TABLE 8.4 Equations and calculations for the initial leaving basic variable in the example for the upper bound technique

Initial Set of Equations	Maximum Feasible Value of x_1
(0) $Z - 2x_1 = 20$	$x_1 \le 4$ (since $u_1 = 4$)
(1) $4x_1 + x_2 = 12$	$x_1 \le \frac{12}{4} = 3$
(2) $-2x_1 + x_3 = 4$	$x_1 \le \frac{6-4}{2} = 1 \leftarrow$ minimum (because $u_3 = 6$)

Because Eq. (2) has the *smallest* maximum feasible value of x_1 in Table 8.4, the basic variable in this equation (x_3) provides the *leaving* basic variable. However, because x_3 reached its *upper* bound, replace x_3 by $6 - y_3$, so that $y_3 = 0$ becomes the new nonbasic variable for the next BF solution and x_1 becomes the new basic variable in Eq. (2). This replacement leads to the following changes in this equation:

$$\begin{aligned}
(2) \qquad & -2x_1 + x_3 = 4 \\
\rightarrow\ & -2x_1 + 6 - y_3 = 4 \\
\rightarrow\ & -2x_1 - y_3 = -2 \\
\rightarrow\ & x_1 + \frac{1}{2}y_3 = 1
\end{aligned}$$

Therefore, after we eliminate x_1 algebraically from the other equations, the *second* complete set of equations becomes

$$\begin{aligned}
(0) \qquad & Z + y_3 = 22 \\
(1) \qquad & x_2 - 2y_3 = 8 \\
(2) \qquad & x_1 + \frac{1}{2}y_3 = 1.
\end{aligned}$$

The resulting BF solution is $x_1 = 1$, $x_2 = 8$, $y_3 = 0$. By the optimality test, it also is an optimal solution, so $x_1 = 1$, $x_2 = 8$, $x_3 = 6 - y_3 = 6$ is the desired solution for the original problem.

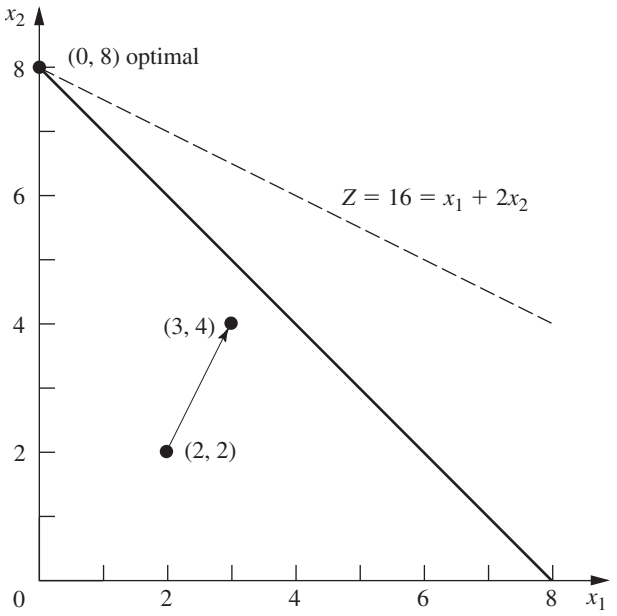
If you would like to see **another example** of the upper bound technique, the Solved Examples section of the book's website includes one.

8.4 AN INTERIOR-POINT ALGORITHM

In Sec. 4.9 we discussed a dramatic development in linear programming that occurred in 1984, namely, the invention by Narendra Karmarkar of AT&T Bell Laboratories of a powerful algorithm for solving huge linear programming problems with an approach very different from the simplex method. We now introduce the nature of Karmarkar's approach by describing a relatively elementary variant (the "affine" or "affine-scaling" variant) of his algorithm.[6] (Your IOR Tutorial also includes this variant under the title, *Solve Automatically by the Interior-Point Algorithm.*)

Throughout this section we shall focus on Karmarkar's main ideas on an intuitive level while avoiding mathematical details. In particular, we shall bypass certain details

[6]The basic approach for this variant actually was proposed in 1967 by a Russian mathematician I. I. Dikin and then rediscovered soon after the appearance of Karmarkar's work by a number of researchers, including E. R. Barnes, T. M. Cavalier, and A. L. Soyster. Also see R. J. Vanderbei, M. S. Meketon, and B. A. Freedman, "A Modification of Karmarkar's Linear Programming Algorithm," *Algorithmica,* **1**(4) (Special Issue on New Approaches to Linear Programming): 395–407, 1986.



