no $\underline{x}_0$ exists. If $df(x)/dx > 0$ for all x, then no $\bar{x}_0$ exists. In either case, $f(x)$ does not possess a global maximum.

(f) If $f(x)$ is concave and $\lim_{x\to-\infty} f(x)/dx < 0$, then no $\underline{x}_0$ exists. If $f(x)$ is concave and $\lim_{x\to\infty} df(x)/dx > 0$, then no $\bar{x}_0$ exists. In either case, $f(x)$ does not possess a global maximum.

I **13.4-7.** Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 32x_1 + 50x_2 - 10x_2^2 + x_2^3 - x_1^4 - x_2^4,$$

subject to

$$3x_1 + x_2 \le 11$$
$$2x_1 + 5x_2 \le 16$$

and

$$x_1 \ge 0, \quad x_2 \ge 0.$$

Ignore the constraints and solve the resulting two *one-variable unconstrained optimization* problems. Use calculus to solve the problem involving x_1 and use the *one-dimensional search procedure* with $\epsilon = 0.001$ and initial bounds 0 and 4 to solve the problem involving x_2. Show that the resulting solution for (x_1, x_2) satisfies all of the constraints, so it is actually optimal for the original problem.

13.5-1. Consider the following unconstrained optimization problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 2x_1x_2 + x_2 - x_1^2 - 2x_2^2.$$

D,I **(a)** Starting from the initial trial solution $(x_1, x_2) = (1, 1)$, interactively apply the gradient search procedure with $\epsilon = 0.25$ to obtain an approximate solution.

(b) Solve the system of linear equations obtained by setting $\nabla f(\mathbf{x}) = \mathbf{0}$ to obtain the exact solution.

(c) Referring to Fig 13.14 as a sample for a similar problem, draw the path of trial solutions you obtained in part (*a*). Then show the apparent *continuation* of this path with your best guess for the next three trial solutions [based on the pattern in part (*a*) and in Fig. 13.14]. Also show the exact solution from part (*b*) toward which this sequence of trial solutions is converging.

C **(d)** Apply the automatic routine for the gradient search procedure (with $\epsilon = 0.01$) in your OR Courseware to this problem.

13.5-2. Repeat the four parts of Prob. 13.5-1 (except with $\epsilon = 0.5$) for the following unconstrained optimization problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 2x_1x_2 - 2x_1^2 - x_2^2.$$

D,I,C **13.5-3.** Starting from the initial trial solution $(x_1, x_2) = (1, 1)$, interactively apply two iterations of the gradient search procedure to begin solving the following problem, and then apply the automatic routine for this procedure (with $\epsilon = 0.01$).

$$\text{Maximize} \quad f(\mathbf{x}) = 4x_1x_2 - 2x_1^2 - 3x_2^2.$$

Then solve $\nabla f(\mathbf{x}) = \mathbf{0}$ directly to obtain the exact solution.

D,I,C **13.5-4.*** Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, interactively apply the gradient search procedure with $\epsilon = 0.3$ to obtain an approximate solution for the following problem, and then apply the automatic routine for this procedure (with $\epsilon = 0.01$).

$$\text{Maximize} \quad f(\mathbf{x}) = 8x_1 - x_1^2 - 12x_2 - 2x_2^2 + 2x_1x_2.$$

Then solve $\nabla f(\mathbf{x}) = \mathbf{0}$ directly to obtain the exact solution.

D,I,C **13.5-5.** Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, interactively apply two iterations of the gradient search procedure to begin solving the following problem, and then apply the automatic routine for this procedure (with $\epsilon = 0.01$).

$$\text{Maximize} \quad f(\mathbf{x}) = 6x_1 + 2x_1x_2 - 2x_2 - 2x_1^2 - x_2^2.$$

Then solve $\nabla f(\mathbf{x}) = \mathbf{0}$ directly to obtain the exact solution.

13.5-6. Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, apply *one* iteration of the gradient search procedure to the following problem by hand:

$$\text{Maximize} \quad f(\mathbf{x}) = 4x_1 + 2x_2 + x_1^2 - x_1^4 - 2x_1x_2 - x_2^2.$$

To complete this iteration, approximately solve for t^* by manually applying *two* iterations of the one-dimensional search procedure with initial bounds $\underline{t} = 0$, $\bar{t} = 1$.

13.5-7. Consider the following unconstrained optimization problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 3x_1x_2 + 3x_2x_3 - x_1^2 - 6x_2^2 - x_3^2.$$

(a) Describe how solving this problem can be reduced to solving a *two-variable* unconstrained optimization problem.

D,I **(b)** Starting from the initial trial solution $(x_1, x_2, x_3) = (1, 1, 1)$, interactively apply the gradient search procedure with $\epsilon = 0.05$ to solve (approximately) the two-variable problem identified in part (*a*).

C **(c)** Repeat part (*b*) with the automatic routine for this procedure (with $\epsilon = 0.005$).

D,I,C **13.5-8.*** Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, interactively apply the *gradient search procedure* with $\epsilon = 1$ to solve (approximately) each of the following problems, and then apply the automatic routine for this procedure (with $\epsilon = 0.01$).

(a) Maximize $\quad f(\mathbf{x}) = x_1x_2 + 3x_2 - x_1^2 - x_2^2.$

(b) Minimize $\quad f(\mathbf{x}) = x_1^2x_2^2 + 2x_1^2 + 2x_2^2 - 4x_1 + 4x_2.$

13.6-1. Reconsider the one-variable convex programming model given in Prob. 13.4-5. Use the KKT conditions to derive an optimal solution for this model.

13.6-2. Reconsider Prob. 13.2-9. Use the KKT conditions to check whether $(x_1, x_2) = (1/\sqrt{2}, 1/\sqrt{2})$ is optimal.

13.6-3.* Reconsider the model given in Prob. 13.3-3. What are the KKT conditions for this model? Use these conditions to determine whether $(x_1, x_2) = (0, 10)$ can be optimal.

13.6-4. Consider the following convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 24x_1 - x_1^2 + 10x_2 - x_2^2,$$

subject to

$$x_1 \le 8,$$
$$x_2 \le 7,$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Use the KKT conditions for this problem to derive an optimal solution.

(b) Decompose this problem into two separate constrained optimization problems involving just x_1 and just x_2, respectively. For each of these two problems, plot the objective function over the feasible region in order to *demonstrate* that the value of x_1 or x_2 derived in part (*a*) is indeed optimal. Then *prove* that this value is optimal by using just the first and second derivatives of the objective function and the constraints for the respective problems.

13.6-5. Consider the following linearly constrained optimization problem:

$$\text{Maximize} \quad f(\mathbf{x}) = \ln(1 + x_1 + x_2),$$

subject to

$$x_1 + 2x_2 \le 5$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0,$$

where ln denotes the natural logarithm.

