Chapter 3

The simplex method

Contents

We saw in Chapter 2, that if a linear programming problem in standard form has an optimal solution, then there exists a basic feasible solution that is optimal. The simplex method is based on this fact and searches for an optimal sclution by moving from one basic feasible solution to another, along the edges of the feasible set, always in a cost reducing direction. Eventually, a basic feasible solution is reached at which none of the available edges leads tc a cost reduction; such a basic feasible solution is optimal and the algorithm terminates. In this chapter, we provide a detailed development of the simplex method and discuss a few different implementations, including the simplex tableau and the revised simplex method. We also address some difficulties that may arise in the presence of degeneracy. We provide an interpretation of the simplex method in terms of column geometry, and we conclude with a discussion of its running time, as a function of the dimension of the problem being solved.

Throughout this chapter, we consider the standard form problem

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

and we let P be the corresponding feasible set. We assume that the dimensions of the matrix $\mathbf{A}$ are $m \times n$ and that its rows are linearly independent. We continue using our previous notation: $\mathbf{A}_i$ is the ith column of the matrix $\mathbf{A}$, and $\mathbf{a}_i'$ is its ith row.

3.1 Optimality conditions

Many optimization algorithms are structured as follows: given a feasible solution, we search its neighborhood to find a nearby feasible solution with lower cost. If no nearby feasible solution leads to a cost improvement, the algorithm terminates and we have a *locally optimal* solution. For general optimization problems, a locally optimal solution need not be (globally) optimal. Fortunately, in linear programming, local optimality implies global optimality; this is because we are minimizing a convex function over a convex set (cf. Exercise 3.1). In this section, we concentrate on the problem of searching for a direction of cost decrease in a neighborhood of a given basic feasible solution, and on the associated optimality conditions.

Suppose that we are at a point $\mathbf{x} \in P$ and that we contemplate moving away from $\mathbf{x}$, in the direction of a vector $\mathbf{d} \in \Re^n$. Clearly, we should only consider those choices of $\mathbf{d}$ that do not immediately take us outside the feasible set. This leads to the following definition, illustrated in Figure 3.1.

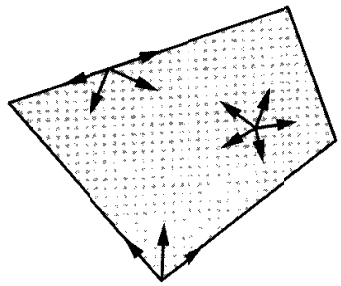

Figure 3.1: Feasible directions at different points of a polyhedron.

Definition 3.1 *Let $\mathbf{x}$ be an element of a polyhedron P. A vector $\mathbf{d} \in \Re^n$ is said to be a* **feasible direction** *at $\mathbf{x}$, if there exists a positive scalar θ for which $\mathbf{x} + \theta\mathbf{d} \in P$.*

Let $\mathbf{x}$ be a basic feasible solution to the standard form problem, let $B(1), \ldots, B(m)$ be the indices of the basic variables, and let $\mathbf{B} = [\mathbf{A}_{B(1)} \cdots \mathbf{A}_{B(m)}]$ be the corresponding basis matrix. In particular, we have $x_i = 0$ for every nonbasic variable, while the vector $\mathbf{x}_B = (x_{B(1)}, \ldots, x_{B(m)})$ of basic variables is given by

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}.$$

We consider the possibility of moving away from $\mathbf{x}$, to a new vector $\mathbf{x} + \theta\mathbf{d}$, by selecting a nonbasic variable x_j (which is initially at zero level), and increasing it to a positive value θ, while keeping the remaining nonbasic variables at zero. Algebraically, $d_j = 1$, and $d_i = 0$ for every nonbasic index i other than j. At the same time, the vector $\mathbf{x}_B$ of basic variables changes to $\mathbf{x}_B + \theta\mathbf{d}_B$, where $\mathbf{d}_B = (d_{B(1)}, d_{B(2)}, \ldots, d_{B(m)})$ is the vector with those components of $\mathbf{d}$ that correspond to the basic variables.

Given that we are only interested in feasible solutions, we require $\mathbf{A}(\mathbf{x} + \theta\mathbf{d}) = \mathbf{b}$, and since $\mathbf{x}$ is feasible, we also have $\mathbf{A}\mathbf{x} = \mathbf{b}$. Thus, for the equality constraints to be satisfied for $\theta > 0$, we need $\mathbf{A}\mathbf{d} = \mathbf{0}$. Recall now that $d_j = 1$, and that $d_i = 0$ for all other nonbasic indices i. Then,

$$\mathbf{0} = \mathbf{A}\mathbf{d} = \sum_{i=1}^{n} \mathbf{A}_i d_i = \sum_{i=1}^{m} \mathbf{A}_{B(i)} d_{B(i)} + \mathbf{A}_j = \mathbf{B}\mathbf{d}_B + \mathbf{A}_j.$$

Since the basis matrix $\mathbf{B}$ is invertible, we obtain

$$\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{A}_j. \tag{3.1}$$

The direction vector $\mathbf{d}$ that we have just constructed will be referred to as the jth *basic direction*. We have so far guaranteed that the equality constraints are respected as we move away from $\mathbf{x}$ along the basic direction $\mathbf{d}$. How about the nonnegativity constraints? We recall that the variable x_j is increased, and all other nonbasic variables stay at zero level. Thus, we need only worry about the basic variables. We distinguish two cases:

(a) Suppose that $\mathbf{x}$ is a nondegenerate basic feasible solution. Then, $\mathbf{x}_B > \mathbf{0}$, from which it follows that $\mathbf{x}_B + \theta \mathbf{d}_B \geq \mathbf{0}$, and feasibility is maintained, when θ is sufficiently small. In particular, $\mathbf{d}$ is a feasible direction.

(b) Suppose now that $\mathbf{x}$ is degenerate. Then, $\mathbf{d}$ is not always a feasible direction. Indeed, it is possible that a basic variable $x_{B(i)}$ is zero, while the corresponding component $d_{B(i)}$ of $\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{A}_j$ is negative. In that case, if we follow the jth basic direction, the nonnegativity constraint for $x_{B(i)}$ is immediately violated, and we are led to infeasible solutions; see Figure 3.2.

We now study the effects on the cost function if we move along a basic direction. If $\mathbf{d}$ is the jth basic direction, then the rate $\mathbf{c}'\mathbf{d}$ of cost change along the direction $\mathbf{d}$ is given by $\mathbf{c}'_B\mathbf{d}_B + c_j$, where $\mathbf{c}_B = (c_{B(1)}, \ldots, c_{B(m)})$. Using Eq. (3.1), this is the same as $c_j - \mathbf{c}'_B\mathbf{B}^{-1}\mathbf{A}_j$. This quantity is important enough to warrant a definition. For an intuitive interpretation, c_j is the cost per unit increase in the variable x_j, and the term $-\mathbf{c}'_B\mathbf{B}^{-1}\mathbf{A}_j$ is the cost of the compensating change in the basic variables necessitated by the constraint $\mathbf{Ax} = \mathbf{b}$.

Definition 3.2 *Let $\mathbf{x}$ be a basic solution, let $\mathbf{B}$ be an associated basis matrix, and let $\mathbf{c}_B$ be the vector of costs of the basic variables. For each j, we define the* **reduced cost** *$\bar{c}_j$ of the variable x_j according to the formula*

$$\bar{c}_j = c_j - \mathbf{c}'_B\mathbf{B}^{-1}\mathbf{A}_j.$$

Example 3.1 Consider the linear programming problem

$$\begin{array}{lrcl} \text{minimize} & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 & & \\ \text{subject to} & x_1 + x_2 + x_3 + x_4 & = & 2 \\ & 2x_1 + 3x_3 + 4x_4 & = & 2 \\ & x_1, x_2, x_3, x_4 \geq 0. & & \end{array}$$

The first two columns of the matrix $\mathbf{A}$ are $\mathbf{A}_1 = (1, 2)$ and $\mathbf{A}_2 = (1, 0)$. Since they are linearly independent, we can choose x_1 and x_2 as our basic variables. The corresponding basis matrix is

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}.$$

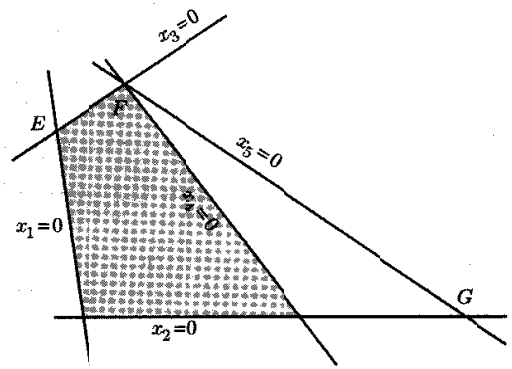

Figure 3.2: Let $n = 5, n-m = 2$. As discussed in Section 1.4, we can visualize the feasible set by standing on the two-dimensional set defined by the constraint $\mathbf{Ax} = \mathbf{b}$, in which case, the edges of the feasible set are associated with the nonnegativity constraints $x_i \geq 0$. At the nondegenerate basic feasible solution E, the variables x_1 and x_3 are at zero level (nonbasic) and x_2, x_4, x_5 are positive basic variables. The first basic direction is obtained by increasing x_1, while keeping the other nonbasic variable x_3 at zero level. This is the direction corresponding to the edge EF. Consider now the degenerate basic feasible solution F and let x_3, x_5 be the nonbasic variables. Note that x_4 is a basic variable at zero level. A basic direction is obtained by increasing x_3, while keeping the other nonbasic variable x_5 at zero level. This is the direction corresponding to the line FG and it takes us outside the feasible set. Thus, this basic direction is not a feasible direction.

We set $x_3 = x_4 = 0$, and solve for x_1, x_2, to obtain $x_1 = 1$ and $x_2 = 1$. We have thus obtained a nondegenerate basic feasible solution.

A basic direction corresponding to an increase in the nonbasic variable x_3, is constructed as follows. We have $d_3 = 1$ and $d_4 = 0$. The direction of change of the basic variables is obtained using Eq. (3.1):

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} d_{B(1)} \\ d_{B(2)} \end{bmatrix} = \mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{A}_3 = -\begin{bmatrix} 0 & 1/2 \\ 1 & -1/2 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} -3/2 \\ 1/2 \end{bmatrix}.$$

The cost of moving along this basic direction is $\mathbf{c}'\mathbf{d} = -3c_1/2 + c_2/2 + c_3$. This is the same as the reduced cost of the variable x_3.

Consider now Definition 3.2 for the case of a basic variable. Since $\mathbf{B}$ is the matrix $[\mathbf{A}_{B(1)} \cdots \mathbf{A}_{B(m)}]$, we have $\mathbf{B}^{-1}[\mathbf{A}_{B(1)} \cdots \mathbf{A}_{B(m)}] = \mathbf{I}$, where

$\mathbf{I}$ is the $m \times m$ identity matrix. In particular, $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$ is the ith column of the identity matrix, which is the ith unit vector $\mathbf{e}_i$. Therefore, for every basic variable $x_{B(i)}$, we have

$$\bar{c}_{B(i)} = c_{B(i)} - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A}_{B(i)} = c_{B(i)} - \mathbf{c}_B'\mathbf{e}_i = c_{B(i)} - c_{B(i)} = 0,$$

and we see that the reduced cost of every basic variable is zero.

Our next result provides us with optimality conditions. Given our interpretation of the reduced costs as rates of cost change along certain directions, this result is intuitive.

Theorem 3.1 *Consider a basic feasible solution $\mathbf{x}$ associated with a basis matrix $\mathbf{B}$, and let $\bar{\mathbf{c}}$ be the corresponding vector of reduced costs.*

(a) *If $\bar{\mathbf{c}} \geq \mathbf{0}$, then $\mathbf{x}$ is optimal.*

(b) *If $\mathbf{x}$ is optimal and nondegenerate, then $\bar{\mathbf{c}} \geq \mathbf{0}$.*

Proof.