■ **FIGURE 8.3**
Example for the interior-point algorithm.

that are needed for the full implementation of the algorithm (e.g., how to find an initial feasible trial solution) but are not central to a basic conceptual understanding. The ideas to be described can be summarized as follows:

Concept 1: Shoot through the *interior* of the feasible region toward an optimal solution.
Concept 2: Move in a direction that improves the objective function value at the fastest possible rate.
Concept 3: Transform the feasible region to place the current trial solution near its center, thereby enabling a large improvement when concept 2 is implemented.

To illustrate these ideas throughout the section, we shall use the following example:

Maximize $Z = x_1 + 2x_2$,

subject to

$$x_1 + x_2 \leq 8$$

and

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

This problem is depicted graphically in Fig. 8.3, where the optimal solution is seen to be $(x_1, x_2) = (0, 8)$ with $Z = 16$. (We will describe the significance of the arrow in the figure shortly.)

You will see that our interior-point algorithm requires a considerable amount of work to solve this tiny example. The reason is that the algorithm is designed to solve *huge* problems efficiently, but is much less efficient than the simplex method (or the graphical method in this case) for small problems. However, having an example with only two variables will allow us to depict graphically what the algorithm is doing.

The Relevance of the Gradient for Concepts 1 and 2

The algorithm begins with an initial trial solution that (like all subsequent trial solutions) lies in the *interior* of the feasible region, i.e., *inside the boundary* of the feasible region. Thus, for the example, the solution must not lie on any of the three lines ($x_1 = 0$, $x_2 = 0$,

$x_1 + x_2 = 8$) that form the boundary of this region in Fig. 8.3. (A trial solution that lies on the boundary cannot be used because this would lead to the undefined mathematical operation of division by zero at one point in the algorithm.) We have arbitrarily chosen $(x_1, x_2) = (2, 2)$ to be the initial trial solution.

To begin implementing concepts 1 and 2, note in Fig. 8.3 that the direction of movement from (2, 2) that increases Z at the fastest possible rate is *perpendicular* to (and toward) the objective function line $Z = 16 = x_1 + 2x_2$. We have shown this direction by the arrow from (2, 2) to (3, 4). Using vector addition, we have

$$(3, 4) = (2, 2) + (1, 2),$$

where the vector (1, 2) is the **gradient** of the objective function. (We will discuss gradients further in Sec. 13.5 in the broader context of *nonlinear programming,* where algorithms similar to Karmarkar's have long been used.) The components of (1, 2) are just the coefficients in the objective function. Thus, with one subsequent modification, the gradient (1, 2) defines the ideal direction to which to move, where the question of the *distance to move* will be considered later.

The algorithm actually operates on linear programming problems after they have been rewritten in augmented form. Letting x_3 be the slack variable for the functional constraint of the example, we see that this form is

$$\text{Maximize} \qquad Z = x_1 + 2x_2,$$

subject to

$$x_1 + x_2 + x_3 = 8$$

and

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

In matrix notation (slightly different from Chap. 5 because the slack variable now is incorporated into the notation), the augmented form can be written in general as