(a) Verify that this problem is a convex programming problem.

(b) Use the KKT conditions to derive an optimal solution.

(c) Use intuitive reasoning to demonstrate that the solution obtained in part (*b*) is indeed optimal. [*Hint:* Note that $\ln(1 + x_1 + x_2)$ is a monotonic strictly increasing function of $1 + x_1 + x_2$.]

13.6-6. Consider the following linearly constrained optimization problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = \ln(x_1 + 1) - x_2^2,$$

subject to

$$x_1 + 2x_2 \leq 3$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0,$$

where ln denotes the natural logarithm,

(a) Verify that this problem is a convex programming problem.

(b) Use the KKT conditions to derive an optimal solution.

(c) Use intuitive reasoning to demonstrate that the solution obtained in part (*b*) is indeed optimal.

13.6-7. Consider the following convex programming problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = 10x_1 - 2x_1^2 - x_1^3 + 8x_2 - x_2^2,$$

subject to

$$x_1 + x_2 \leq 2$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Use the KKT conditions to demonstrate that $(x_1, x_2) = (1, 1)$ is *not* an optimal solution.

(b) Use the KKT conditions to derive an optimal solution.

13.6-8.* Consider the nonlinear programming problem given in Prob. 11.3-14. Determine whether $(x_1, x_2) = (1, 2)$ can be optimal by applying the KKT conditions.

13.6-9. Consider the following nonlinear programming problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = \frac{x_1}{x_2 + 1},$$

subject to

$$x_1 - x_2 \leq 2$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Use the KKT conditions to demonstrate that $(x_1, x_2) = (4, 2)$ is *not* optimal.

(b) Derive a solution that does satisfy the KKT conditions.

(c) Show that this problem is *not* a convex programming problem.

(d) Despite the conclusion in part (*c*), use *intuitive* reasoning to show that the solution obtained in part (*b*) is, in fact, optimal. [The theoretical reason is that $f(\mathbf{x})$ is *pseudo-concave.*]

(e) Use the fact that this problem is a linear fractional programming problem to transform it into an equivalent linear programming problem. Solve the latter problem and thereby identify the optimal solution for the original problem. (*Hint:* Use the equality constraint in the linear programming problem to substitute one of the variables out of the model, and then solve the model graphically.)

13.6-10.* Use the KKT conditions to derive an optimal solution for each of the following problems.

(a) $\quad$ Maximize $\quad f(\mathbf{x}) = x_1 + 2x_2 - x_2^3,$

subject to

$$x_1 + x_2 \leq 1$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(b) $\quad$ Maximize $\quad f(\mathbf{x}) = 20x_1 + 10x_2,$

subject to

$$\begin{aligned} x_1^2 + \ \ x_2^2 &\leq 1 \\ x_1 + 2x_2 &\leq 2 \end{aligned}$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

13.6-11. Reconsider the nonlinear programming model given in Prob. 11.3-16.

(a) Use the KKT conditions to determine whether $(x_1, x_2, x_3) = (1, 1, 1)$ can be optimal.

(b) If a specific solution satisfies the KKT conditions for this problem, can you draw the definite conclusion that this solution is optimal? Why?

13.6-12. What are the KKT conditions for nonlinear programming problems of the following form?

$$\text{Minimize} \qquad f(\mathbf{x}),$$

subject to

$$g_i(\mathbf{x}) \geq b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

and

$$\mathbf{x} \geq \mathbf{0}.$$

(*Hint:* Convert this form to our standard form assumed in this chapter by using the techniques presented in Sec. 4.6 and then applying the KKT conditions as given in Sec. 13.6.)

13.6-13. Consider the following nonlinear programming problem:

$$\text{Minimize} \quad Z = 2x_1^2 + x_2^2,$$

subject to

$$x_1 + x_2 = 10$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Of the special types of nonlinear programming problems described in Sec. 13.3, to which type or types can this particular problem be fitted? Justify your answer. (*Hint:* First convert this problem to an equivalent nonlinear programming problem that fits the form given in the second paragraph of the chapter, with $m = 2$ and $n = 2$.)
(b) Obtain the KKT conditions for this problem.
(c) Use the KKT conditions to derive an optimal solution.

13.6-14. Consider the following linearly constrained programming problem:

$$\text{Minimize} \quad f(\mathbf{x}) = x_1^3 + 4x_2^2 + 16x_3,$$

subject to

$$x_1 + x_2 + x_3 = 5$$

and

$$x_1 \geq 1, \qquad x_2 \geq 1, \qquad x_3 \geq 1.$$

(a) Convert this problem to an equivalent nonlinear programming problem that fits the form given at the beginning of the chapter (second paragraph), with $m = 2$ and $n = 3$.
(b) Use the form obtained in part (*a*) to construct the KKT conditions for this problem.
(c) Use the KKT conditions to check whether $(x_1, x_2, x_3) = (2, 1, 2)$ is optimal.

13.6-15. Consider the following linearly constrained convex programming problem:

$$\text{Minimize} \quad Z = x_1^2 - 6x_1 + x_2^3 - 3x_2,$$

subject to

$$x_1 + x_2 \leq 1$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Obtain the KKT conditions for this problem.
(b) Use the KKT conditions to check whether $(x_1, x_2) = (\frac{1}{2}, \frac{1}{2})$ is an optimal solution.
(c) Use the KKT conditions to derive an optimal solution.

13.6-16. Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 8x_1 - x_1^2 + 2x_2 + x_3,$$

subject to

$$x_1 + 3x_2 + 2x_3 \leq 12$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 \geq 0.$$

(a) Use the KKT conditions to demonstrate that $(x_1, x_2, x_3) = (2, 2, 2)$ is *not* an optimal solution.
(b) Use the KKT conditions to derive an optimal solution. (*Hint:* Do some preliminary intuitive analysis to determine the most promising case regarding which variables are nonzero and which are zero.)

13.6-17. Use the KKT conditions to determine whether $(x_1, x_2, x_3) = (1, 1, 1)$ can be optimal for the following problem:

$$\text{Minimize} \quad Z = 2x_1 + x_2^3 + x_3^2,$$

subject to

$$x_1^2 + 2x_2^2 + x_3^2 \geq 4$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 \geq 0.$$

13.6-18. Reconsider the model given in Prob. 13.2-10. What are the KKT conditions for this problem? Use these conditions to determine whether $(x_1, x_2) = (1, 1)$ can be optimal.

13.6-19. Reconsider the linearly constrained convex programming model given in Prob. 13.4-7. Use the KKT conditions to determine whether $(x_1, x_2) = (2, 2)$ can be optimal.