(a) We assume that $\bar{\mathbf{c}} \geq 0$, we let $\mathbf{y}$ be an arbitrary feasible solution, and we define $\mathbf{d} = \mathbf{y} - \mathbf{x}$. Feasibility implies that $\mathbf{Ax} = \mathbf{Ay} = \mathbf{b}$ and, therefore, $\mathbf{Ad} = \mathbf{0}$. The latter equality can be rewritten in the form

$$\mathbf{B}\mathbf{d}_B + \sum_{i \in N} \mathbf{A}_i d_i = \mathbf{0},$$

where N is the set of indices corresponding to the nonbasic variables under the given basis. Since $\mathbf{B}$ is invertible, we obtain

$$\mathbf{d}_B = -\sum_{i \in N} \mathbf{B}^{-1}\mathbf{A}_i d_i,$$

and

$$\mathbf{c}'\mathbf{d} = \mathbf{c}_B'\mathbf{d}_B + \sum_{i \in N} c_i d_i = \sum_{i \in N} (c_i - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A}_i) d_i = \sum_{i \in N} \bar{c}_i d_i.$$

For any nonbasic index $i \in N$, we must have $x_i = 0$ and, since $\mathbf{y}$ is feasible, $y_i \geq 0$. Thus, $d_i \geq 0$ and $\bar{c}_i d_i \geq 0$, for all $i \in N$. We conclude that $\mathbf{c}'(\mathbf{y} - \mathbf{x}) = \mathbf{c}'\mathbf{d} \geq 0$, and since $\mathbf{y}$ was an arbitrary feasible solution, $\mathbf{x}$ is optimal.

(b) Suppose that $\mathbf{x}$ is a nondegenerate basic feasible solution and that $\bar{c}_j < 0$ for some j. Since the reduced cost of a basic variable is always zero, x_j must be a nonbasic variable and $\bar{c}_j$ is the rate of cost change along the jth basic direction. Since $\mathbf{x}$ is nondegenerate, the jth basic direction is a feasible direction of cost decrease, as discussed earlier. By moving in that direction, we obtain feasible solutions whose cost is less than that of $\mathbf{x}$, and $\mathbf{x}$ is not optimal. □

Note that Theorem 3.1 allows the possibility that $\mathbf{x}$ is a (degenerate) optimal basic feasible solution, but that $\bar{c}_j < 0$ for some nonbasic index j. There is an analog of Theorem 3.1 that provides conditions under which a basic feasible solution $\mathbf{x}$ is a unique optimal solution; see Exercise 3.6. A related view of the optimality conditions is developed in Exercises 3.2 and 3.3.

According to Theorem 3.1, in order to decide whether a nondegenerate basic feasible solution is optimal, we need only check whether all reduced costs are nonnegative, which is the same as examining the $n - m$ basic directions. If $\mathbf{x}$ is a degenerate basic feasible solution, an equally simple computational test for determining whether $\mathbf{x}$ is optimal is not available (see Exercises 3.7 and 3.8). Fortunately, the simplex method, as developed in subsequent sections, manages to get around this difficulty in an effective manner.

Note that in order to use Theorem 3.1 and assert that a certain basic solution is optimal, we need to satisfy two conditions: feasibility, and nonnegativity of the reduced costs. This leads us to the following definition.

Definition 3.3 *A basis matrix $\mathbf{B}$ is said to be* **optimal** *if:*

(a) $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, *and*

(b) $\bar{\mathbf{c}}' = \mathbf{c}' - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A} \geq \mathbf{0}'$.

Clearly, if an optimal basis is found, the corresponding basic solution is feasible, satisfies the optimality conditions, and is therefore optimal. On the other hand, in the degenerate case, having an optimal basic feasible solution does not necessarily mean that the reduced costs are nonnegative.

3.2 Development of the simplex method

We will now complete the development of the simplex method. Our main task is to work out the details of how to move to a better basic feasible solution, whenever a profitable basic direction is discovered.

Let us assume that every basic feasible solution is nondegenerate. This assumption will remain in effect until it is explicitly relaxed later in this section. Suppose that we are at a basic feasible solution $\mathbf{x}$ and that we have computed the reduced costs $\bar{c}_j$ of the nonbasic variables. If all of them are nonnegative, Theorem 3.1 shows that we have an optimal solution, and we stop. If on the other hand, the reduced cost $\bar{c}_j$ of a nonbasic variable x_j is negative, the jth basic direction $\mathbf{d}$ is a feasible direction of cost decrease. [This is the direction obtained by letting $d_j = 1$, $d_i = 0$ for $i \neq B(1), \ldots, B(m), j$, and $\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{A}_j$.] While moving along this direction $\mathbf{d}$, the nonbasic variable x_j becomes positive and all other nonbasic

variables remain at zero. We describe this situation by saying that x_j (or $\mathbf{A}_j$) *enters* or *is brought into the basis*.

Once we start moving away from $\mathbf{x}$ along the direction $\mathbf{d}$, we are tracing points of the form $\mathbf{x}+\theta\mathbf{d}$, where $\theta \geq 0$. Since costs decrease along the direction $\mathbf{d}$, it is desirable to move as far as possible. This takes us to the point $\mathbf{x}+\theta^*\mathbf{d}$, where

$$\theta^* = \max\{\theta \geq 0 \mid \mathbf{x}+\theta\mathbf{d} \in P\}.$$

The resulting cost change is $\theta^*\mathbf{c}'\mathbf{d}$, which is the same as $\theta^*\bar{c}_j$.

We now derive a formula for θ^*. Given that $\mathbf{Ad}=\mathbf{0}$, we have $\mathbf{A}(\mathbf{x}+\theta\mathbf{d}) = \mathbf{Ax} = \mathbf{b}$ for all θ, and the equality constraints will never be violated. Thus, $\mathbf{x}+\theta\mathbf{d}$ can become infeasible only if one of its components becomes negative. We distinguish two cases:

(a) If $\mathbf{d} \geq \mathbf{0}$, then $\mathbf{x}+\theta\mathbf{d} \geq \mathbf{0}$ for all $\theta \geq 0$, the vector $\mathbf{x}+\theta\mathbf{d}$ never becomes infeasible, and we let $\theta^* = \infty$.

(b) If $d_i < 0$ for some i, the constraint $x_i + \theta d_i \geq 0$ becomes $\theta \leq -x_i/d_i$. This constraint on θ must be satisfied for every i with $d_i < 0$. Thus, the largest possible value of θ is

$$\theta^* = \min_{\{i \mid d_i<0\}} \left(-\frac{x_i}{d_i}\right).$$

Recall that if x_i is a nonbasic variable, then either x_i is the entering variable and $d_i = 1$, or else $d_i = 0$. In either case, d_i is nonnegative. Thus, we only need to consider the basic variables and we have the equivalent formula

$$\theta^* = \min_{\{i=1,\ldots,m \mid d_{B(i)}<0\}} \left(-\frac{x_{B(i)}}{d_{B(i)}}\right). \tag{3.2}$$

Note that $\theta^* > 0$, because $x_{B(i)} > 0$ for all i, as a consequence of nondegeneracy.

Example 3.2 This is a continuation of Example 3.1 from the previous section, dealing with the linear programming problem

$$\begin{array}{lrcrcrcrcl} \text{minimize} & c_1x_1 & + & c_2x_2 & + & c_3x_3 & + & c_4x_4 & & \\ \text{subject to} & x_1 & + & x_2 & + & x_3 & + & x_4 & = & 2 \\ & 2x_1 & & & + & 3x_3 & + & 4x_4 & = & 2 \\ & x_1, x_2, x_3, x_4 & \geq & 0. & & & & & & \end{array}$$

Let us again consider the basic feasible solution $\mathbf{x} = (1,1,0,0)$ and recall that the reduced cost $\bar{c}_3$ of the nonbasic variable x_3 was found to be $-3c_1/2 + c_2/2 + c_3$. Suppose that $\mathbf{c} = (2,0,0,0)$, in which case, we have $\bar{c}_3 = -3$. Since $\bar{c}_3$ is negative, we form the corresponding basic direction, which is $\mathbf{d} = (-3/2, 1/2, 1, 0)$, and consider vectors of the form $\mathbf{x}+\theta\mathbf{d}$, with $\theta \geq 0$. As θ increases, the only component of $\mathbf{x}$ that decreases is the first one (because $d_1 < 0$). The largest possible value of θ is given by $\theta^* = -(x_1/d_1) = 2/3$. This takes us to the point $\mathbf{y} = \mathbf{x}+2\mathbf{d}/3 = (0, 4/3, 2/3, 0)$. Note that the columns $\mathbf{A}_2$ and $\mathbf{A}_3$ corresponding to the nonzero variables at the new vector $\mathbf{y}$ are $(1,0)$ and $(1,3)$, respectively, and are linearly independent. Therefore, they form a basis and the vector $\mathbf{y}$ is a new basic feasible solution. In particular, the variable x_3 has entered the basis and the variable x_1 has exited the basis.

Once θ^* is chosen, and assuming it is finite, we move to the new feasible solution $\mathbf{y} = \mathbf{x}+\theta^*\mathbf{d}$. Since $x_j = 0$ and $d_j = 1$, we have $y_j = \theta^* > 0$. Let ℓ be a minimizing index in Eq. (3.2), that is,

$$-\frac{x_{B(\ell)}}{d_{B(\ell)}} = \min_{\{i=1,\ldots,m \mid d_{B(i)}<0\}} \left(-\frac{x_{B(i)}}{d_{B(i)}}\right) = \theta^*;$$

in particular,

$$d_{B(\ell)} < 0,$$

and

$$x_{B(\ell)} + \theta^* d_{B(\ell)} = 0.$$

We observe that the basic variable $x_{B(\ell)}$ has become zero, whereas the nonbasic variable x_j has now become positive, which suggests that x_j should replace $x_{B(\ell)}$ in the basis. Accordingly, we take the old basis matrix $\mathbf{B}$ and replace $\mathbf{A}_{B(\ell)}$ with $\mathbf{A}_j$, thus obtaining the matrix

$$\overline{\mathbf{B}} = \begin{bmatrix} | & & | & | & | & & | \\ \mathbf{A}_{B(1)} & \cdots & \mathbf{A}_{B(\ell-1)} & \mathbf{A}_j & \mathbf{A}_{B(\ell+1)} & \cdots & \mathbf{A}_{B(m)} \\ | & & | & | & | & & | \end{bmatrix}. \tag{3.3}$$

Equivalently, we are replacing the set $\{B(1),\ldots,B(m)\}$ of basic indices by a new set $\{\overline{B}(1),\ldots,\overline{B}(m)\}$ of indices given by

$$\overline{B}(i) = \begin{cases} B(i), & i \neq \ell, \\ j, & i = \ell. \end{cases} \tag{3.4}$$

Theorem 3.2

(a) *The columns $\mathbf{A}_{B(i)}$, $i \neq \ell$, and $\mathbf{A}_j$ are linearly independent and, therefore, $\overline{\mathbf{B}}$ is a basis matrix.*

(b) *The vector $\mathbf{y} = \mathbf{x}+\theta^*\mathbf{d}$ is a basic feasible solution associated with the basis matrix $\overline{\mathbf{B}}$.*

Proof.