$$\text{Maximize} \qquad Z = \mathbf{c}^T\mathbf{x},$$

subject to

$$\mathbf{Ax} = \mathbf{b}$$

and

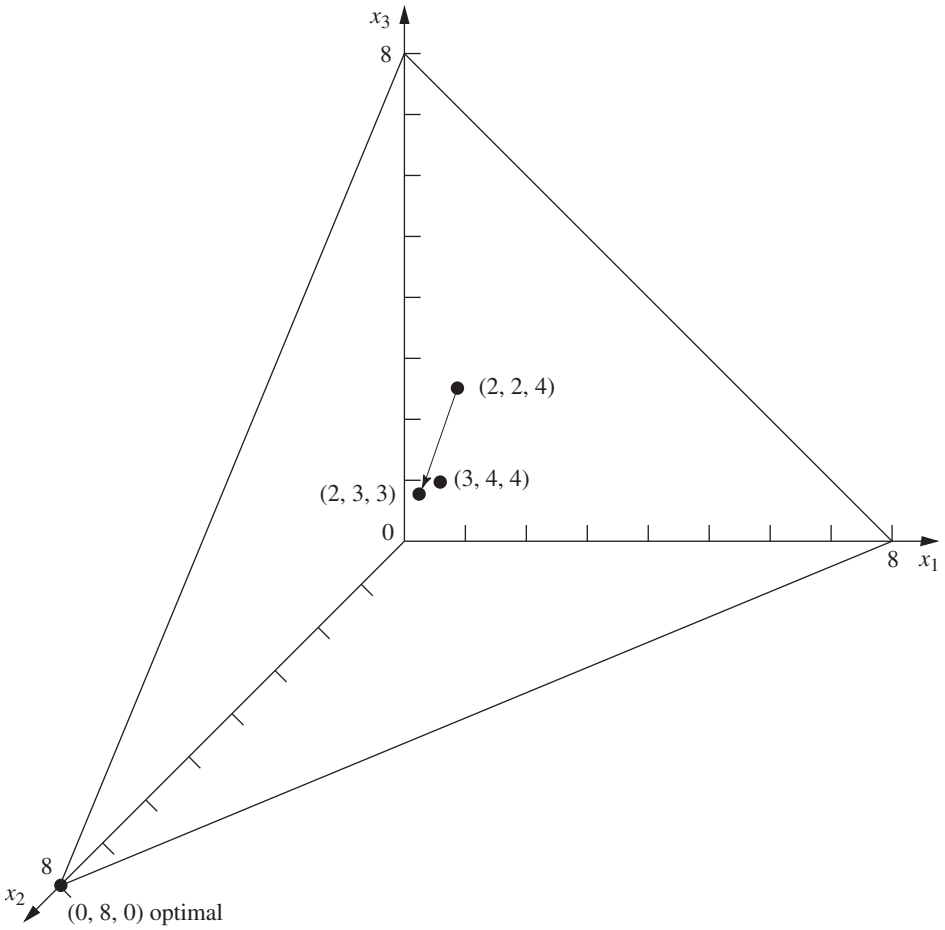
$$\mathbf{x} \geq \mathbf{0},$$

where

$$\mathbf{c} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \qquad \mathbf{A} = [1, \quad 1, \quad 1], \qquad \mathbf{b} = [8], \qquad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

for the example. Note that $\mathbf{c}^T = [1, 2, 0]$ now is the gradient of the objective function.

The augmented form of the example is depicted graphically in Fig. 8.4. The feasible region now consists of the triangle with vertices (8, 0, 0), (0, 8, 0), and (0, 0, 8). Points in the interior of this feasible region are those where $x_1 > 0$, $x_2 > 0$, and $x_3 > 0$. Each of these three $x_j > 0$ conditions has the effect of forcing (x_1, x_2) away from one of the three lines forming the boundary of the feasible region in Fig. 8.3.



■ **FIGURE 8.4**
Example in augmented form for the interior-point algorithm.

Using the Projected Gradient to Implement Concepts 1 and 2

In augmented form, the initial trial solution for the example is $(x_1, x_2, x_3) = (2, 2, 4)$. Adding the gradient (1, 2, 0) leads to

$$(3, 4, 4) = (2, 2, 4) + (1, 2, 0).$$

However, now there is a complication. The algorithm cannot move from (2, 2, 4) to (3, 4, 4), because (3, 4, 4) is infeasible! When $x_1 = 3$ and $x_2 = 4$, then $x_3 = 8 - x_1 - x_2 = 1$ instead of 4. The point (3, 4, 4) lies on the near side as you look down on the feasible triangle in Fig. 8.4. Therefore, to remain feasible, the algorithm (indirectly) *projects* the point (3, 4, 4) down onto the feasible triangle by dropping a line that is *perpendicular* to this triangle. A vector from (0, 0, 0) to (1, 1, 1) is perpendicular to this triangle, so the perpendicular line through (3, 4, 4) is given by the equation

$$(x_1, x_2, x_3) = (3, 4, 4) - \theta(1, 1, 1),$$

where θ is a scalar. Since the triangle satisfies the equation $x_1 + x_2 + x_3 = 8$, this perpendicular line intersects the triangle at (2, 3, 3). Because

$$(2, 3, 3) = (2, 2, 4) + (0, 1, -1),$$

the **projected gradient** of the objective function (the gradient projected onto the feasible region) is (0, 1, −1). It is this projected gradient that defines the direction of movement from (2, 2, 4) for the algorithm, as shown by the arrow in Fig. 8.4.