13.7-1. Consider the quadratic programming example presented in Sec. 13.7.

(a) Use the test given in Appendix 2 to show that the objective function is *strictly concave.*
(b) Verify that the objective function is strictly concave by demonstrating that $\mathbf{Q}$ is a *positive definite* matrix; that is, $\mathbf{x}^T\mathbf{Q}\mathbf{x} > \mathbf{0}$ for all $\mathbf{x} \neq \mathbf{0}$. (*Hint:* Reduce $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ to a sum of squares.)
(c) Show that $x_1 = 12$, $x_2 = 9$, and $u_1 = 3$ satisfy the KKT conditions when they are written in the form given in Sec. 13.6.

13.7-2.* Consider the following quadratic programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 8x_1 - x_1^2 + 4x_2 - x_2^2,$$

subject to

$$x_1 + x_2 \leq 2$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Use the KKT conditions to derive an optimal solution.

(b) Now suppose that this problem is to be solved by the modified simplex method. Formulate the linear programming problem that is to be addressed explicitly, and then identify the additional complementarity constraint that is enforced automatically by the algorithm.

I **(c)** Apply the modified simplex method to the problem as formulated in part (*b*).

C **(d)** Use the computer to solve the quadratic programming problem directly.

13.7-3. Consider the following quadratic programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 20x_1 - 20x_1^2 + 50x_2 - 5x_2^2 + 18x_1x_2,$$

subject to

$$x_1 + x_2 \leq 6$$
$$x_1 + 4x_2 \leq 18$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

Suppose that this problem is to be solved by the modified simplex method.

(a) Formulate the linear programming problem that is to be addressed explicitly, and then identify the additional complementarity constraint that is enforced automatically by the algorithm.

I **(b)** Apply the modified simplex method to the problem as formulated in part (*a*).

13.7-4. Consider the following quadratic programming problem.

$$\text{Maximize} \quad f(\mathbf{x}) = 2x_1 + 3x_2 - x_1^2 - x_2^2,$$

subject to

$$x_1 + x_2 \leq 2$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0.$$

(a) Use the KKT conditions to derive an optimal solution directly.

(b) Now suppose that this problem is to be solved by the modified simplex method. Formulate the linear programming problem that is to be addressed explicitly, and then identify the additional complementarity constraint that is enforced automatically by the algorithm.

(c) Without applying the modified simplex method, show that the solution derived in part (*a*) is indeed optimal ($Z = 0$) for the equivalent problem formulated in part (*b*).

I **(d)** Apply the modified simplex method to the problem as formulated in part (*b*).

C **(e)** Use the computer to solve the quadratic programming problem directly.

13.7-5. Reconsider the first quadratic programming variation of the Wyndor Glass Co. problem presented in Sec. 13.2 (see Fig. 13.6). Analyze this problem by following the instructions of parts (*a*), (*b*), and (*c*) of Prob. 13.7-4.

C **13.7-6.** Reconsider Prob. 13.1-3 and its quadratic programming model.

(a) Display this model [including the values of $R(\mathbf{x})$ and $V(\mathbf{x})$] on an Excel spreadsheet.

(b) Solve this model for four cases: minimum acceptable expected return = 13, 14, 15, 16.

(c) For typical probability distributions (with mean μ and variance σ^2) of the total return from the entire portfolio, the probability is fairly high (about 0.8 or 0.9) that the return will exceed $\mu - \sigma$, and the probability is extremely high (often close to 0.999) that the return will exceed $\mu - 3\sigma$. Calculate $\mu - \sigma$ and $\mu - 3\sigma$ for the four portfolios obtained in part (*b*). Which portfolio will give the highest μ among those that also give $\mu - \sigma \geq 0$?

13.7-7. Jim Matthews, Vice President for Marketing of the J. R. Nickel Company, is planning advertising campaigns for two unrelated products. These two campaigns need to use some of the same resources. Therefore, Jim knows that his decisions on the levels of the two campaigns need to be made jointly after considering these resource constraints. In particular, letting x_1 and x_2 denote the levels of campaigns 1 and 2, respectively, these constraints are $4x_1 + x_2 \leq 20$ and $x_1 + 4x_2 \leq 20$.

In facing these decisions, Jim is well aware that there is a point of diminishing returns when raising the level of an advertising campaign too far. At that point, the cost of additional advertising becomes larger than the increase in net revenue (excluding advertising costs) generated by the advertising. After careful analysis, he and his staff estimate that the net profit from the first product (including advertising costs) when conducting the first campaign at level x_1 would be $3x_1 - (x_1 - 1)^2$ in millions of dollars. The corresponding estimate for the second product is $3x_2 - (x_2 - 2)^2$.

This analysis led to the following quadratic programming model for determining the levels of the two advertising campaigns:

$$\text{Maximize} \quad Z = 3x_1 - (x_1 - 1)^2 + 3x_2 - (x_2 - 2)^2,$$

subject to

$$\begin{aligned} 4x_1 + x_2 &\le 20 \\ x_1 + 4x_2 &\le 20 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Obtain the KKT conditions for this problem in the form given in Sec. 13.6.

(b) You are given the information that the optimal solution does *not* lie on the boundary of the feasible region. Use this information to derive the optimal solution from the KKT conditions.

(c) Now suppose that this problem is to be solved by the modified simplex method. Formulate the linear programming problem that is to be addressed explicitly, and then identify the additional complementarity constraint that is enforced automatically by the algorithm.

(d) Apply the modified simplex method to the problem as formulated in part (*c*).

C **(e)** Use the computer to solve the quadratic programming problem directly.

13.8-1. Reconsider the quadratic programming model given in Prob. 13.7-7.

(a) Use the separable programming formulation presented in Sec. 13.8 to formulate an approximate linear programming model for this problem. Use $x_1, x_2 = 0, 2.5, 5$ as the breakpoints of the piecewise linear functions.

C **(b)** Use the computer to solve the model formulated in part (*a*). Then reexpress this solution in terms of the *original* variables of the problem.

C **(c)** To improve the approximation, now use $x_1, x_2 = 0, 1, 2, 3, 4, 5$ as the breakpoints of the piecewise linear functions and repeat parts (*a*) and (*b*).

13.8-2. The MFG Corporation is planning to produce and market three different products. Let x_1, x_2, and x_3 denote the number of units of the three respective products to be produced. The preliminary estimates of their potential profitability are as follows.

For the first 15 units produced of Product 1, the unit profit would be approximately \$360. The unit profit would be only \$30 for any additional units of Product 1. For the first 20 units produced of Product 2, the unit profit is estimated at \$240. The unit profit would be \$120 for each of the next 20 units and \$90 for any additional units. For the first 10 units of Product 3, the unit profit would be \$450. The unit profit would be \$300 for each of the next 5 units and \$180 for any additional units.