(a) If the vectors $\mathbf{A}_{\overline{B}(i)}$, $i = 1,\ldots,m$, are linearly dependent, then there exist coefficients $\lambda_1,\ldots,\lambda_m$, not all of them zero, such that

$$\sum_{i=1}^{m} \lambda_i \mathbf{A}_{\overline{B}(i)} = \mathbf{0},$$

which implies that

$$\sum_{i=1}^{n} \lambda_i \mathbf{B}^{-1}\mathbf{A}_{\overline{B}(i)} = \mathbf{0},$$

and the vectors $\mathbf{B}^{-1}\mathbf{A}_{\overline{B}(i)}$ are also linearly dependent. To show that this is not the case, we will prove that the vectors $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$, $i \neq \ell$, and $\mathbf{B}^{-1}\mathbf{A}_j$ are linearly independent. We have $\mathbf{B}^{-1}\mathbf{B} = \mathbf{I}$. Since $\mathbf{A}_{B(i)}$ is the ith column of $\mathbf{B}$, it follows that the vectors $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$, $i \neq \ell$, are all the unit vectors except for the ℓth unit vector. In particular, they are linearly independent and their ℓth component is zero. On the other hand, $\mathbf{B}^{-1}\mathbf{A}_j$ is equal to $-\mathbf{d}_B$. Its ℓth entry, $-d_{B(\ell)}$, is nonzero by the definition of ℓ. Thus, $\mathbf{B}^{-1}\mathbf{A}_j$ is linearly independent from the unit vectors $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$, $i \neq \ell$.

(b) We have $\mathbf{y} \geq \mathbf{0}$, $\mathbf{A}\mathbf{y} = \mathbf{b}$, and $y_i = 0$ for $i \neq \overline{B}(1), \ldots, \overline{B}(m)$. Furthermore, the columns $\mathbf{A}_{\overline{B}(1)}, \ldots, \mathbf{A}_{\overline{B}(m)}$ have just been shown to be linearly independent. It follows that $\mathbf{y}$ is a basic feasible solution associated with the basis matrix $\overline{\mathbf{B}}$. □

Since θ^* is positive, the new basic feasible solution $\mathbf{x} + \theta^*\mathbf{d}$ is distinct from $\mathbf{x}$; since $\mathbf{d}$ is a direction of cost decrease, the cost of this new basic feasible solution is strictly smaller. We have therefore accomplished our objective of moving to a new basic feasible solution with lower cost. We can now summarize a typical iteration of the simplex method, also known as a *pivot* (see Section 3.6 for a discussion of the origins of this term). For our purposes, it is convenient to define a vector $\mathbf{u} = (u_1, \ldots, u_m)$ by letting

$$\mathbf{u} = -\mathbf{d}_B = \mathbf{B}^{-1}\mathbf{A}_j,$$

where $\mathbf{A}_j$ is the column that enters the basis; in particular, $u_i = -d_{B(i)}$, for $i = 1, \ldots, m$.

An iteration of the simplex method

1. In a typical iteration, we start with a basis consisting of the basic columns $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(m)}$ and an associated basic feasible solution $\mathbf{x}$.
2. Compute the reduced costs $\overline{c}_j = c_j - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A}_j$ for all nonbasic indices j. If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates; else, choose some j for which $\overline{c}_j < 0$.
3. Compute $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$. If no component of $\mathbf{u}$ is positive, we have $\theta^* = \infty$, the optimal cost is $-\infty$, and the algorithm terminates.

4. If some component of $\mathbf{u}$ is positive, let

$$\theta^* = \min_{\{i=1,\ldots,m \mid u_i > 0\}} \frac{x_{B(i)}}{u_i}.$$

5. Let ℓ be such that $\theta^* = x_{B(\ell)}/u_\ell$. Form a new basis by replacing $\mathbf{A}_{B(\ell)}$ with $\mathbf{A}_j$. If $\mathbf{y}$ is the new basic feasible solution, the values of the new basic variables are $y_j = \theta^*$ and $y_{B(i)} = x_{B(i)} - \theta^* u_i$, $i \neq \ell$.

The simplex method is initialized with an arbitrary basic feasible solution, which, for feasible standard form problems, is guaranteed to exist. The following theorem states that, in the nondegenerate case, the simplex method works correctly and terminates after a finite number of iterations.

Theorem 3.3 *Assume that the feasible set is nonempty and that every basic feasible solution is nondegenerate. Then, the simplex method terminates after a finite number of iterations. At termination, there are the following two possibilities:*

(a) *We have an optimal basis $\mathbf{B}$ and an associated basic feasible solution which is optimal.*

(b) *We have found a vector $\mathbf{d}$ satisfying $\mathbf{A}\mathbf{d} = \mathbf{0}$, $\mathbf{d} \geq \mathbf{0}$, and $\mathbf{c}'\mathbf{d} < 0$, and the optimal cost is $-\infty$.*

Proof. If the algorithm terminates due to the stopping criterion in Step 2, then the optimality conditions in Theorem 3.1 have been met, $\mathbf{B}$ is an optimal basis, and the current basic feasible solution is optimal.

If the algorithm terminates because the criterion in Step 3 has been met, then we are at a basic feasible solution $\mathbf{x}$ and we have discovered a nonbasic variable x_j such that $\overline{c}_j < 0$ and such that the corresponding basic direction $\mathbf{d}$ satisfies $\mathbf{A}\mathbf{d} = \mathbf{0}$ and $\mathbf{d} \geq \mathbf{0}$. In particular, $\mathbf{x} + \theta\mathbf{d} \in P$ for all $\theta > 0$. Since $\mathbf{c}'\mathbf{d} = \overline{c}_j < 0$, by taking θ arbitrarily large, the cost can be made arbitrarily negative, and the optimal cost is $-\infty$.

At each iteration, the algorithm moves by a positive amount θ^* along a direction $\mathbf{d}$ that satisfies $\mathbf{c}'\mathbf{d} < 0$. Therefore, the cost of every successive basic feasible solution visited by the algorithm is strictly less than the cost of the previous one, and no basic feasible solution can be visited twice. Since there is a finite number of basic feasible solutions, the algorithm must eventually terminate. □

Theorem 3.3 provides an independent proof of some of the results of Chapter 2 for nondegenerate standard form problems. In particular, it shows that for feasible and nondegenerate problems, either the optimal

cost is $-\infty$, or there exists a basic feasible solution which is optimal (cf. Theorem 2.8 in Section 2.6). While the proof given here might appear more elementary, its extension to the degenerate case is not as simple.

The simplex method for degenerate problems

We have been working so far under the assumption that all basic feasible solutions are nondegenerate. Suppose now that the exact same algorithm is used in the presence of degeneracy. Then, the following new possibilities may be encountered in the course of the algorithm.

(a) If the current basic feasible solution $\mathbf{x}$ is degenerate, θ^* can be equal to zero, in which case, the new basic feasible solution $\mathbf{y}$ is the same as $\mathbf{x}$. This happens if some basic variable $x_{B(\ell)}$ is equal to zero and the corresponding component $d_{B(\ell)}$ of the direction vector $\mathbf{d}$ is negative. Nevertheless, we can still define a new basis $\overline{\mathbf{B}}$, by replacing $\mathbf{A}_{B(\ell)}$ with $\mathbf{A}_j$ [cf. Eqs. (3.3)-(3.4)], and Theorem 3.2 is still valid.

(b) Even if θ^* is positive, it may happen that more than one of the original basic variables becomes zero at the new point $\mathbf{x}+\theta^*\mathbf{d}$. Since only one of them exits the basis, the others remain in the basis at zero level, and the new basic feasible solution is degenerate.

Basis changes while staying at the same basic feasible solution are not in vain. As illustrated in Figure 3.3, a sequence of such basis changes may lead to the eventual discovery of a cost reducing feasible direction. On the other hand, a sequence of basis changes might lead back to the initial basis, in which case the algorithm may loop indefinitely. This undesirable phenomenon is called *cycling*. An example of cycling is given in Section 3.3, after we develop some bookkeeping tools for carrying out the mechanics of the algorithm. It is sometimes maintained that cycling is an exceptionally rare phenomenon. However, for many highly structured linear programming problems, most basic feasible solutions are degenerate, and cycling is a real possibility. Cycling can be avoided by judiciously choosing the variables that will enter or exit the basis (see Section 3.4). We now discuss the freedom available in this respect.

Pivot Selection

The simplex algorithm, as we described it, has certain degrees of freedom: in Step 2, we are free to choose any j whose reduced cost $\overline{c}_j$ is negative; also, in Step 5, there may be several indices ℓ that attain the minimum in the definition of θ^*, and we are free to choose any one of them. Rules for making such choices are called *pivoting rules*.

Regarding the choice of the entering column, the following rules are some natural candidates:

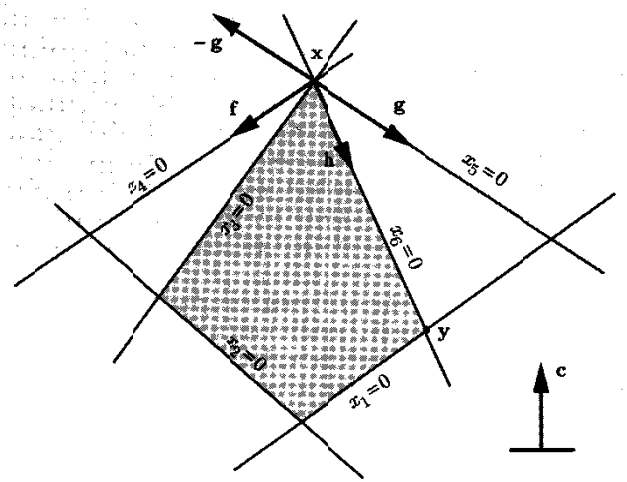

Figure 3.3: We visualize a problem in standard form, with $n-m=2$, by standing on the two-dimensional plane defined by the equality constraints $\mathbf{Ax}=\mathbf{b}$. The basic feasible solution $\mathbf{x}$ is degenerate. If x_4 and x_5 are the nonbasic variables, then the two corresponding basic directions are the vectors $\mathbf{g}$ and $\mathbf{f}$. For either of these two basic directions, we have $\theta^*=0$. However, if we perform a change of basis, with x_4 entering the basis and x_6 exiting, the new nonbasic variables are x_5 and x_6, and the two basic directions are $\mathbf{h}$ and $-\mathbf{g}$. (The direction $-\mathbf{g}$ is the one followed if x_6 is increased while x_5 is kept at zero.) In particular, we can now follow direction $\mathbf{h}$ to reach a new basic feasible solution $\mathbf{y}$ with lower cost.

(a) *Choose a column $\mathbf{A}_j$, with $\overline{c}_j<0$, whose reduced cost is the most negative.* Since the reduced cost is the rate of change of the cost function, this rule chooses a direction along which costs decrease at the fastest rate. However, the actual cost decrease depends on how far we move along the chosen direction. This suggests the next rule.

(b) *Choose a column with $\overline{c}_j<0$ for which the corresponding cost decrease $\theta^*|\overline{c}_j|$ is largest.* This rule offers the possibility of reaching optimality after a smaller number of iterations. On the other hand, the computational burden at each iteration is larger, because we need to compute θ^* for each column with $\overline{c}_j<0$. The available empirical evidence suggests that the overall running time does not improve.

For large problems, even the rule that chooses the most negative $\overline{c}_j$ can be computationally expensive, because it requires the computation of the reduced cost of every variable. In practice, simpler rules are sometimes

used, such as the *smallest subscript* rule, that chooses the smallest j for which $\bar{c}_j$ is negative. Under this rule, once a negative reduced cost is discovered, there is no reason to compute the remaining reduced costs. Other criteria that have been found to improve the overall running time are the *Devex* (Harris, 1973) and the *steepest edge* rule (Goldfarb and Reid, 1977). Finally, there are methods based on *candidate lists* whereby one examines the reduced costs of nonbasic variables by picking them one at a time from a prioritized list. There are different ways of maintaining such prioritized lists, depending on the rule used for adding, removing, or reordering elements of the list.