A formula is available for computing the projected gradient directly. By defining the *projection matrix* $\mathbf{P}$ as

$$\mathbf{P} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A},$$

the *projected gradient* (in column form) is

$$\mathbf{c}_p = \mathbf{P}\mathbf{c}.$$

Thus, for the example,

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \left([1 \quad 1 \quad 1] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right)^{-1} [1 \quad 1 \quad 1]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1 \quad 1 \quad 1]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix},$$

so

$$\mathbf{c}_p = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}.$$

Moving from (2, 2, 4) in the direction of the projected gradient (0, 1, −1) involves increasing α from zero in the formula

$$\mathbf{x} = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} + 4\alpha\mathbf{c}_p = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} + 4\alpha \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix},$$

where the coefficient 4 is used simply to give an upper bound of 1 for α to maintain feasibility (all $x_j \geq 0$). Note that increasing α to $\alpha = 1$ would cause x_3 to decrease to $x_3 = 4 + 4(1)(-1) = 0$, where $\alpha > 1$ yields $x_3 < 0$. Thus, α measures the fraction used of the distance that could be moved before the feasible region is left.

How large should α be made for moving to the next trial solution? Because the increase in Z is proportional to α, a value close to the upper bound of 1 is good for giving a relatively large step toward optimality on the current iteration. However, the problem with a value too close to 1 is that the next trial solution then is jammed against a constraint boundary, thereby making it difficult to take large improving steps during subsequent iterations. Therefore, it is very helpful for trial solutions to be near the center of the feasible region (or at least near the center of the portion of the feasible region in the vicinity of an optimal solution), and not too close to any constraint boundary. With this in mind, Karmarkar has stated for his algorithm that a value as large as $\alpha = 0.25$ should be "safe." In practice, much larger values (for example, $\alpha = 0.9$) sometimes are used. For the purposes of this example (and the problems at the end of the chapter), we have chosen $\alpha = 0.5$. (Your IOR Tutorial uses $\alpha = 0.5$ as the default value, but also has $\alpha = 0.9$ available.)

A Centering Scheme for Implementing Concept 3

We now have just one more step to complete the description of the algorithm, namely, a special scheme for transforming the feasible region to place the current trial solution near its center. We have just described the benefit of having the trial solution near the center, but another important benefit of this centering scheme is that it keeps turning the direction of the projected gradient to point more nearly toward an optimal solution as the algorithm converges toward this solution.

The basic idea of the centering scheme is straightforward—simply change the scale (units) for each of the variables so that the trial solution becomes equidistant from the constraint boundaries in the new coordinate system. (Karmarkar's original algorithm uses a more sophisticated centering scheme.)

For the example, there are three constraint boundaries in Fig. 8.3, each one corresponding to a zero value for one of the three variables of the problem in augmented form, namely, $x_1 = 0$, $x_2 = 0$, and $x_3 = 0$. In Fig. 8.4, see how these three constraint boundaries intersect the $\mathbf{Ax} = \mathbf{b}$ ($x_1 + x_2 + x_3 = 8$) plane to form the boundary of the feasible region. The initial trial solution is $(x_1, x_2, x_3) = (2, 2, 4)$, so this solution is 2 units away from the $x_1 = 0$ and $x_2 = 0$ constraint boundaries and 4 units away from the $x_3 = 0$ constraint boundary, when the units of the respective variables are used. However, whatever these units are in each case, they are quite arbitrary and can be changed as desired without changing the problem. Therefore, let us rescale the variables as follows:

$$\tilde{x}_1 = \frac{x_1}{2}, \qquad \tilde{x}_2 = \frac{x_2}{2}, \qquad \tilde{x}_3 = \frac{x_3}{4}$$

in order to make the current trial solution of $(x_1, x_2, x_3) = (2, 2, 4)$ become

$$(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (1, 1, 1).$$

In these new coordinates (substituting $2\tilde{x}_1$ for x_1, $2\tilde{x}_2$ for x_2, and $4\tilde{x}_3$ for x_3), the problem becomes