Certain limitations on the use of needed resources impose the following constraints on the production of the three products:

$$\begin{aligned} x_1 + x_2 + x_3 &\le 60 \\ 3x_1 + 2x_2 &\le 200 \\ x_1 + 2x_3 &\le 70. \end{aligned}$$

Management wants to know what values of x_1, x_2 and x_3 should be chosen to maximize the total profit.

(a) Plot the profit graph for each of the three products.

(b) Use separable programming to formulate a linear programming model for this problem.

C **(c)** Solve the model. What is the resulting recommendation to management about the values of x_1, x_2, and x_3 to use?

(d) Now suppose that there is an additional constraint that the profit from products 1 and 2 must total at least \$9,000. Use the technique presented in the "Extensions" subsection of Sec. 13.8 to add this constraint to the model formulated in part (*b*).

C **(e)** Repeat part (*c*) for the model formulated in part (*d*).

13.8-3.* The Dorwyn Company has two new products that will compete with the two new products for the Wyndor Glass Co. (described in Sec. 3.1). Using units of hundreds of dollars for the objective function, the linear programming model shown below has been formulated to determine the most profitable product mix.

$$\text{Maximize} \qquad Z = 4x_1 + 6x_2,$$

subject to

$$\begin{aligned} x_1 + 3x_2 &\le 8 \\ 5x_1 + 2x_2 &\le 14 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

However, because of the strong competition from Wyndor, Dorwyn management now realizes that the company will need to make a strong marketing effort to generate substantial sales of these products. In particular, it is estimated that achieving a production and sales rate of x_1 units of Product 1 per week will require weekly marketing costs of x_1^3 hundred dollars. The corresponding marketing costs for Product 2 are estimated to be $2x_2^2$ hundred dollars. Thus, the objective function in the model should be $Z = 4x_1 + 6x_2 - x_1^3 - 2x_2^2$.

Dorwyn management now would like to use the revised model to determine the most profitable product mix.

(a) Verify that $(x_1, x_2) = (2/\sqrt{3}, \frac{3}{2})$ is an optimal solution by applying the KKT conditions.

(b) Construct tables to show the profit data for each product when the production rate is 0, 1, 2, 3.

(c) Draw a figure like Fig. 13.15*b* that plots the weekly profit points for each product when the production rate is 0, 1, 2, 3. Connect the pairs of consecutive points with (dashed) line segments.

(d) Use separable programming based on this figure to formulate an approximate linear programming model for this problem.

C **(e)** Solve the model. What does this say to Dorwyn management about which product mix to use?

13.8-4. Reconsider the production scheduling problem of the Build-Em-Fast Company described in Prob. 8.1-9. The special restriction for such a situation is that overtime should not be used in any particular period unless regular time in that period is completely used up. Explain why the logic of separable programming implies that this restriction will be satisfied automatically by any optimal solution for the transportation problem formulation of the problem.

13.8-5. The B. J. Jensen Company specializes in the production of power saws and power drills for home use. Sales are relatively stable throughout the year except for a jump upward during the Christmas season. Since the production work requires considerable work and experience, the company maintains a stable employment level and then uses overtime to increase production in November. The workers also welcome this opportunity to earn extra money for the holidays.

B. J. Jensen, Jr., the current president of the company, is overseeing the production plans being made for the upcoming November. He has obtained the following data.

	Maximum Monthly Production*		**Profit per Unit Produced**	
	Regular Time	**Overtime**	**Regular Time**	**Overtime**
Power saws	3,000	2,000	$150	$50
Power drills	5,000	3,000	$100	$75

*Assuming adequate supplies of materials from the company's vendors.

However, Mr. Jensen now has learned that, in addition to the limited number of labor hours available, two other factors will limit the production levels that can be achieved this November. One is that the company's vendor for power supply units will only be able to provide 10,000 of these units for November (2,000 more than his usual monthly shipment). Each power saw and each power drill requires one of these units. Second, the vendor who supplies a key part for the gear assemblies will only be able to provide 15,000 for November (4,000 more than for other months). Each power saw requires two of these parts and each power drill requires one.

Mr. Jensen now wants to determine how many power saws and how many power drills to produce in November to maximize the company's total profit.

(a) Draw the profit graph for each of these two products.

(b) Use separable programming to formulate a linear programming model for this problem.

C **(c)** Solve the model. What does this say about how many power saws and how many power drills to produce in November?

13.8-6. Reconsider the linearly constrained convex programming model given in Prob. 13.4-7.

(a) Use the separable programming technique presented in Sec. 13.8 to formulate an approximate linear programming model for this problem. Use $x_1 = 0, 1, 2, 3$ and $x_2 = 0, 1, 2, 3$ as the breakpoints of the piecewise linear functions.

C **(b)** Use the simplex method to solve the model formulated in part (*a*). Then reexpress this solution in terms of the *original* variables of the problem.

13.8-7. Suppose that the separable programming technique has been applied to a certain problem (the "original problem") to convert it to the following equivalent linear programming problem:

$$\text{Maximize} \quad Z = 5x_{11} + 4x_{12} + 2x_{13} + 4x_{21} + x_{22},$$

subject to

$$\begin{aligned} 3x_{11} + 3x_{12} + 3x_{13} + 2x_{21} + 2x_{22} &\le 25 \\ 2x_{11} + 2x_{12} + 2x_{13} - \ \ x_{21} - \ \ x_{22} &\le 10 \end{aligned}$$

and

$$\begin{array}{ll} 0 \le x_{11} \le 2 & 0 \le x_{21} \le 3 \\ 0 \le x_{12} \le 3 & 0 \le x_{22} \le 1. \\ 0 \le x_{13} & \end{array}$$

What was the mathematical model for the original problem? (You may define the objective function either algebraically or graphically, but express the constraints algebraically.)

13.8-8. For each of the following cases, *prove* that the key property of separable programming given in Sec. 13.8 must hold. (*Hint:* Assume that there exists an optimal solution that violates this property, and then contradict this assumption by showing that there exists a better feasible solution.)

(a) The special case of separable programming where all the $g_i(\mathbf{x})$ are linear functions.

(b) The general case of separable programming where all the functions are nonlinear functions of the designated form. [*Hint:* Think of the functional constraints as constraints on resources, where $g_{ij}(x_j)$ represents the amount of resource i used by running activity j at level x_j, and then use what the convexity assumption implies about the slopes of the approximating piecewise linear function.]