Regarding the choice of the exiting column, the simplest option is again the *smallest subscript* rule: out of all variables eligible to exit the basis, choose one with the smallest subscript. It turns out that by following the smallest subscript rule for both the entering and the exiting column, cycling can be avoided (cf. Section 3.4).

3.3 Implementations of the simplex method

In this section, we discuss some ways of carrying out the mechanics of the simplex method. It should be clear from the statement of the algorithm that the vectors $\mathbf{B}^{-1}\mathbf{A}_j$ play a key role. If these vectors are available, the reduced costs, the direction of motion, and the stepsize θ^* are easily computed. Thus, the main difference between alternative implementations lies in the way that the vectors $\mathbf{B}^{-1}\mathbf{A}_j$ are computed and on the amount of related information that is carried from one iteration to the next.

When comparing different implementations, it is important to keep the following facts in mind (cf. Section 1.6). If $\mathbf{B}$ is a given $m \times m$ matrix and $\mathbf{b} \in \Re^m$ is a given vector, computing the inverse of $\mathbf{B}$ or solving a linear system of the form $\mathbf{Bx} = \mathbf{b}$ takes $O(m^3)$ arithmetic operations. Computing a matrix-vector product $\mathbf{Bb}$ takes $O(m^2)$ operations. Finally, computing an inner product $\mathbf{p'b}$ of two m-dimensional vectors takes $O(m)$ arithmetic operations.

Naive implementation

We start by describing the most straightforward implementation in which no auxiliary information is carried from one iteration to the next. At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables. We form the basis matrix $\mathbf{B}$ and compute $\mathbf{p}' = \mathbf{c}'_B\mathbf{B}^{-1}$, by solving the linear system $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ for the unknown vector $\mathbf{p}$. (This vector $\mathbf{p}$ is called the vector of *simplex multipliers* associated with the basis $\mathbf{B}$.) The reduced cost $\bar{c}_j = c_j - \mathbf{c}'_B\mathbf{B}^{-1}\mathbf{A}_j$ of any variable x_j is then obtained according to the formula

$$\bar{c}_j = c_j - \mathbf{p}'\mathbf{A}_j.$$

Depending on the pivoting rule employed, we may have to compute all of the reduced costs or we may compute them one at a time until a variable with a negative reduced cost is encountered. Once a column $\mathbf{A}_j$ is selected to enter the basis, we solve the linear system $\mathbf{Bu} = \mathbf{A}_j$ in order to determine the vector $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$. At this point, we can form the direction along which we will be moving away from the current basic feasible solution. We finally determine θ^* and the variable that will exit the basis, and construct the new basic feasible solution.

We note that we need $O(m^3)$ arithmetic operations to solve the systems $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ and $\mathbf{Bu} = \mathbf{A}_j$. In addition, computing the reduced costs of all variables requires $O(mn)$ arithmetic operations, because we need to form the inner product of the vector $\mathbf{p}$ with each one of the nonbasic columns $\mathbf{A}_j$. Thus, the total computational effort per iteration is $O(m^3 + mn)$. We will see shortly that alternative implementations require only $O(m^2+mn)$ arithmetic operations. Therefore, the implementation described here is rather inefficient, in general. On the other hand, for certain problems with a special structure, the linear systems $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ and $\mathbf{Bu} = \mathbf{A}_j$ can be solved very fast, in which case this implementation can be of practical interest. We will revisit this point in Chapter 7, when we apply the simplex method to network flow problems.

Revised simplex method

Much of the computational burden in the naive implementation is due to the need for solving two linear systems of equations. In an alternative implementation, the matrix $\mathbf{B}^{-1}$ is made available at the beginning of each iteration, and the vectors $\mathbf{c}'_B\mathbf{B}^{-1}$ and $\mathbf{B}^{-1}\mathbf{A}_j$ are computed by a matrix-vector multiplication. For this approach to be practical, we need an efficient method for updating the matrix $\mathbf{B}^{-1}$ each time that we effect a change of basis. This is discussed next.

Let

$$\mathbf{B} = \left[\mathbf{A}_{B(1)} \cdots \mathbf{A}_{B(m)}\right]$$

be the basis matrix at the beginning of an iteration and let

$$\overline{\mathbf{B}} = \left[\mathbf{A}_{B(1)} \cdots \mathbf{A}_{B(\ell-1)} \;\; \mathbf{A}_j \;\; \mathbf{A}_{B(\ell+1)} \cdots \mathbf{A}_{B(m)}\right]$$

be the basis matrix at the beginning of the next iteration. These two basis matrices have the same columns except that the ℓth column $\mathbf{A}_{B(\ell)}$ (the one that exits the basis) has been replaced by $\mathbf{A}_j$. It is then reasonable to expect that $\mathbf{B}^{-1}$ contains information that can be exploited in the computation of $\overline{\mathbf{B}}^{-1}$. After we develop some needed tools and terminology, we will see that this is indeed the case. An alternative explanation and line of development is outlined in Exercise 3.13.

Definition 3.4 *Given a matrix, not necessarily square, the operation of adding a constant multiple of one row to the same or to another row is called* **an elementary row operation.**

The example that follows indicates that performing an elementary row operation on a matrix $\mathbf{C}$ is equivalent to forming the matrix $\mathbf{QC}$, where $\mathbf{Q}$ is a suitably constructed square matrix.

Example 3.3 Let

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix},$$

and note that

$$\mathbf{QC} = \begin{bmatrix} 11 & 14 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

In particular, multiplication from the left by the matrix $\mathbf{Q}$ has the effect of multiplying the third row of $\mathbf{C}$ by two and adding it to the first row.

Generalizing Example 3.3, we see that multiplying the jth row by β and adding it to the ith row (for $i \neq j$) is the same as left-multiplying by the matrix $\mathbf{Q} = \mathbf{I} + \mathbf{D}_{ij}$, where $\mathbf{D}_{ij}$ is a matrix with all entries equal to zero, except for the (i, j)th entry which is equal to β. The determinant of such a matrix $\mathbf{Q}$ is equal to 1 and, therefore, $\mathbf{Q}$ is invertible.

Suppose now that we apply a *sequence* of K elementary row operations and that the kth such operation corresponds to left-multiplication by a certain invertible matrix $\mathbf{Q}_k$. Then, the sequence of these elementary row operations is the same as left-multiplication by the invertible matrix $\mathbf{Q}_K \mathbf{Q}_{K-1} \cdots \mathbf{Q}_2 \mathbf{Q}_1$. We conclude that performing a *sequence* of elementary row operations on a given matrix is equivalent to left-multiplying that matrix by a certain invertible matrix.

Since $\mathbf{B}^{-1}\mathbf{B} = \mathbf{I}$, we see that $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$ is the ith unit vector $\mathbf{e}_i$. Using this observation, we have

$$\mathbf{B}^{-1}\overline{\mathbf{B}} = \begin{bmatrix} | & & | & | & | & & | \\ \mathbf{e}_1 & \cdots & \mathbf{e}_{\ell-1} & \mathbf{u} & \mathbf{e}_{\ell+1} & \cdots & \mathbf{e}_m \\ | & & | & | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & u_1 & & \\ & \ddots & \vdots & & \\ & & u_\ell & & \\ & & \vdots & \ddots & \\ & & u_m & & 1 \end{bmatrix},$$

where $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$. Let us apply a sequence of elementary row operations that will change the above matrix to the identity matrix. In particular, consider the following sequence of elementary row operations.

(a) For each $i \neq \ell$, we add the ℓth row times $-u_i/u_\ell$ to the ith row. (Recall that $u_\ell > 0$.) This replaces u_i by zero.

(b) We divide the ℓth row by u_ℓ. This replaces u_ℓ by one.

In words, we are adding to each row a multiple of the ℓth row to replace the ℓth column $\mathbf{u}$ by the ℓth unit vector $\mathbf{e}_\ell$. This sequence of elementary row operations is equivalent to left-multiplying $\mathbf{B}^{-1}\overline{\mathbf{B}}$ by a certain invertible matrix $\mathbf{Q}$. Since the result is the identity, we have $\mathbf{QB}^{-1}\overline{\mathbf{B}} = \mathbf{I}$, which yields $\mathbf{QB}^{-1} = \overline{\mathbf{B}}^{-1}$. The last equation shows that if we apply the same sequence of row operations to the matrix $\mathbf{B}^{-1}$ (equivalently, left-multiply by $\mathbf{Q}$), we obtain $\overline{\mathbf{B}}^{-1}$. We conclude that all it takes to generate $\overline{\mathbf{B}}^{-1}$, is to start with $\mathbf{B}^{-1}$ and apply the sequence of elementary row operations described above.

Example 3.4 Let

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 3 & 1 \\ 4 & -3 & -2 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} -4 \\ 2 \\ 2 \end{bmatrix},$$

and suppose that $\ell = 3$. Thus, our objective is to transform the vector $\mathbf{u}$ to the unit vector $\mathbf{e}_3 = (0, 0, 1)$. We multiply the third row by 2 and add it to the first row. We subtract the third row from the second row. Finally, we divide the third row by 2. We obtain

$$\overline{\mathbf{B}}^{-1} = \begin{bmatrix} 9 & -4 & -1 \\ -6 & 6 & 3 \\ 2 & -1.5 & -1 \end{bmatrix}.$$

When the matrix $\mathbf{B}^{-1}$ is updated in the manner we have described, we obtain an implementation of the simplex method known as the *revised simplex method*, which we summarize below.

An iteration of the revised simplex method

1. In a typical iteration, we start with a basis consisting of the basic columns $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(m)}$, an associated basic feasible solution $\mathbf{x}$, and the inverse $\mathbf{B}^{-1}$ of the basis matrix.
2. Compute the row vector $\mathbf{p}' = \mathbf{c}'_B\mathbf{B}^{-1}$ and then compute the reduced costs $\bar{c}_j = c_j - \mathbf{p}'\mathbf{A}_j$. If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates; else, choose some j for which $\bar{c}_j < 0$.
3. Compute $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$. If no component of $\mathbf{u}$ is positive, the optimal cost is $-\infty$, and the algorithm terminates.

4. If some component of $\mathbf{u}$ is positive, let
$$\theta^* = \min_{\{i=1,\ldots,m \mid u_i > 0\}} \frac{x_{B(i)}}{u_i}.$$
5. Let ℓ be such that $\theta^* = x_{B(\ell)}/u_\ell$. Form a new basis by replacing $\mathbf{A}_{B(\ell)}$ with $\mathbf{A}_j$. If $\mathbf{y}$ is the new basic feasible solution, the values of the new basic variables are $y_j = \theta^*$ and $y_{B(i)} = x_{B(i)} - \theta^* u_i$, $i \neq \ell$.
6. Form the $m \times (m+1)$ matrix $[\mathbf{B}^{-1} \mid \mathbf{u}]$. Add to each one of its rows a multiple of the ℓth row to make the last column equal to the unit vector $\mathbf{e}_\ell$. The first m columns of the result is the matrix $\overline{\mathbf{B}}^{-1}$.

The full tableau implementation

We finally describe the implementation of simplex method in terms of the so-called *full tableau*. Here, instead of maintaining and updating the matrix $\mathbf{B}^{-1}$, we maintain and update the $m \times (n+1)$ matrix

$$\mathbf{B}^{-1}\left[\mathbf{b} \mid \mathbf{A}\right]$$

with columns $\mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{B}^{-1}\mathbf{A}_1, \ldots, \mathbf{B}^{-1}\mathbf{A}_n$. This matrix is called the *simplex tableau*. Note that the column $\mathbf{B}^{-1}\mathbf{b}$, called the *zeroth column*, contains the values of the basic variables. The column $\mathbf{B}^{-1}\mathbf{A}_i$ is called the ith column of the tableau. The column $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$ corresponding to the variable that enters the basis is called the *pivot column*. If the ℓth basic variable exits the basis, the ℓth row of the tableau is called the *pivot row*. Finally, the element belonging to both the pivot row and the pivot column is called the *pivot element*. Note that the pivot element is u_ℓ and is always positive (unless $\mathbf{u} \leq \mathbf{0}$, in which case the algorithm has met the termination condition in Step 3).