$$\text{Maximize} \qquad Z = 2\tilde{x}_1 + 4\tilde{x}_2,$$

subject to

$$2\tilde{x}_1 + 2\tilde{x}_2 + 4\tilde{x}_3 = 8$$

and

$$\tilde{x}_1 \geq 0, \qquad \tilde{x}_2 \geq 0, \qquad \tilde{x}_3 \geq 0,$$

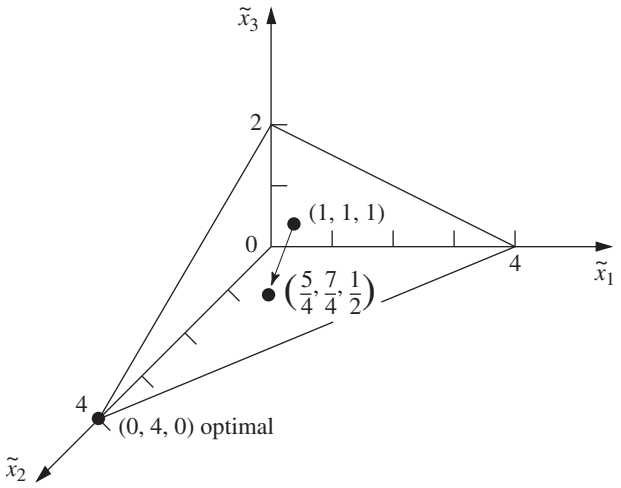
as depicted graphically in Fig. 8.5.

Note that the trial solution (1, 1, 1) in Fig. 8.5 is equidistant from the three constraint boundaries $\tilde{x}_1 = 0$, $\tilde{x}_2 = 0$, $\tilde{x}_3 = 0$. For each subsequent iteration as well, the problem is rescaled again to achieve this same property, so that the current trial solution always is (1, 1, 1) in the current coordinates.

Summary and Illustration of the Algorithm

Now let us summarize and illustrate the algorithm by going through the first iteration for the example, then giving a summary of the general procedure, and finally applying this summary to a second iteration.

Iteration 1. Given the initial trial solution $(x_1, x_2, x_3) = (2, 2, 4)$, let $\mathbf{D}$ be the corresponding *diagonal matrix* such that $\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}}$, so that



■ **FIGURE 8.5**
Example after rescaling for iteration 1.

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

The rescaled variables then are the components of

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\mathbf{x} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{x_1}{2} \\ \frac{x_2}{2} \\ \frac{x_3}{4} \end{bmatrix}.$$

In these new coordinates, $\mathbf{A}$ and $\mathbf{c}$ have become

$$\tilde{\mathbf{A}} = \mathbf{AD} = [1 \quad 1 \quad 1] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} = [2 \quad 2 \quad 4],$$

$$\tilde{\mathbf{c}} = \mathbf{Dc} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}.$$

Therefore, the projection matrix is

$$\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}\tilde{\mathbf{A}}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} \left([2 \quad 2 \quad 4] \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} \right)^{-1} [2 \quad 2 \quad 4]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{24}\begin{bmatrix} 4 & 4 & 8 \\ 4 & 4 & 8 \\ 8 & 8 & 16 \end{bmatrix} = \begin{bmatrix} \frac{5}{6} & -\frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

so that the projected gradient is

$$\mathbf{c}_p = \mathbf{P}\tilde{\mathbf{c}} = \begin{bmatrix} \frac{5}{6} & -\frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \end{bmatrix}\begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ -2 \end{bmatrix}.$$

Define v as the *absolute value* of the *negative* component of $\mathbf{c}_p$ having the *largest* absolute value, so that $v = |-2| = 2$ in this case. Consequently, in the current coordinates, the algorithm now moves from the current trial solution $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (1, 1, 1)$ to the next trial solution

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{\alpha}{v}\mathbf{c}_p = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{0.5}{2}\begin{bmatrix} 1 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} \frac{5}{4} \\ \frac{7}{4} \\ \frac{1}{2} \end{bmatrix},$$

as shown in Fig. 8.5. (The definition of v has been chosen to make the smallest component of $\tilde{\mathbf{x}}$ equal to zero when $\alpha = 1$ in this equation for the next trial solution.) In the original coordinates, this solution is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{D}\tilde{\mathbf{x}} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}\begin{bmatrix} \frac{5}{4} \\ \frac{7}{4} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{2} \\ \frac{7}{2} \\ 2 \end{bmatrix}.$$

This completes the iteration, and this new solution will be used to start the next iteration.

These steps can be summarized as follows for any iteration.

Summary of the Interior-Point Algorithm

1. Given the current trial solution $(x_1, x_2, \ldots, x_n)$, set

$$\mathbf{D} = \begin{bmatrix} x_1 & 0 & 0 & \cdots & 0 \\ 0 & x_2 & 0 & \cdots & 0 \\ 0 & 0 & x_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & x_n \end{bmatrix}$$

2. Calculate $\tilde{\mathbf{A}} = \mathbf{AD}$ and $\tilde{\mathbf{c}} = \mathbf{Dc}$.

3. Calculate $\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}\tilde{\mathbf{A}}$ and $\mathbf{c}_p = \mathbf{P}\tilde{\mathbf{c}}$.