13.8-9. The MFG Company produces a certain subassembly in each of two separate plants. These subassemblies are then brought to a third nearby plant where they are used in the production of a certain product. The peak season of demand for this product is approaching, so to maintain the production rate within a desired range, it is necessary to use temporarily some overtime in making the subassemblies. The cost per subassembly on regular time (RT) and on overtime (OT) is shown in the following table for both plants, along with the maximum number of subassemblies that can be produced on RT and on OT each day.

	Unit Cost		Capacity	
	RT	OT	RT	OT
Plant 1	\$15	\$25	2,000	1,000
Plant 2	\$16	\$24	1,000	500

Let x_1 and x_2 denote the total number of subassemblies produced per day at plants 1 and 2, respectively. The objective is to maximize $Z = x_1 + x_2$, subject to the constraint that the total daily cost not exceed \$60,000. Note that the mathematical programming formulation of this problem (with x_1 and x_2 as decision variables) has the same form as the main case of the separable programming model described in Sec. 13.8, except that the separable functions appear in a constraint function rather than the objective function. However, the same approach can be used to reformulate the problem as a linear programming model where it is feasible to use OT even when the RT capacity at that plant is not fully used.

(a) Formulate this linear programming model.

(b) Explain why the logic of separable programming also applies here to guarantee that an optimal solution for the model formulated in part (*a*) never uses OT unless the RT capacity at that plant has been fully used.

13.8-10. Consider the following nonlinear programming problem (first considered in Prob. 11.3-23).

$$\text{Maximize} \quad Z = 5x_1 + x_2,$$

subject to

$$2x_1^2 + x_2 \le 13$$
$$x_1^2 + x_2 \le 9$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Show that this problem is a convex programming problem.

(b) Use the separable programming technique discussed at the end of Sec. 13.8 to formulate an approximate linear programming model for this problem. Use the integers as the breakpoints of the piecewise linear function.

C **(c)** Use the computer to solve the model formulated in part (*b*). Then reexpress this solution in terms of the *original* variables of the problem.

13.8-11. Consider the following convex programming problem:

$$\text{Maximize} \quad Z = 32x_1 - x_1^4 + 4x_2 - x_2^2,$$

subject to

$$x_1^2 + x_2^2 \le 9$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Apply the separable programming technique discussed at the end of Sec. 13.8, with $x_1 = 0, 1, 2, 3$ and $x_2 = 0, 1, 2, 3$ as the breakpoint of the piecewise linear functions, to formulate an approximate linear programming model for this problem.

C **(b)** Use the computer to solve the model formulated in part (*a*). Then reexpress this solution in terms of the *original* variables of the problem.

(c) Use the KKT conditions to determine whether the solution for the original variables obtained in part (*b*) actually is optimal for the original problem (not the approximate model).

13.8-12. Reconsider the integer nonlinear programming model given in Prob. 11.3-11.

(a) Show that the objective function is not concave.

(b) Formulate an equivalent *pure binary* integer *linear* programming model for this problem as follows. Apply the separable programming technique with the feasible integers as the breakpoints of the piecewise linear functions, so that the auxiliary variables are binary variables. Then add some linear programming constraints on these binary variables to enforce the *special restriction* of separable programming. (Note that the *key property* of separable programming does not hold for this problem because the objective function is not concave.)

C **(c)** Use the computer to solve this problem as formulated in part (*b*). Then reexpress this solution in terms of the *original* variables of the problem.

D,I **13.9-1.*** Reconsider the linearly constrained convex programming model given in Prob. 13.6-5. Starting from the initial trial solution $(x_1, x_2) = (1, 1)$, use one iteration of the Frank-Wolfe algorithm to obtain exactly the same solution you found in part (*b*) of Prob. 13.6-5, and then use a second iteration to verify that it is an optimal solution (because it is replicated exactly). Explain why exactly the same results would be obtained on these two iterations with any other initial trial solution.

D,I **13.9-2.** Reconsider the linearly constrained convex programming model given in Prob. 13.6-6. Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, use one iteration of the Frank-Wolfe algorithm to obtain exactly the same solution you found in part (*b*) of Prob. 13.6-6, and then use a second iteration to verify that it is an optimal solution (because it is replicated exactly).

D,I **13.9-3.** Reconsider the linearly constrained convex programming model given in Prob. 13.6-15. Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, use one iteration of the Frank-Wolfe algorithm to obtain exactly the same solution you found in part (*c*) of Prob. 13.6-15, and then use a second iteration to verify that it is an optimal solution (because it is replicated exactly). Explain why exactly the same results would be obtained on these two iterations with any other trial solution.

D,I **13.9-4.** Reconsider the linearly constrained convex programming model given in Prob. 13.6-16. Starting from the initial trial solution $(x_1, x_2, x_3) = (0, 0, 0)$, apply two iterations of the Frank-Wolfe algorithm.

D,I **13.9-5.** Consider the quadratic programming example presented in Sec. 13.7. Starting from the initial trial solution $(x_1, x_2) = (5, 5)$, apply seven iterations of the Frank-Wolfe algorithm.

13.9-6. Reconsider the quadratic programming model given in Prob. 13.7-4.

D,I **(a)** Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, use the Frank-Wolfe algorithm (six iterations) to solve the problem (approximately).

(b) Show graphically how the sequence of trial solutions obtained in part (*a*) can be extrapolated to obtain a closer approximation of an optimal solution. What is your resulting estimate of this solution?

D,I **13.9-7.** Reconsider the first quadratic programming variation of the Wyndor Glass Co. problem presented in Sec. 13.2 (see Fig. 13.6). Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, use three iterations of the Frank-Wolfe algorithm to obtain and verify the optimal solution.

D,I **13.9-8.** Reconsider the linearly constrained convex programming model given in Prob. 13.4-7. Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, use the Frank-Wolfe algorithm (four iterations) to solve this model (approximately).

D,I **13.9-9.** Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 3x_1x_2 + 40x_1 + 30x_2 - 4x_1^2 - x_1^4 - 3x_2^2 - x_2^4,$$

subject to

$$4x_1 + 3x_2 \le 12$$
$$x_1 + 2x_2 \le 4$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

Starting from the initial trial solution $(x_1, x_2) = (0, 0)$, apply two iterations of the Frank-Wolfe algorithm.

D,I **13.9-10.*** Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 3x_1 + 4x_2 - x_1^3 - x_2^2,$$

subject to

$$x_1 + x_2 \le 1$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Starting from the initial trial solution $(x_1, x_2) = (\frac{1}{4}, \frac{1}{4})$, apply three iterations of the Frank-Wolfe algorithm.

(b) Use the KKT conditions to check whether the solution obtained in part (*a*) is, in fact, optimal.

13.9-11. Consider the following linearly constrained convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = 4x_1 - x_1^4 + 2x_2 - x_2^2,$$

subject to

$$4x_1 + 2x_2 \le 5$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Starting from the initial trial solution $(x_1, x_2) = (\frac{1}{2}, \frac{1}{2})$, apply four iterations of the Frank-Wolfe algorithm.