The information contained in the rows of the tableau admits the following interpretation. The equality constraints are initially given to us in the form $\mathbf{b} = \mathbf{A}\mathbf{x}$. Given the current basis matrix $\mathbf{B}$, these equality constraints can also be expressed in the equivalent form

$$\mathbf{B}^{-1}\mathbf{b} = \mathbf{B}^{-1}\mathbf{A}\mathbf{x},$$

which is precisely the information in the tableau. In other words, the rows of the tableau provide us with the coefficients of the equality constraints $\mathbf{B}^{-1}\mathbf{b} = \mathbf{B}^{-1}\mathbf{A}\mathbf{x}$.

At the end of each iteration, we need to update the tableau $\mathbf{B}^{-1}[\mathbf{b} \mid \mathbf{A}]$ and compute $\overline{\mathbf{B}}^{-1}[\mathbf{b} \;\; \mathbf{A}]$. This can be accomplished by left-multiplying the

simplex tableau with a matrix $\mathbf{Q}$ satisfying $\mathbf{Q}\mathbf{B}^{-1} = \overline{\mathbf{B}}^{-1}$. As explained earlier, this is the same as performing those elementary row operations that turn $\mathbf{B}^{-1}$ to $\overline{\mathbf{B}}^{-1}$; that is, we add to each row a multiple of the pivot row to set all entries of the pivot column to zero, with the exception of the pivot element which is set to one.

Regarding the determination of the exiting column $\mathbf{A}_{B(\ell)}$ and the stepsize θ^*, Steps 4 and 5 in the summary of the simplex method amount to the following: $x_{B(i)}/u_i$ is the ratio of the ith entry in the zeroth column of the tableau to the ith entry in the pivot column of the tableau. We only consider those i for which u_i is positive. The smallest ratio is equal to θ^* and determines ℓ.

It is customary to augment the simplex tableau by including a top row, to be referred to as the *zeroth row*. The entry at the top left corner contains the value $-\mathbf{c}_B'\mathbf{x}_B$, which is the negative of the current cost. (The reason for the minus sign is that it allows for a simple update rule, as will be seen shortly.) The rest of the zeroth row is the row vector of reduced costs, that is, the vector $\bar{\mathbf{c}}' = \mathbf{c}' - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A}$. Thus, the structure of the tableau is:

$-\mathbf{c}_B'\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{c}' - \mathbf{c}_B'\mathbf{B}^{-1}\mathbf{A}$
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$

or, in more detail,

$-\mathbf{c}_B'\mathbf{x}_B$	$\bar{c}_1$	$\ldots$	$\bar{c}_n$
$x_{B(1)}$			
$\vdots$	$\mathbf{B}^{-1}\mathbf{A}_1$	$\ldots$	$\mathbf{B}^{-1}\mathbf{A}_n$
$x_{B(m)}$			

The rule for updating the zeroth row turns out to be identical to the rule used for the other rows of the tableau: add a multiple of the pivot row to the zeroth row to set the reduced cost of the entering variable to zero. We will now verify that this update rule produces the correct results for the zeroth row.

At the beginning of a typical iteration, the zeroth row is of the form

$$[0 \mid \mathbf{c}'] - \mathbf{g}'[\mathbf{b} \mid \mathbf{A}]$$

where $\mathbf{g}' = \mathbf{c}_B'\mathbf{B}^{-1}$. Hence, the zeroth row is equal to $[0 \mid \mathbf{c}']$ plus a linear combination of the rows of $[\mathbf{b} \mid \mathbf{A}]$. Let column j be the pivot column, and row ℓ be the pivot row. Note that the pivot row is of the form $\mathbf{h}'[\mathbf{b} \mid \mathbf{A}]$, where the vector $\mathbf{h}'$ is the ℓth row of $\mathbf{B}^{-1}$. Hence, after a multiple of the

pivot row is added to the zeroth row, that row is again equal to $[0 \mid \mathbf{c}']$ plus a (different) linear combination of the rows of $[\mathbf{b} \mid \mathbf{A}]$, and is of the form

$$[0 \mid \mathbf{c}'] - \mathbf{p}'[\mathbf{b} \mid \mathbf{A}],$$

for some vector $\mathbf{p}$. Recall that our update rule is such that the pivot column entry of the zeroth row becomes zero, that is,

$$c_{\overline{B}(\ell)} - \mathbf{p}'\mathbf{A}_{\overline{B}(\ell)} = c_j - \mathbf{p}'\mathbf{A}_j = 0.$$

Consider now the $\overline{B}(i)$th column for $i \neq \ell$. (This is a column corresponding to a basic variable that stays in the basis.) The zeroth row entry of that column is zero, before the change of basis, since it is the reduced cost of a basic variable. Because $\mathbf{B}^{-1}\mathbf{A}_{B(i)}$ is the ith unit vector and $i \neq \ell$, the entry in the pivot row for that column is also equal to zero. Hence, adding a multiple of the pivot row to the zeroth row of the tableau does not affect the zeroth row entry of that column, which is left at zero. We conclude that the vector $\mathbf{p}$ satisfies $c_{\overline{B}(i)} - \mathbf{p}'\mathbf{A}_{\overline{B}(i)} = 0$ for every column $\mathbf{A}_{\overline{B}(i)}$ in the new basis. This implies that $\mathbf{c}'_{\overline{B}} - \mathbf{p}'\overline{\mathbf{B}} = \mathbf{0}$, and $\mathbf{p}' = \mathbf{c}'_{\overline{B}}\overline{\mathbf{B}}^{-1}$. Hence, with our update rule, the updated zeroth row of the tableau is equal to

$$[0 \mid \mathbf{c}'] - \mathbf{c}'_{\overline{B}}\overline{\mathbf{B}}^{-1}[\mathbf{b} \mid \mathbf{A}],$$

as desired.

We can now summarize the mechanics of the full tableau implementation.

An iteration of the full tableau implementation

1. A typical iteration starts with the tableau associated with a basis matrix $\mathbf{B}$ and the corresponding basic feasible solution $\mathbf{x}$.
2. Examine the reduced costs in the zeroth row of the tableau. If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates; else, choose some j for which $\overline{c}_j < 0$.
3. Consider the vector $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$, which is the jth column (the pivot column) of the tableau. If no component of $\mathbf{u}$ is positive, the optimal cost is $-\infty$, and the algorithm terminates.
4. For each i for which u_i is positive, compute the ratio $x_{B(i)}/u_i$. Let ℓ be the index of a row that corresponds to the smallest ratio. The column $\mathbf{A}_{B(\ell)}$ exits the basis and the column $\mathbf{A}_j$ enters the basis.
5. Add to each row of the tableau a constant multiple of the ℓth row (the pivot row) so that u_ℓ (the pivot element) becomes one and all other entries of the pivot column become zero.

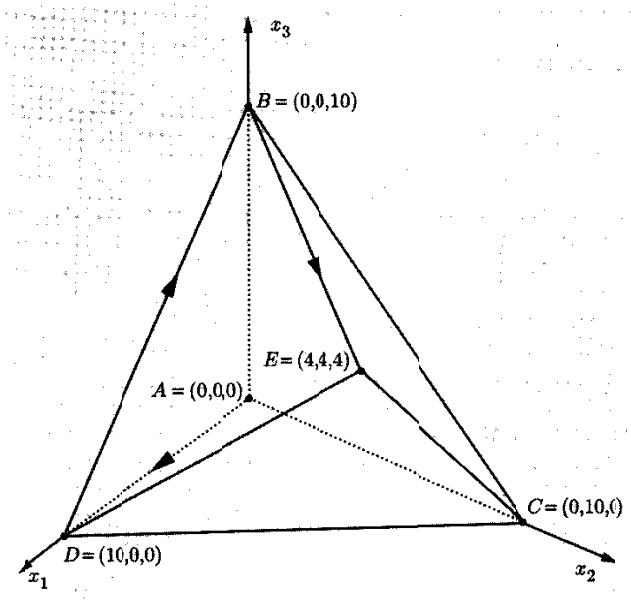

Figure 3.4: The feasible set in Example 3.5. Note that we have five extreme points. These are $A = (0,0,0)$ with cost 0, $B = (0,0,10)$ with cost -120, $C = (0,10,0)$ with cost -120, $D = (10,0,0)$ with cost -100, and $E = (4,4,4)$ with cost -136. In particular, E is the unique optimal solution.

Example 3.5 Consider the problem

$$\begin{array}{lrcl} \text{minimize} & -10x_1 - 12x_2 - 12x_3 & & \\ \text{subject to} & x_1 + 2x_2 + 2x_3 & \leq & 20 \\ & 2x_1 + x_2 + 2x_3 & \leq & 20 \\ & 2x_1 + 2x_2 + x_3 & \leq & 20 \\ & x_1, x_2, x_3 \geq 0. & & \end{array}$$

The feasible set is shown in Figure 3.4.

After introducing slack variables, we obtain the following standard form problem:

$$\begin{array}{lrcl} \text{minimize} & -10x_1 - 12x_2 - 12x_3 & & \\ \text{subject to} & x_1 + 2x_2 + 2x_3 + x_4 & = & 20 \\ & 2x_1 + x_2 + 2x_3 + x_5 & = & 20 \\ & 2x_1 + 2x_2 + x_3 + x_6 & = & 20 \\ & x_1, \ldots, x_6 \geq 0. & & \end{array}$$

Note that $\mathbf{x} = (0,0,0,20,20,20)$ is a basic feasible solution and can be used to start the algorithm. Let accordingly, $B(1) = 4$, $B(2) = 5$, and $B(3) = 6$. The

corresponding basis matrix is the identity matrix $\mathbf{I}$. To obtain the zeroth row of the initial tableau, we note that $\mathbf{c}_B = \mathbf{0}$ and, therefore, $\mathbf{c}_B'\mathbf{x}_B = 0$ and $\bar{\mathbf{c}} = \mathbf{c}$. Hence, we have the following initial tableau:

		x_1	x_2	x_3	x_4	x_5	x_6
	0	−10	−12	−12	0	0	0
$x_4 =$	20	1	2	2	1	0	0
$x_5 =$	20	2*	1	2	0	1	0
$x_6 =$	20	2	2	1	0	0	1

We note a few conventions in the format of the above tableau: the label x_i on top of the ith column indicates the variable associated with that column. The labels "$x_i =$" to the left of the tableau tell us which are the basic variables and in what order. For example, the first basic variable $x_{B(1)}$ is x_4, and its value is 20. Similarly, $x_{B(2)} = x_5 = 20$, and $x_{B(3)} = x_6 = 20$. Strictly speaking, these labels are not quite necessary. We know that the column in the tableau associated with the first basic variable must be the first unit vector. Once we observe that the column associated with the variable x_4 is the first unit vector, it follows that x_4 is the first basic variable.