4. Identify the negative component of $\mathbf{c}_p$ having the largest absolute value, and set v equal to this absolute value. Then calculate

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \frac{\alpha}{v}\mathbf{c}_p,$$

where α is a selected constant between 0 and 1 (for example, $\alpha = 0.5$).

5. Calculate $\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}}$ as the trial solution for the next iteration (step 1). (If this trial solution is virtually unchanged from the preceding one, then the algorithm has virtually converged to an optimal solution, so stop.)

Now let us apply this summary to iteration 2 for the example.

Iteration 2

Step 1:

Given the current trial solution $(x_1, x_2, x_3) = (\frac{5}{2}, \frac{7}{2}, 2)$, set

$$\mathbf{D} = \begin{bmatrix} \frac{5}{2} & 0 & 0 \\ 0 & \frac{7}{2} & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

(Note that the rescaled variables are

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix} = \mathbf{D}^{-1}\mathbf{x} = \begin{bmatrix} \frac{2}{5} & 0 & 0 \\ 0 & \frac{2}{7} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{5}x_1 \\ \frac{2}{7}x_2 \\ \frac{1}{2}x_3 \end{bmatrix},$$

so that the BF solutions in these new coordinates are

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{16}{5} \\ 0 \\ 0 \end{bmatrix}, \qquad \tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 0 \\ 8 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{16}{7} \\ 0 \end{bmatrix},$$

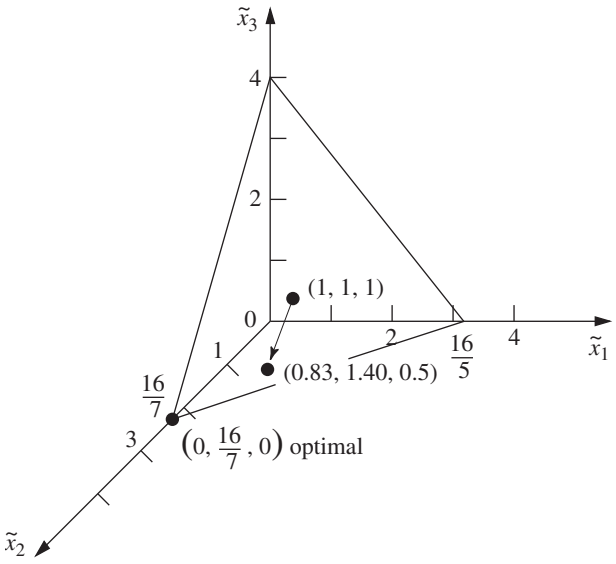
and

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 0 \\ 0 \\ 8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix},$$

as depicted in Fig. 8.6.)

Step 2:

$$\tilde{\mathbf{A}} = \mathbf{AD} = [\tfrac{5}{2}, \tfrac{7}{2}, 2] \qquad \text{and} \qquad \tilde{\mathbf{c}} = \mathbf{Dc} = \begin{bmatrix} \frac{5}{2} \\ 7 \\ 0 \end{bmatrix}.$$



■ **FIGURE 8.6**
Example after rescaling for iteration 2.

Step 3:

$$\mathbf{P} = \begin{bmatrix} \frac{13}{18} & -\frac{7}{18} & -\frac{2}{9} \\ -\frac{7}{18} & \frac{41}{90} & -\frac{14}{45} \\ -\frac{2}{9} & -\frac{14}{45} & \frac{37}{45} \end{bmatrix} \quad \text{and} \quad \mathbf{c}_p = \begin{bmatrix} -\frac{11}{12} \\ \frac{133}{60} \\ -\frac{41}{15} \end{bmatrix}.$$

Step 4:

$\left|-\frac{41}{15}\right| > \left|-\frac{11}{12}\right|$, so $v = \frac{41}{15}$ and