(b) Show graphically how the sequence of trial solutions obtained in part (*a*) can be extrapolated to obtain a closer approximation of an optimal solution. What is your resulting estimate of this solution?

(c) Use the KKT conditions to check whether the solution you obtained in part (*b*) is, in fact, optimal. If not, use these conditions to derive the exact optimal solution.

13.10-1. Reconsider the linearly constrained convex programming model given in Prob. 13.9-10.

(a) If SUMT were to be applied to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be maximized at each iteration?

(b) Setting $r = 1$ and using $(\frac{1}{4}, \frac{1}{4})$ as the initial trial solution, manually apply one iteration of the gradient search procedure (except stop before solving for t^*) to begin maximizing the function $P(\mathbf{x}; r)$ you obtained in part (*a*).

D,C **(c)** Beginning with the same initial trial solution as in part (*b*), use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 1, 10^{-2}, 10^{-4}$.

(d) Compare the final solution obtained in part (*c*) to the true optimal solution for Prob. 13.9-10 given in the back of the book. What is the percentage error in x_1, in x_2, and in $f(\mathbf{x})$?

13.10-2. Reconsider the linearly constrained convex programming model given in Prob. 13.9-11. Follow the instructions of parts (*a*), (*b*), and (*c*) of Prob. 13.10-1 for this model, except use $(x_1, x_2) = (\frac{1}{2}, \frac{1}{2})$ as the initial trial solution and use $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

13.10-3. Reconsider the model given in Prob. 13.3-3.

(a) If SUMT were to be applied directly to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be *minimized* at each iteration?

(b) Setting $r = 100$ and using $(x_1, x_2) = (5, 5)$ as the initial trial solution, manually apply one iteration of the gradient search procedure (except stop before solving for t^*) to begin minimizing the function $P(\mathbf{x}; r)$ you obtained in part (*a*).

D,C **(c)** Beginning with the same initial trial solution as in part (*b*), use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 100, 1, 10^{-2}, 10^{-4}$. (*Hint:* The computer routine assumes that the problem has been converted to *maximization* form with the functional constraints in $\le$ form.)

13.10-4. Consider the example for applying SUMT given in Sec. 13.10.

(a) Show that $(x_1, x_2) = (1, 2)$ satisfies the KKT conditions.

(b) Display the feasible region graphically, and then plot the locus of points $x_1x_2 = 2$ to demonstrate that $(x_1, x_2) = (1, 2)$ with $f(1, 2) = 2$ is, in fact, a *global maximum.*

13.10-5.* Consider the following convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = -2x_1 - (x_2 - 3)^2,$$

subject to

$$x_1 \ge 3 \quad \text{and} \quad x_2 \ge 3.$$

(a) If SUMT were applied to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be maximized at each iteration?

(b) Derive the maximizing solution of $P(\mathbf{x}; r)$ analytically, and then give this solution for $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

D,C **(c)** Beginning with the initial trial solution $(x_1, x_2) = (4, 4)$, use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

13.10-6. Use SUMT to solve the following convex programming problem:

$$\text{Minimize} \quad f(\mathbf{x}) = \frac{(x_1 + 1)^3}{3} + x_2,$$

subject to

$$x_1 \ge 1 \quad \text{and} \quad x_2 \ge 0.$$

(a) If SUMT were applied directly to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be minimized at each iteration?

(b) Derive the minimizing solution of $P(\mathbf{x}; r)$ analytically, and then give this solution for $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

D,C **(c)** Beginning with the initial trial solution $(x_1, x_2) = (2, 1)$, use the automatic routine in your OR Courseware to apply SUMT to this problem (in maximization form) with $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

D,C **13.10-7.** Consider the following convex programming problem:

$$\text{Maximize} \quad f(\mathbf{x}) = x_1x_2 - x_1 - x_1^2 - x_2 - x_2^2,$$

subject to

$$x_2 \ge 0.$$

Beginning with the initial trial solution $(x_1, x_2) = (1, 1)$, use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 1, 10^{-2}, 10^{-4}$.

D,C **13.10-8.** Reconsider the quadratic programming model given in Prob. 13.7-4. Beginning with the initial trial solution $(x_1, x_2) = (\frac{1}{2}, \frac{1}{2})$, use the automatic routine in your OR Courseware to apply SUMT to this model with $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

D,C **13.10-9.** Reconsider the first quadratic programming variation of the Wyndor Glass Co. problem presented in Sec. 13.2 (see Fig. 13.6). Beginning with the initial trial solution $(x_1, x_2) = (2, 3)$, use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 10^2, 1, 10^{-2}, 10^{-4}$.

13.10-10. Consider the following nonconvex programming problem:

$$\text{Maximize} \quad f(x) = 1{,}000x - 400x^2 + 40x^3 - x^4,$$

subject to

$$x^2 + x \le 500$$

and

$$x \ge 0.$$

(a) Identify the feasible values for x. Obtain general expressions for the first three derivatives of $f(x)$. Use this information to help you draw a rough sketch of $f(x)$ over the feasible region for x. Without calculating their values, mark the points on your graph that correspond to *local* maxima and minima.

I **(b)** Use the one-dimensional search procedure with $\epsilon = 0.05$ to find each of the local maxima. Use your sketch from part (*a*) to identify appropriate initial bounds for each of these searches. Which of the local maxima is a global maximum?

D,C **(c)** Use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 10^3, 10^2, 10, 1$ to find each of the local maxima. Use $x = 3$ and $x = 15$ as the initial trial solutions for these searches. Which of the local maxima is a global maximum?

13.10-11. Consider the following nonconvex programming problem:

$$\text{Maximize} \qquad f(\mathbf{x}) = 3x_1x_2 - 2x_1^2 - x_2^2,$$

subject to

$$\begin{aligned} x_1^2 + 2x_2^2 &\le 4 \\ 2x_1 - x_2 &\le 3 \\ x_1x_2^2 + x_1^2x_2 &= 2 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) If SUMT were to be applied to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be maximized at each iteration?

D,C **(b)** Starting from the initial trial solution $(x_1, x_2) = (1, 1)$, use the automatic routine in your OR Courseware to apply SUMT to this problem with $r = 1, 10^{-2}, 10^{-4}$.

13.10-12. Reconsider the convex programming model with an equality constraint given in Prob. 13.6-14.

(a) If SUMT were to be applied to this model, what would be the unconstrained function $P(\mathbf{x}; r)$ to be *minimized* at each iteration?

D,C **(b)** Starting from the initial trial solution $(x_1, x_2, x_3) = (\frac{3}{2}, \frac{3}{2}, 2)$, use the automatic routine in your OR Courseware to apply SUMT to this model with $r = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$.