We continue with our example. The reduced cost of x_1 is negative and we let that variable enter the basis. The pivot column is $\mathbf{u} = (1, 2, 2)$. We form the ratios $x_{B(i)}/u_i$, $i = 1,2,3$; the smallest ratio corresponds to $i = 2$ and $i = 3$. We break this tie by choosing $\ell = 2$. This determines the pivot element, which we indicate by an asterisk. The second basic variable $x_{B(2)}$, which is x_5, exits the basis. The new basis is given by $\overline{B}(1) = 4, \overline{B}(2) = 1$, and $\overline{B}(3) = 6$. We multiply the pivot row by 5 and add it to the zeroth row. We multiply the pivot row by 1/2 and subtract it from the first row. We subtract the pivot row from the third row. Finally, we divide the pivot row by 2. This leads us to the new tableau:

		x_1	x_2	x_3	x_4	x_5	x_6
	100	0	−7	−2	0	5	0
$x_4 =$	10	0	1.5	1*	1	−0.5	0
$x_1 =$	10	1	0.5	1	0	0.5	0
$x_6 =$	0	0	1	−1	0	−1	1

The corresponding basic feasible solution is $\mathbf{x} = (10, 0, 0, 10, 0, 0)$. In terms of the original variables x_1, x_2, x_3, we have moved to point $D = (10, 0, 0)$ in Figure 3.4. Note that this is a degenerate basic feasible solution, because the basic variable x_6 is equal to zero. This agrees with Figure 3.4 where we observe that there are four active constraints at point D.

We have mentioned earlier that the rows of the tableau (other than the zeroth row) amount to a representation of the equality constraints $\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \mathbf{B}^{-1}\mathbf{b}$, which are equivalent to the original constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$. In our current

example, the tableau indicates that the equality constraints can be written in the equivalent form:

$$\begin{aligned} 10 &= \quad 1.5x_2 + x_3 + x_4 - 0.5x_5 \\ 10 &= x_1 + 0.5x_2 + x_3 \quad + 0.5x_5 \\ 0 &= \quad x_2 - x_3 \quad - x_5 + x_6. \end{aligned}$$

We now return to the simplex method. With the current tableau, the variables x_2 and x_3 have negative reduced costs. Let us choose x_3 to be the one that enters the basis. The pivot column is $\mathbf{u} = (1, 1, -1)$. Since $u_3 < 0$, we only form the ratios $x_{B(i)}/u_i$, for $i = 1, 2$. There is again a tie, which we break by letting $\ell = 1$, and the first basic variable, x_4, exits the basis. The pivot element is again indicated by an asterisk. After carrying out the necessary elementary row operations, we obtain the following new tableau:

		x_1	x_2	x_3	x_4	x_5	x_6
	120	0	−4	0	2	4	0
$x_3 =$	10	0	1.5	1	1	−0.5	0
$x_1 =$	0	1	−1	0	−1	1	0
$x_6 =$	10	0	2.5*	0	1	−1.5	1

In terms of Figure 3.4, we have moved to point $B = (0, 0, 10)$, and the cost has been reduced to -120. At this point, x_2 is the only variable with negative reduced cost. We bring x_2 into the basis, x_6 exits, and the resulting tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6
	136	0	0	0	3.3	1.6	1.6
$x_3 =$	4	0	0	1	0.4	0.4	−0.6
$x_1 =$	4	1	0	0	−0.3	0.4	0.4
$x_2 =$	4	0	1	0	0.4	−0.6	0.4

We have now reached point E in Figure 3.4. Its optimality is confirmed by observing that all reduced costs are nonnegative.

In this example, the simplex method took three changes of basis to reach the optimal solution, and it traced the path $A - D - B - E$ in Figure 3.4. With different pivoting rules, a different path would have been traced. Could the simplex method have solved the problem by tracing the path $A - D - E$, which involves only two edges, with only two iterations? The answer is no. The initial and final bases differ in three columns, and therefore at least three basis changes are required. In particular, if the method were to trace the path $A - D - E$, there would be a degenerate change of basis at point D (with no edge being traversed), which would again bring the total to three.

Example 3.6 This example shows that the simplex method can indeed cycle. We consider a problem described in terms of the following initial tableau.

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	−3/4	20	−1/2	6	0	0	0
$x_5 =$	0	1/4*	−8	−1	9	1	0	0
$x_6 =$	0	1/2	−12	−1/2	3	0	1	0
$x_7 =$	1	0	0	1	0	0	0	1

We use the following pivoting rules:

(a) We select a nonbasic variable with the most negative reduced cost $\bar{c}_j$ to be the one that enters the basis.

(b) Out of all basic variables that are eligible to exit the basis, we select the one with the smallest subscript.

We then obtain the following sequence of tableaux (the pivot element is indicated by an asterisk):

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	0	−4	−7/2	33	3	0	0
$x_1 =$	0	1	−32	−4	36	4	0	0
$x_6 =$	0	0	4*	3/2	−15	−2	1	0
$x_7 =$	1	0	0	1	0	0	0	1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	0	0	−2	18	1	1	0
$x_1 =$	0	1	0	8*	−84	−12	8	0
$x_2 =$	0	0	1	3/8	−15/4	−1/2	1/4	0
$x_7 =$	1	0	0	1	0	0	0	1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	1/4	0	0	−3	−2	3	0
$x_3 =$	0	1/8	0	1	−21/2	−3/2	1	0
$x_2 =$	0	−3/64	1	0	3/16*	1/16	−1/8	0
$x_7 =$	1	−1/8	0	0	21/2	3/2	−1	1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	−1/2	16	0	0	−1	1	0
$x_3 =$	0	−5/2	56	1	0	2*	−6	0
$x_4 =$	0	−1/4	16/3	0	1	1/3	−2/3	0
$x_7 =$	1	5/2	−56	0	0	−2	6	1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	−7/4	44	1/2	0	0	−2	0
$x_5 =$	0	−5/4	28	1/2	0	1	−3	0
$x_4 =$	0	1/6	−4	−1/6	1	0	1/3*	0
$x_7 =$	1	0	0	1	0	0	0	1

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	−3/4	20	−1/2	6	0	0	0
$x_5 =$	0	1/4	−8	−1	9	1	0	0
$x_3 =$	0	1/2	−12	−1/2	3	0	1	0
$x_7 =$	1	0	0	1	0	0	0	1

After six pivots, we have the same basis and the same tableau that we started with. At each basis change, we had $\theta^* = 0$. In particular, for each intermediate tableau, we had the same feasible solution and the same cost. The same sequence of pivots can be repeated over and over, and the simplex method never terminates.

Comparison of the full tableau and the revised simplex methods

Let us pretend that the problem is changed to

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\mathbf{x} + \mathbf{0}'\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0}. \end{aligned}$$

We implement the simplex method on this new problem, except that we never allow any of the components of the vector $\mathbf{y}$ to become basic. Then, the simplex method performs basis changes as if the vector $\mathbf{y}$ were entirely

absent. Note also that the vector of reduced costs in the augmented problem is

$$[\mathbf{c}' \mid \mathbf{0}'] - \mathbf{c}'_B\mathbf{B}^{-1}[\mathbf{A} \mid \mathbf{I}] = [\bar{\mathbf{c}}' \mid -\mathbf{c}'_B\mathbf{B}^{-1}].$$

Thus, the simplex tableau for the augmented problem takes the form

$-\mathbf{c}'_B\mathbf{B}^{-1}\mathbf{b}$	$\bar{\mathbf{c}}'$	$-\mathbf{c}'_B\mathbf{B}^{-1}$
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$	$\mathbf{B}^{-1}$

In particular, by following the mechanics of the full tableau method on the above tableau, the inverse basis matrix $\mathbf{B}^{-1}$ is made available at each iteration. We can now think of the revised simplex method as being essentially the same as the full tableau method applied to the above augmented problem, except that the part of the tableau containing $\mathbf{B}^{-1}\mathbf{A}$ is never formed explicitly; instead, once the entering variable x_j is chosen, the pivot column $\mathbf{B}^{-1}\mathbf{A}_j$ is computed on the fly. Thus, the revised simplex method is just a variant of the full tableau method, with more efficient bookkeeping. If the revised simplex method also updates the zeroth row entries that lie on top of $\mathbf{B}^{-1}$ (by the usual elementary operations), the simplex multipliers $\mathbf{p}' = \mathbf{c}'_B\mathbf{B}^{-1}$ become available, thus eliminating the need for solving the linear system $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ at each iteration.

We now discuss the relative merits of the two methods. The full tableau method requires a constant (and small) number of arithmetic operations for updating each entry of the tableau. Thus, the amount of computation per iteration is proportional to the size of the tableau, which is $O(mn)$. The revised simplex method uses similar computations to update $\mathbf{B}^{-1}$ and $\mathbf{c}'_B\mathbf{B}^{-1}$, and since only $O(m^2)$ entries are updated, the computational requirements per iteration are $O(m^2)$. In addition, the reduced cost of each variable x_j can be computed by forming the inner product $\mathbf{p}'\mathbf{A}_j$, which requires $O(m)$ operations. In the worst case, the reduced cost of every variable is computed, for a total of $O(mn)$ computations per iteration. Since $m \leq n$, the worst-case computational effort per iteration is $O(mn+m^2) = O(mn)$, under either implementation. On the other hand, if we consider a pivoting rule that evaluates one reduced cost at a time, until a negative reduced cost is found, a typical iteration of the revised simplex method might require a lot less work. In the best case, if the first reduced cost computed is negative, and the corresponding variable is chosen to enter the basis, the total computational effort is only $O(m^2)$. The conclusion is that the revised simplex method cannot be slower than the full tableau method, and could be much faster during most iterations.

Another important element in favor of the revised simplex method is that memory requirements are reduced from $O(mn)$ to $O(m^2)$. As n is often much larger than m, this effect can be quite significant. It could be counterargued that the memory requirements of the revised simplex method

are also $O(mn)$ because of the need to store the matrix $\mathbf{A}$. However, in most large scale problems that arise in applications, the matrix $\mathbf{A}$ is very sparse (has many zero entries) and can be stored compactly. (Note that the sparsity of $\mathbf{A}$ does not usually help in the storage of the full simplex tableau because even if $\mathbf{A}$ and $\mathbf{B}$ are sparse, $\mathbf{B}^{-1}\mathbf{A}$ is not sparse, in general.)

We summarize this discussion in the following table:

	Full tableau	**Revised simplex**
Memory	$O(mn)$	$O(m^2)$
Worst-case time	$O(mn)$	$O(mn)$
Best-case time	$O(mn)$	$O(m^2)$

Table 3.1: Comparison of the full tableau method and revised simplex. The time requirements refer to a single iteration.

Practical performance enhancements

Practical implementations of the simplex method aimed at solving problems of moderate or large size incorporate a number of additional ideas from numerical linear algebra which we briefly mention.

The first idea is related to *reinversion*. Recall that at each iteration of the revised simplex method, the inverse basis matrix $\mathbf{B}^{-1}$ is updated according to certain rules. Each such iteration may introduce roundoff or truncation errors which accumulate and may eventually lead to highly inaccurate results. For this reason, it is customary to recompute the matrix $\mathbf{B}^{-1}$ from scratch once in a while. The efficiency of such reinversions can be greatly enhanced by using suitable data structures and certain techniques from computational linear algebra.

Another set of ideas is related to the way that the inverse basis matrix $\mathbf{B}^{-1}$ is represented. Suppose that a reinversion has been just carried out and $\mathbf{B}^{-1}$ is available. Subsequent to the current iteration of the revised simplex method, we have the option of generating explicitly and storing the new inverse basis matrix $\overline{\mathbf{B}}^{-1}$. An alternative that carries the same information, is to store a matrix $\mathbf{Q}$ such that $\mathbf{Q}\mathbf{B}^{-1} = \overline{\mathbf{B}}^{-1}$. Note that $\mathbf{Q}$ basically prescribes which elementary row operations need to be applied to $\mathbf{B}^{-1}$ in order to produce $\overline{\mathbf{B}}^{-1}$. It is not a full matrix, and can be completely specified in terms of m coefficients: for each row, we need to know what multiple of the pivot row must be added to it.