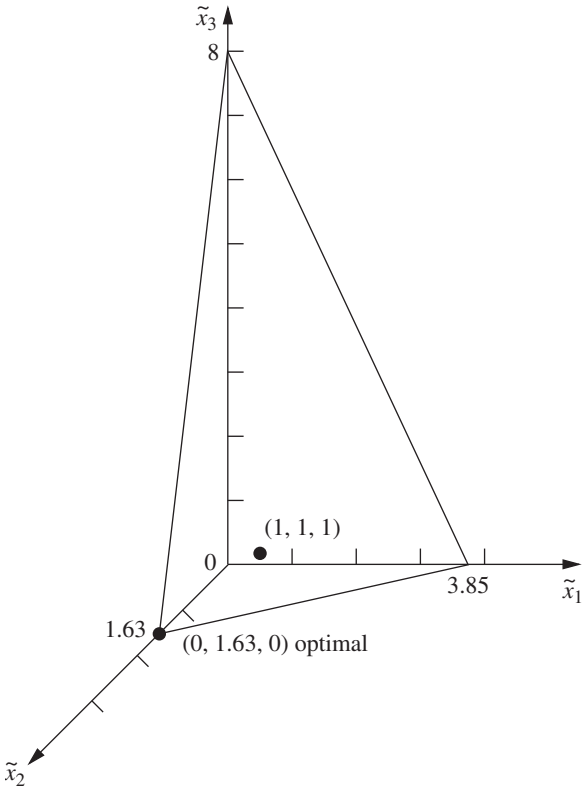
$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{0.5}{\frac{41}{15}} \begin{bmatrix} -\frac{11}{12} \\ \frac{133}{60} \\ -\frac{41}{15} \end{bmatrix} = \begin{bmatrix} \frac{273}{328} \\ \frac{461}{328} \\ \frac{1}{2} \end{bmatrix} \approx \begin{bmatrix} 0.83 \\ 1.40 \\ 0.50 \end{bmatrix}.$$

Step 5:

$$\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}} = \begin{bmatrix} \frac{1365}{656} \\ \frac{3227}{656} \\ 1 \end{bmatrix} \approx \begin{bmatrix} 2.08 \\ 4.92 \\ 1.00 \end{bmatrix}$$

is the trial solution for iteration 3.

Since there is little to be learned by repeating these calculations for additional iterations, we shall stop here. However, we do show in Fig. 8.7 the reconfigured feasible region after rescaling based on the trial solution just obtained for iteration 3. As always,



■ **FIGURE 8.7**
Example after rescaling for iteration 3.

the rescaling has placed the trial solution at $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (1, 1, 1)$, equidistant from the $\tilde{x}_1 = 0$, $\tilde{x}_2 = 0$, and $\tilde{x}_3 = 0$ constraint boundaries. Note in Figs. 8.5, 8.6, and 8.7 how the sequence of iterations and rescaling have the effect of "sliding" the optimal solution toward (1, 1, 1) while the other BF solutions tend to slide away. Eventually, after enough iterations, the optimal solution will lie very near $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (0, 1, 0)$ after rescaling, while the other two BF solutions will be *very* far from the origin on the $\tilde{x}_1$ and $\tilde{x}_3$ axes. Step 5 of that iteration then will yield a solution in the original coordinates very near the optimal solution of $(x_1, x_2, x_3) = (0, 8, 0)$.

Figure 8.8 shows the progress of the algorithm in the original $x_1 = x_2$ coordinate system before the problem is augmented. The three points—$(x_1, x_2) = (2, 2)$, (2.5, 3.5), and (2.08, 4.92)—are the trial solutions for initiating iterations 1, 2, and 3, respectively. We then have drawn a smooth curve through and beyond these points to show the trajectory of the algorithm in subsequent iterations as it approaches $(x_1, x_2) = (0, 8)$.

The functional constraint for this particular example happened to be an inequality constraint. However, equality constraints cause no difficulty for the algorithm, since it deals with the constraints only after any necessary augmenting has been done to convert them to equality form ($\mathbf{Ax} = \mathbf{b}$) anyway. To illustrate, suppose that the only change in the example is that the constraint $x_1 + x_2 \leq 8$ is changed to $x_1 + x_2 = 8$. Thus, the feasible region in Fig. 8.3 changes to just the line segment between (8, 0) and (0, 8). Given an initial feasible trial solution in the interior ($x_1 > 0$ and $x_2 > 0$) of this line segment—say, $(x_1, x_2) = (4, 4)$—the algorithm can proceed just as presented in the five-step summary with just the two variables and $\mathbf{A} = [1, 1]$. For each iteration, the projected gradient points along this line segment in the direction of (0, 8). With $\alpha = \frac{1}{2}$, iteration 1 leads from (4, 4) to (2, 6), iteration 2 leads from (2, 6) to (1, 7), etc. (Problem 8.4-3 asks you to verify these results.)