13.10-13. Consider the following nonconvex programming problem.

$$\text{Minimize} \qquad f(\mathbf{x}) = \sin 3x_1 + \cos 3x_2 + \sin(x_1 + x_2),$$

subject to

$$\begin{aligned} x_1^2 - 10x_2 &\ge -1 \\ 10x_1 + x_2^2 &\le 100 \end{aligned}$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) If SUMT were applied to this problem, what would be the unconstrained function $P(\mathbf{x}; r)$ to be minimized at each iteration?

(b) Describe how SUMT should be applied to attempt to obtain a global minimum. (Do not actually solve.)

13.11-1. Consider the following problem:

$$\text{Maximize} \qquad Z = 4x_1 - x_1^2 + 10x_2 - x_2^2,$$

subject to

$$x_1^2 + 4x_2^2 \le 16$$

and

$$x_1 \ge 0, \qquad x_2 \ge 0.$$

(a) Is this a convex programming problem? Answer yes or no, and then justify your answer.

(b) Can the modified simplex method be used to solve this problem? Answer yes or no, and then justify your answer (but do not actually solve.)

(c) Can the Frank-Wolfe algorithm be used to solve this problem? Answer yes or no, and then justify your answer (but do not actually solve).

(d) What are the KKT conditions for this problem? Use these conditions to determine whether $(x_1, x_2) = (1, 1)$ can be optimal.

(e) Use the separable programming technique to formulate an *approximate* linear programming model for this problem. Use the feasible integers as the breakpoints for each piecewise linear function.

C **(f)** Use the simplex method to solve the problem as formulated in part (*e*).

(g) Give the function $P(\mathbf{x}; r)$ to be maximized at each iteration when applying SUMT to this problem. (Do not actually solve.)

D,C **(h)** Use SUMT (the automatic routine in your OR Courseware) to solve the problem as formulated in part (*g*). Begin with the initial trial solution $(x_1, x_2) = (2, 1)$ and use $r = 1, 10^{-2}, 10^{-4}, 10^{-6}$.

CASE 13.1 SAVVY STOCK SELECTION

Ever since the day she took her first economics class in high school, Lydia wondered about the financial practices of her parents. They worked very hard to earn enough money to live a comfortable middle-class life, but they never made their money work for them. They simply deposited their hard-earned paychecks in savings accounts earning a nominal amount of interest. (Fortunately, there always was enough money when it came time to pay her college bills.) She promised herself that when she became an adult, she would not follow the same financially conservative practices as her parents.

And Lydia kept this promise. Every morning while getting ready for work, she watches the CNN financial reports. She plays investment games on the World Wide Web, finding portfolios that maximize her return while minimizing her risk. She reads the *Wall Street Journal* and *Financial Times* with a thirst she cannot quench.

Lydia also reads the investment advice columns of the financial magazines, and she has noticed that on average, the advice of the investment advisers turns out to be very good. Therefore, she decides to follow the advice given in the latest issue of one of the magazines. In his monthly column the editor Jonathan Taylor recommends three stocks that he believes will rise far above market average. In addition, the well-known mutual fund guru Donna Carter advocates the purchase of three more stocks that she thinks will outperform the market over the next year.

BIGBELL (ticker symbol on the stock exchange: BB), one of the nation's largest telecommunications companies, trades at a price-earnings ratio well below market average. Huge investments over the last 8 months have depressed earnings considerably. However, with their new cutting edge technology, the company is expected to significantly raise their profit margins. Taylor predicts that the stock will rise from its current price of $60 per share to $72 per share within the next year.

LOTSOFPLACE (LOP) is one of the leading hard drive manufacturers in the world. The industry recently underwent major consolidation, as fierce price wars over the last few years were followed by many competitors going bankrupt or being bought by LOTSOFPLACE and its competitors. Due to reduced competition in the hard drive market, revenues and earnings are expected to rise considerably over the next year. Taylor predicts a one-year increase of 42 percent in the stock of LOTSOFPLACE from the current price of $127 per share.

INTERNETLIFE (ILI) has survived the many ups and downs of Internet companies. With the next Internet frenzy just around the corner, Taylor expects a doubling of this company's stock price from $4 to $8 within a year.

HEALTHTOMORROW (HEAL) is a leading biotechnology company that is about to get approval for several new drugs from the Food and Drug Administration, which will help earnings to grow 20 percent over the next few years. In particular a new drug to significantly reduce the risk of heart attacks is supposed to reap huge profits. Also, due to several new great-tasting medications for children, the company has been able to build an excellent image in the media. This public relations coup will surely have positive effects for the sale of its over-the-counter medications. Carter is convinced that the stock will rise from $50 to $75 per share within a year.

QUICKY (QUI) is a fast-food chain which has been vastly expanding its network of restaurants all over the United States. Carter has followed this company closely since it went public some 15 years ago when it had only a few dozen restaurants on the west coast of the United States. Since then the company has expanded, and it now has restaurants in every state. Due to its emphasis on healthy foods, it is capturing a growing market share. Carter believes that the stock will continue to perform well above market average for an increase of 46 percent in one year from its current stock price of $150.

AUTOMOBILE ALLIANCE (AUA) is a leading car manufacturer from the Detroit area that just recently introduced two new models. These models show very strong initial sales, and therefore the company's stock is predicted to rise from \$20 to \$26 over the next year.

On the World Wide Web Lydia found data about the risk involved in the stocks of these companies. The historical variances of return of the six stocks and their covariances are shown below.

Company	BB	LOP	ILI	HEAL	QUI	AUA
Variance	0.032	0.1	0.333	0.125	0.065	0.08

Covariances	LOP	ILI	HEAL	QUI	AUA
BB	0.005	0.03	−0.031	−0.027	0.01
LOP		0.085	−0.07	−0.05	0.02
ILI			−0.11	−0.02	0.042
HEAL				0.05	−0.06
QUI					−0.02

(a) At first, Lydia wants to ignore the risk of all the investments. Given this strategy, what is her optimal investment portfolio; that is, what fraction of her money should she invest in each of the six different stocks? What is the total risk of her portfolio?

(b) Lydia decides that she doesn't want to invest more than 40 percent in any individual stock. While still ignoring risk, what is her new optimal investment portfolio? What is the total risk of her new portfolio?

(c) Now Lydia wants to take into account the risk of her investment opportunities. Identify one of the model types described in this chapter that is applicable to her problem, and then formulate a model of this kind to be used in the following parts.

(d) What fractions of her money should Lydia put into the various stocks if she decides to maximize the expected return minus beta times the risk of her investment for beta = 0.25?