Suppose now that we wish to solve the system $\overline{\mathbf{B}}\mathbf{u} = \mathbf{A}_j$ for $\mathbf{u}$, where $\mathbf{A}_j$ is the entering column, as is required by the revised simplex method. We have $\mathbf{u} = \overline{\mathbf{B}}^{-1}\mathbf{A}_j = \mathbf{Q}\mathbf{B}^{-1}\mathbf{A}_j$, which shows that we can first compute

absent. Note also that the vector of reduced costs in the augmented problem is

$$[\mathbf{c}' \mid \mathbf{0}'] - \mathbf{c}'_B\mathbf{B}^{-1}[\mathbf{A} \mid \mathbf{I}] = [\bar{\mathbf{c}}' \mid -\mathbf{c}'_B\mathbf{B}^{-1}].$$

Thus, the simplex tableau for the augmented problem takes the form

$-\mathbf{c}'_B\mathbf{B}^{-1}\mathbf{b}$	$\bar{\mathbf{c}}'$	$-\mathbf{c}'_B\mathbf{B}^{-1}$
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$	$\mathbf{B}^{-1}$

In particular, by following the mechanics of the full tableau method on the above tableau, the inverse basis matrix $\mathbf{B}^{-1}$ is made available at each iteration. We can now think of the revised simplex method as being essentially the same as the full tableau method applied to the above augmented problem, except that the part of the tableau containing $\mathbf{B}^{-1}\mathbf{A}$ is never formed explicitly; instead, once the entering variable x_j is chosen, the pivot column $\mathbf{B}^{-1}\mathbf{A}_j$ is computed on the fly. Thus, the revised simplex method is just a variant of the full tableau method, with more efficient bookkeeping. If the revised simplex method also updates the zeroth row entries that lie on top of $\mathbf{B}^{-1}$ (by the usual elementary operations), the simplex multipliers $\mathbf{p}' = \mathbf{c}'_B\mathbf{B}^{-1}$ become available, thus eliminating the need for solving the linear system $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ at each iteration.

We now discuss the relative merits of the two methods. The full tableau method requires a constant (and small) number of arithmetic operations for updating each entry of the tableau. Thus, the amount of computation per iteration is proportional to the size of the tableau, which is $O(mn)$. The revised simplex method uses similar computations to update $\mathbf{B}^{-1}$ and $\mathbf{c}'_B\mathbf{B}^{-1}$, and since only $O(m^2)$ entries are updated, the computational requirements per iteration are $O(m^2)$. In addition, the reduced cost of each variable x_j can be computed by forming the inner product $\mathbf{p}'\mathbf{A}_j$, which requires $O(m)$ operations. In the worst case, the reduced cost of every variable is computed, for a total of $O(mn)$ computations per iteration. Since $m \leq n$, the worst-case computational effort per iteration is $O(mn+m^2) = O(mn)$, under either implementation. On the other hand, if we consider a pivoting rule that evaluates one reduced cost at a time, until a negative reduced cost is found, a typical iteration of the revised simplex method might require a lot less work. In the best case, if the first reduced cost computed is negative, and the corresponding variable is chosen to enter the basis, the total computational effort is only $O(m^2)$. The conclusion is that the revised simplex method cannot be slower than the full tableau method, and could be much faster during most iterations.

Another important element in favor of the revised simplex method is that memory requirements are reduced from $O(mn)$ to $O(m^2)$. As n is often much larger than m, this effect can be quite significant. It could be counterargued that the memory requirements of the revised simplex method

are also $O(mn)$ because of the need to store the matrix $\mathbf{A}$. However, in most large scale problems that arise in applications, the matrix $\mathbf{A}$ is very sparse (has many zero entries) and can be stored compactly. (Note that the sparsity of $\mathbf{A}$ does not usually help in the storage of the full simplex tableau because even if $\mathbf{A}$ and $\mathbf{B}$ are sparse, $\mathbf{B}^{-1}\mathbf{A}$ is not sparse, in general.)

We summarize this discussion in the following table:

	Full tableau	**Revised simplex**
Memory	$O(mn)$	$O(m^2)$
Worst-case time	$O(mn)$	$O(mn)$
Best-case time	$O(mn)$	$O(m^2)$

Table 3.1: Comparison of the full tableau method and revised simplex. The time requirements refer to a single iteration.

Practical performance enhancements

Practical implementations of the simplex method aimed at solving problems of moderate or large size incorporate a number of additional ideas from numerical linear algebra which we briefly mention.

The first idea is related to *reinversion*. Recall that at each iteration of the revised simplex method, the inverse basis matrix $\mathbf{B}^{-1}$ is updated according to certain rules. Each such iteration may introduce roundoff or truncation errors which accumulate and may eventually lead to highly inaccurate results. For this reason, it is customary to recompute the matrix $\mathbf{B}^{-1}$ from scratch once in a while. The efficiency of such reinversions can be greatly enhanced by using suitable data structures and certain techniques from computational linear algebra.

Another set of ideas is related to the way that the inverse basis matrix $\mathbf{B}^{-1}$ is represented. Suppose that a reinversion has been just carried out and $\mathbf{B}^{-1}$ is available. Subsequent to the current iteration of the revised simplex method, we have the option of generating explicitly and storing the new inverse basis matrix $\overline{\mathbf{B}}^{-1}$. An alternative that carries the same information, is to store a matrix $\mathbf{Q}$ such that $\mathbf{Q}\mathbf{B}^{-1} = \overline{\mathbf{B}}^{-1}$. Note that $\mathbf{Q}$ basically prescribes which elementary row operations need to be applied to $\mathbf{B}^{-1}$ in order to produce $\overline{\mathbf{B}}^{-1}$. It is not a full matrix, and can be completely specified in terms of m coefficients: for each row, we need to know what multiple of the pivot row must be added to it.

Suppose now that we wish to solve the system $\overline{\mathbf{B}}\mathbf{u} = \mathbf{A}_j$ for $\mathbf{u}$, where $\mathbf{A}_j$ is the entering column, as is required by the revised simplex method. We have $\mathbf{u} = \overline{\mathbf{B}}^{-1}\mathbf{A}_j = \mathbf{Q}\mathbf{B}^{-1}\mathbf{A}_j$, which shows that we can first compute

absent. Note also that the vector of reduced costs in the augmented problem is

$$[\mathbf{c}' \mid \mathbf{0}'] - \mathbf{c}'_B\mathbf{B}^{-1}[\mathbf{A} \mid \mathbf{I}] = [\bar{\mathbf{c}}' \mid -\mathbf{c}'_B\mathbf{B}^{-1}].$$

Thus, the simplex tableau for the augmented problem takes the form

$-\mathbf{c}'_B\mathbf{B}^{-1}\mathbf{b}$	$\bar{\mathbf{c}}'$	$-\mathbf{c}'_B\mathbf{B}^{-1}$
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$	$\mathbf{B}^{-1}$

In particular, by following the mechanics of the full tableau method on the above tableau, the inverse basis matrix $\mathbf{B}^{-1}$ is made available at each iteration. We can now think of the revised simplex method as being essentially the same as the full tableau method applied to the above augmented problem, except that the part of the tableau containing $\mathbf{B}^{-1}\mathbf{A}$ is never formed explicitly; instead, once the entering variable x_j is chosen, the pivot column $\mathbf{B}^{-1}\mathbf{A}_j$ is computed on the fly. Thus, the revised simplex method is just a variant of the full tableau method, with more efficient bookkeeping. If the revised simplex method also updates the zeroth row entries that lie on top of $\mathbf{B}^{-1}$ (by the usual elementary operations), the simplex multipliers $\mathbf{p}' = \mathbf{c}'_B\mathbf{B}^{-1}$ become available, thus eliminating the need for solving the linear system $\mathbf{p}'\mathbf{B} = \mathbf{c}'_B$ at each iteration.

We now discuss the relative merits of the two methods. The full tableau method requires a constant (and small) number of arithmetic operations for updating each entry of the tableau. Thus, the amount of computation per iteration is proportional to the size of the tableau, which is $O(mn)$. The revised simplex method uses similar computations to update $\mathbf{B}^{-1}$ and $\mathbf{c}'_B\mathbf{B}^{-1}$, and since only $O(m^2)$ entries are updated, the computational requirements per iteration are $O(m^2)$. In addition, the reduced cost of each variable x_j can be computed by forming the inner product $\mathbf{p}'\mathbf{A}_j$, which requires $O(m)$ operations. In the worst case, the reduced cost of every variable is computed, for a total of $O(mn)$ computations per iteration. Since $m \leq n$, the worst-case computational effort per iteration is $O(mn+m^2) = O(mn)$ under either implementation. On the other hand, if we consider a pivoting rule that evaluates one reduced cost at a time, until a negative reduced cost is found, a typical iteration of the revised simplex method might require a lot less work. In the best case, if the first reduced cost computed is negative, and the corresponding variable is chosen to enter the basis, the total computational effort is only $O(m^2)$. The conclusion is that the revised simplex method cannot be slower than the full tableau method, and could be much faster during most iterations.

Another important element in favor of the revised simplex method is that memory requirements are reduced from $O(mn)$ to $O(m^2)$. As n is often much larger than m, this effect can be quite significant. It could be counterargued that the memory requirements of the revised simplex method

are also $O(mn)$ because of the need to store the matrix $\mathbf{A}$. However, in most large scale problems that arise in applications, the matrix $\mathbf{A}$ is very sparse (has many zero entries) and can be stored compactly. (Note that the sparsity of $\mathbf{A}$ does not usually help in the storage of the full simplex tableau because even if $\mathbf{A}$ and $\mathbf{B}$ are sparse, $\mathbf{B}^{-1}\mathbf{A}$ is not sparse, in general.)

We summarize this discussion in the following table:

	Full tableau	**Revised simplex**
Memory	$O(mn)$	$O(m^2)$
Worst-case time	$O(mn)$	$O(mn)$
Best-case time	$O(mn)$	$O(m^2)$

Table 3.1: Comparison of the full tableau method and revised simplex. The time requirements refer to a single iteration.

Practical performance enhancements

Practical implementations of the simplex method aimed at solving problems of moderate or large size incorporate a number of additional ideas from numerical linear algebra which we briefly mention.

The first idea is related to *reinversion*. Recall that at each iteration of the revised simplex method, the inverse basis matrix $\mathbf{B}^{-1}$ is updated according to certain rules. Each such iteration may introduce roundoff or truncation errors which accumulate and may eventually lead to highly inaccurate results. For this reason, it is customary to recompute the matrix $\mathbf{B}^{-1}$ from scratch once in a while. The efficiency of such reinversions can be greatly enhanced by using suitable data structures and certain techniques from computational linear algebra.

Another set of ideas is related to the way that the inverse basis matrix $\mathbf{B}^{-1}$ is represented. Suppose that a reinversion has been just carried out and $\mathbf{B}^{-1}$ is available. Subsequent to the current iteration of the revised simplex method, we have the option of generating explicitly and storing the new inverse basis matrix $\overline{\mathbf{B}}^{-1}$. An alternative that carries the same information, is to store a matrix $\mathbf{Q}$ such that $\mathbf{Q}\mathbf{B}^{-1} = \overline{\mathbf{B}}^{-1}$. Note that $\mathbf{Q}$ basically prescribes which elementary row operations need to be applied to $\mathbf{B}^{-1}$ in order to produce $\overline{\mathbf{B}}^{-1}$. It is not a full matrix, and can be completely specified in terms of m coefficients: for each row, we need to know what multiple of the pivot row must be added to it.

Suppose now that we wish to solve the system $\overline{\mathbf{B}}\mathbf{u} = \mathbf{A}_j$ for $\mathbf{u}$, where $\mathbf{A}_j$ is the entering column, as is required by the revised simplex method. We have $\mathbf{u} = \overline{\mathbf{B}}^{-1}\mathbf{A}_j = \mathbf{Q}\mathbf{B}^{-1}\mathbf{A}_j$, which shows that we can first compute

$\mathbf{B}^{-1}\mathbf{A}_j$ and then left-multiply by $\mathbf{Q}$ (equivalently, apply a sequence of elementary row operations) to produce $\mathbf{u}$. The same idea can also be used to represent the inverse basis matrix after several simplex iterations, as a product of the initial inverse basis matrix and several sparse matrices like $\mathbf{Q}$.