Although either version of the example has only one functional constraint, having more than one leads to just one change in the procedure as already illustrated (other than more extensive calculations). Having a single functional constraint in the example meant that **A**

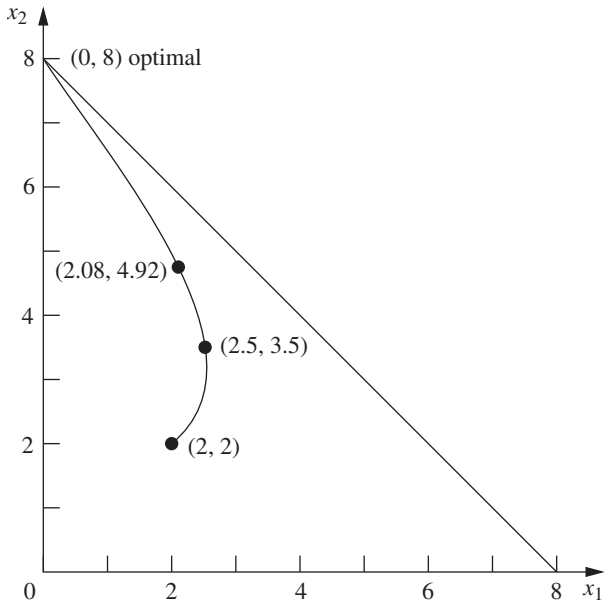


FIGURE 8.8
Trajectory of the interior-point algorithm for the example in the original x_1-x_2 coordinate system.

had only a single row, so the $(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}$ term in step 3 only involved taking the reciprocal of the number obtained from the vector product $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T$. Multiple functional constraints mean that $\mathbf{A}$ has multiple rows, so then the $(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}$ term involves finding the *inverse* of the matrix obtained from the matrix product $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T$.

To conclude, we need to add a comment to place the algorithm into better perspective. For our extremely small example, the algorithm requires relatively extensive calculations and then, after many iterations, obtains only an approximation of the optimal solution. By contrast, the graphical procedure of Sec. 3.1 finds the optimal solution in Fig. 8.3 immediately, and the simplex method requires only one quick iteration. However, do not let this contrast fool you into downgrading the efficiency of the interior-point algorithm. This algorithm is designed for dealing with *big* problems that may have many thousands of functional constraints. The simplex method typically requires thousands of iterations on such problems. By "shooting" through the interior of the feasible region, the interior-point algorithm tends to require a substantially smaller number of iterations (although with considerably more work per iteration). This sometimes enables an interior-point algorithm to efficiently solve huge linear programming problems that might even be beyond the reach of either the simplex method or the dual simplex method. Therefore, interior-point algorithms similar to the one presented here plays an important role in linear programming.

See Sec. 4.9 for a comparison of the interior-point approach with the simplex method. Section 4.9 also discusses the complementary roles of the interior-point approach and the simplex method, including how they can even be combined into a hybrid algorithm.

Finally, we should emphasize that this section has provided only a conceptual introduction to the interior-point approach to linear programming by describing a relatively elementary variant of Karmakar's path-breaking 1984 algorithm. Over the many subsequent years, a number of top-notch researchers have developed many key advances in the interior-point approach. The resulting interior-point algorithms now are commonly referred to as *barrier algorithms* (or barrier methods). Further coverage of this advanced topic is beyond the scope of this book. However, the interested reader can find many details in the selected references listed at the end of this chapter.

8.5 CONCLUSIONS

The *dual simplex method* and *parametric linear programming* are especially valuable for postoptimality analysis, although they also can be very useful in other contexts.

The *upper bound technique* provides a way of streamlining the simplex method for the common situation in which many or all of the variables have explicit upper bounds. It can greatly reduce the computational effort for large problems.

Mathematical-programming computer packages usually include all three of these procedures, and they are widely used. Because their basic structure is based largely upon the simplex method as presented in Chap. 4, they retain the exceptional computational efficiency possessed by the simplex method.

Various other special-purpose algorithms also have been developed to exploit the special structure of particular types of linear programming problems (such as those to be discussed in Chaps. 9 and 10). Much research continues to be done in this area.

Karmarkar's interior-point algorithm initiated another key line of research into how to solve linear programming problems. Variants of this algorithm now provide a powerful approach for efficiently solving some very large problems.