(e) Lydia recently received a bonus at work, \$15,000 after taxes, that she wishes to invest. For the investment policy in part (*d*), how much money does she invest in the various stocks? How many shares of each stock does she buy?

(f) How does the solution in part (*e*) change if beta = 0.5? If beta = 1? If beta = 2?

(g) Give an intuitive explanation for the change in the expected return and the risk in part (*f*) as beta changes.

(h) Lydia wants to ensure that she receives an expected return of at least 35 percent. She wants to reach this goal at minimum risk. What investment portfolio allows her to do that?

(i) What is the minimum risk Lydia can achieve if she wants an expected return of at least 25 percent? Of at least 40 percent?

(j) Do you see any problems or disadvantages with Lydia's approach to her investment strategy?

CASE 13.2 INTERNATIONAL INVESTMENTS

Charles Rosen relaxes in a plush, overstuffed recliner by the fire, enjoying the final vestiges of his week-long winter vacation. As a financial analyst working for a large investment firm in Germany, Charles has very few occasions to enjoy these private moments, since he is generally catching red-eye flights around the world to evaluate various investment opportunities. Charles pats the loyal golden retriever lying at his feet and takes a swig of brandy, enjoying the warmth of the liquid. He sighs and realizes that he must begin attending to his own financial matters while he still has the time during the holiday. He opens a folder placed conspicuously on the top of a large stack of papers. The folder contains information about an investment Charles made when he graduated from college four years ago. . . .

Charles remembers his graduation day fondly. He obtained a degree in business administration and was full of investment ideas that were born while he had been daydreaming in his numerous finance classes. Charles maintained a well-paying job throughout college, and he was able to save a large portion of the college fund that his parents had invested for him.

Upon graduation, Charles decided that he should transfer the college funds to a more lucrative investment opportunity. Since he had signed to work in Germany, he evaluated investment opportunities in that country. Ultimately, he decided to invest 30,000 German marks (DM) in so-called B-Bonds that would mature in 7 years. Charles purchased the bonds just 4 years ago last week (in early January of what will be called the “first year” in this discussion). He considered the bonds an excellent investment opportunity, since they offered high interest rates (see Table I) that would rise over the subsequent 7 years and because he could sell the bonds whenever he wanted after the first year. He calculated the amount that he would be paid if he sold bonds originally worth DM 100 on the last day of any of the 7 years (see Table II). The amount paid included the principal plus the interest. For example, if he sold bonds originally worth DM 100 on December 31 of the sixth year, he would be paid DM 163.51 (the principal is DM 100, and the interest is DM 63.51).

Charles did not sell any of the bonds during the first four years. Last year, however, the German federal government introduced a capital gains tax on interest income. The German government designated that the first DM 6,100 a single individual earns in interest per year would be tax-free. Any interest income beyond DM 6,100 would

TABLE 1 Interest rates over the 7 years

Year	Interest Rate	Annual Percentage Yield
1	7.50%	7.50%
2	8.50%	8.00%
3	8.50%	8.17%
4	8.75%	8.31%
5	9.00%	8.45%
6	9.00%	8.54%
7	9.00%	8.61%

TABLE II Total return on 100 DM

Year	DM
1	107.50
2	116.64
3	126.55
4	137.62
5	150.01
6	163.51
7	178.23

be taxed at a rate of 30 percent. For example, if Charles earned interest income of DM 10,100, he would be required to pay 30 percent of DM 4,000 (DM 10,100 − DM 6,100) in taxes, or DM 1,200. His after-tax income would therefore be DM 8,900.

Because of the new tax implemented last year, Charles has decided to reevaluate the investment. He knows that the new tax affects his potential return on the B-Bonds, but he also knows that most likely a strategy exists for maximizing his return on the bonds. He might be able to decrease the tax he has to pay on interest income by selling portions of his bonds in different years. Charles considers his strategy viable because the government requires investors to pay taxes on interest income only when they sell their B-Bonds. For example, if Charles were to sell one-third of his B-Bonds on December 31 of the sixth year, he would have to pay taxes on the interest income of DM 251 (DM 6,351 − DM 6,100).

Charles asks himself several questions. Should he keep all the bonds until the end of the seventh year? If so, he would earn 0.7823 times DM 30,000 in interest income, but he would have to pay very substantial taxes for that year. Considering these tax payments, Charles wonders if he should sell a portion of the bonds at the end of this year (the fifth year) and at the end of next year.

If Charles sells his bonds, his alternative investment opportunities are limited. He could purchase a certificate of deposit (CD) paying 4.0 percent interest, so he investigates this alternative. He meets with an investment adviser from the local branch of a bank, and the adviser tells him to keep the B-Bonds until the end of the seventh year. She argues that even if he had to pay 30 percent in taxes on the 9.00 percent rate of interest the B-Bonds would be paying in their last year (see Table I), this strategy would still result in a net rate of 6.30 percent interest, which is much better than the 4.0 percent interest he could obtain on a CD.

Charles concludes that he would make all his transactions on December 31, regardless of the year. Also, since he intends to attend business school in the United States in the fall of the seventh year and plans to pay his tuition for his second, third, and fourth semester with his investment, he does not plan to keep his money in Germany beyond December 31 of the seventh year.

(For the first three parts, assume that if Charles sells a portion of his bonds, he will put the money under his mattress earning zero percent interest. For the subsequent parts, assume that he could invest the proceeds of the bonds in the certificate of deposit.)

(a) Identify one of the model types described in this chapter that is applicable to this problem, and then formulate a model of this kind to be used in the following parts.
(b) What is the optimal investment strategy for Charles?
(c) What is fundamentally wrong with the advice Charles got from the investment adviser at the bank?
(d) Now that Charles is considering investment in the certificate of deposit, what is his optimal investment strategy?
(e) What would his optimal investment strategy for the fifth, sixth, and seventh years have been if he had originally invested DM 50,000?
(f) Charles and his fiancée have been planning to get married after his first year in business school. However, Charles learns that for married couples, the tax-free amount of interest earnings each year is DM 12,200. How much money could Charles save on his DM 30,000 investment by getting married this year (the fifth year for his investment)?
(g) Due to a recession in Germany, interest rates are low and are expected to remain low. However, since the American economy is booming, interest rates are expected to rise in the United States. A rise in interest rates would lead to a rise of the dollar in comparison to the mark. Analysts at Charles' investment bank expect the dollar to remain at the current exchange rate of DM 1.50 per dollar for the fifth year and then to rise to DM 1.80 per dollar by the end of the seventh year. Therefore, Charles is considering investing at the beginning of the sixth year in a 2-year American municipal bond paying 3.6 percent tax-exempt interest to help pay tuition. How much money should he plan to convert into dollars by selling B-Bonds for this investment?