The last idea we mention is the following. Subsequent to a "reinversion," one does not usually compute $\mathbf{B}^{-1}$ explicitly, but $\mathbf{B}^{-1}$ is instead represented in terms of sparse triangular matrices with a special structure.

The methods discussed in this subsection are designed to accomplish two objectives: improve numerical stability (minimize the effect of roundoff errors) and exploit sparsity in the problem data to improve both running time and memory requirements. These methods have a critical effect in practice. Besides having a better chance of producing numerically trustworthy results, they can also speed up considerably the running time of the simplex method. These techniques lie much closer to the subject of numerical linear algebra, as opposed to optimization, and for this reason we do not pursue them in any greater depth.

3.4 Anticycling: lexicography and Bland's rule

In this section, we discuss anticycling rules under which the simplex method is guaranteed to terminate, thus extending Theorem 3.3 to degenerate problems. As an important corollary, we conclude that if the optimal cost is finite, then there exists an optimal basis, that is, a basis satisfying $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$ and $\bar{\mathbf{c}}' = \mathbf{c}' - \mathbf{c}'_B\mathbf{B}^{-1}\mathbf{A} \geq \mathbf{0}'$.

Lexicography

We present here the lexicographic pivoting rule and prove that it prevents the simplex method from cycling. Historically, this pivoting rule was derived by analyzing the behavior of the simplex method on a nondegenerate problem obtained by means of a small perturbation of the right-hand side vector $\mathbf{b}$. This connection is pursued in Exercise 3.15.

We start with a definition.

Definition 3.5 *A vector $\mathbf{u} \in \Re^n$ is said to be* **lexicographically larger** *(or* **smaller***) than another vector $\mathbf{v} \in \Re^n$ if $\mathbf{u} \neq \mathbf{v}$ and the first nonzero component of $\mathbf{u} - \mathbf{v}$ is positive (or negative, respectively). Symbolically, we write*

$$\mathbf{u} \overset{L}{>} \mathbf{v} \quad \text{or} \quad \mathbf{u} \overset{L}{<} \mathbf{v}.$$

For example,

$$(0,\ 2,\ 3,\ 0) \overset{L}{>} (0,\ 2,\ 1,\ 4),$$

$$(0,\ 4,\ 5,\ 0) \overset{L}{<} (1,\ 2,\ 1,\ 2).$$

Lexicographic pivoting rule

1. Choose an entering column $\mathbf{A}_j$ arbitrarily, as long as its reduced cost $\bar{c}_j$ is negative. Let $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$ be the jth column of the tableau.
2. For each i with $u_i > 0$, divide the ith row of the tableau (including the entry in the zeroth column) by u_i and choose the lexicographically smallest row. If row ℓ is lexicographically smallest, then the ℓth basic variable $x_{B(\ell)}$ exits the basis.

Example 3.7 Consider the following tableau (the zeroth row is omitted), and suppose that the pivot column is the third one ($j = 3$).

1	0	5	3	$\cdots$
2	4	6	-1	$\cdots$
3	0	7	9	$\cdots$

Note that there is a tie in trying to determine the exiting variable because $x_{B(1)}/u_1 = 1/3$ and $x_{B(3)}/u_3 = 3/9 = 1/3$. We divide the first and third rows of the tableau by $u_1 = 3$ and $u_3 = 9$, respectively, to obtain:

1/3	0	5/3	1	$\cdots$
*	*	*	*	$\cdots$
1/3	0	7/9	1	$\cdots$

The tie between the first and third rows is resolved by performing a lexicographic comparison. Since $7/9 < 5/3$, the third row is chosen to be the pivot row, and the variable $x_{B(3)}$ exits the basis.

We note that the lexicographic pivoting rule always leads to a unique choice for the exiting variable. Indeed, if this were not the case, two of the rows in the tableau would have to be proportional. But if two rows of the matrix $\mathbf{B}^{-1}\mathbf{A}$ are proportional, the matrix $\mathbf{B}^{-1}\mathbf{A}$ has rank smaller than m and, therefore, $\mathbf{A}$ also has rank less than m, which contradicts our standing assumption that $\mathbf{A}$ has linearly independent rows.

Theorem 3.4 *Suppose that the simplex algorithm starts with all the rows in the simplex tableau, other than the zeroth row, lexicographically positive. Suppose that the lexicographic pivoting rule is followed. Then:*

(a) *Every row of the simplex tableau, other than the zeroth row, remains lexicographically positive throughout the algorithm.*

(b) *The zeroth row strictly increases lexicographically at each iteration.*

(c) *The simplex method terminates after a finite number of iterations.*

Proof.

(a) Suppose that all rows of the simplex tableau, other than the zeroth row, are lexicographically positive at the beginning of a simplex iteration. Suppose that x_j enters the basis and that the pivot row is the ℓth row. According to the lexicographic pivoting rule, we have $u_\ell > 0$ and

$$\frac{(\ell\text{th row})}{u_\ell} \overset{L}{<} \frac{(i\text{th row})}{u_i}, \qquad \text{if } i \neq \ell \text{ and } u_i > 0. \tag{3.5}$$

To determine the new tableau, the ℓth row is divided by the positive pivot element u_ℓ and, therefore, remains lexicographically positive. Consider the ith row and suppose that $u_i < 0$. In order to zero the (i,j)th entry of the tableau, we need to add a positive multiple of the pivot row to the ith row. Due to the lexicographic positivity of both rows, the ith row will remain lexicographically positive after this addition. Finally, consider the ith row for the case where $u_i > 0$ and $i \neq \ell$. We have

$$(\text{new } i\text{th row}) = (\text{old } i\text{th row}) - \frac{u_i}{u_\ell}(\text{old } \ell\text{th row}).$$

Because of the lexicographic inequality (3.5), which is satisfied by the old rows, the new ith row is also lexicographically positive.

(b) At the beginning of an iteration, the reduced cost in the pivot column is negative. In order to make it zero, we need to add a positive multiple of the pivot row. Since the latter row is lexicographically positive, the zeroth row increases lexicographically.

(c) Since the zeroth row increases lexicographically at each iteration, it never returns to a previous value. Since the zeroth row is determined completely by the current basis, no basis can be repeated twice and the simplex method must terminate after a finite number of iterations. □

The lexicographic pivoting rule is straightforward to use if the simplex method is implemented in terms of the full tableau. It can also be used in conjunction with the revised simplex method, provided that the inverse basis matrix $\mathbf{B}^{-1}$ is formed explicitly (see Exercise 3.16). On the other hand, in sophisticated implementations of the revised simplex method, the matrix $\mathbf{B}^{-1}$ is never computed explicitly, and the lexicographic rule is not really suitable.

We finally note that in order to apply the lexicographic pivoting rule, an initial tableau with lexicographically positive rows is required. Let us assume that an initial tableau is available (methods for obtaining an initial tableau are discussed in the next section). We can then rename the variables so that the basic variables are the first m ones. This is equivalent to rearranging the tableau so that the first m columns of $\mathbf{B}^{-1}\mathbf{A}$ are the m unit vectors. The resulting tableau has lexicographically positive rows, as desired.

Bland's rule

The smallest subscript pivoting rule, also known as Bland's rule, is as follows.

Smallest subscript pivoting rule

1. Find the smallest j for which the reduced cost $\bar{c}_j$ is negative and have the column $\mathbf{A}_j$ enter the basis.
2. Out of all variables x_i that are tied in the test for choosing an exiting variable, select the one with the smallest value of i.

This pivoting rule is compatible with an implementation of the revised simplex method in which the reduced costs of the nonbasic variables are computed one at a time, in the natural order, until a negative one is discovered. Under this pivoting rule, it is known that cycling never occurs and the simplex method is guaranteed to terminate after a finite number of iterations.

3.5 Finding an initial basic feasible solution

In order to start the simplex method, we need to find an initial basic feasible solution. Sometimes this is straightforward. For example, suppose that we are dealing with a problem involving constraints of the form $\mathbf{Ax} \leq \mathbf{b}$, where $\mathbf{b} \geq \mathbf{0}$. We can then introduce nonnegative slack variables $\mathbf{s}$ and rewrite the constraints in the form $\mathbf{Ax} + \mathbf{s} = \mathbf{b}$. The vector $(\mathbf{x}, \mathbf{s})$ defined by $\mathbf{x} = \mathbf{0}$ and $\mathbf{s} = \mathbf{b}$ is a basic feasible solution and the corresponding basis matrix is the identity. In general, however, finding an initial basic feasible solution is not easy and requires the solution of an auxiliary linear programming problem, as will be seen shortly.

Consider the problem

$$\begin{array}{ll} \text{minimize} & \mathbf{c}'\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array}$$

By possibly multiplying some of the equality constraints by -1, we can assume, without loss of generality, that $\mathbf{b} \geq \mathbf{0}$. We now introduce a vector $y \in \Re^m$ of *artificial variables* and use the simplex method to solve the auxiliary problem

$$\begin{array}{ll} \text{minimize} & y_1 + y_2 + \cdots + y_m \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \geq \mathbf{0}. \end{array}$$

Initialization is easy for the auxiliary problem: by letting $\mathbf{x} = \mathbf{0}$ and $\mathbf{y} = \mathbf{b}$, we have a basic feasible solution and the corresponding basis matrix is the identity.

If $\mathbf{x}$ is a feasible solution to the original problem, this choice of $\mathbf{x}$ together with $\mathbf{y} = \mathbf{0}$, yields a zero cost solution to the auxiliary problem. Therefore, if the optimal cost in the auxiliary problem is nonzero, we conclude that the original problem is infeasible. If on the other hand, we obtain a zero cost solution to the auxiliary problem, it must satisfy $\mathbf{y} = \mathbf{0}$, and $\mathbf{x}$ is a feasible solution to the original problem.

At this point, we have accomplished our objectives only partially. We have a method that either detects infeasibility or finds a feasible solution to the original problem. However, in order to initialize the simplex method for the original problem, we need a basic feasible solution, an associated basis matrix $\mathbf{B}$, and – depending on the implementation – the corresponding tableau. All this is straightforward if the simplex method, applied to the auxiliary problem, terminates with a basis matrix $\mathbf{B}$ consisting exclusively of columns of $\mathbf{A}$. We can simply drop the columns that correspond to the artificial variables and continue with the simplex method on the original problem, using $\mathbf{B}$ as the starting basis matrix.

Driving artificial variables out of the basis

The situation is more complex if the original problem is feasible, the simplex method applied to the auxiliary problem terminates with a feasible solution $\mathbf{x}^*$ to the original problem, but some of the artificial variables are in the final basis. (Since the final value of the artificial variables is zero, this implies that we have a degenerate basic feasible solution to the auxiliary problem.) Let k be the number of columns of $\mathbf{A}$ that belong to the final basis ($k < m$) and, without loss of generality, assume that these are the columns $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(k)}$. (In particular, $x_{B(1)}, \ldots, x_{B(k)}$ are the only variables that can be at nonzero level.) Note that the columns $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(k)}$ must be linearly independent since they are part of a basis. Under our standard assumption that the matrix $\mathbf{A}$ has full rank, the columns of $\mathbf{A}$ span $\Re^m$, and we can choose $m - k$ additional columns $\mathbf{A}_{B(k+1)}, \ldots, \mathbf{A}_{B(m)}$ of $\mathbf{A}$, to obtain a set of m linearly independent columns, that is, a basis consisting exclusively of columns of $\mathbf{A}$. With this basis, all nonbasic components of $\mathbf{x}^*$ are at zero level, and it follows that $\mathbf{x}^*$ is the basic feasible solution associated with this new basis as well. At this point, the artificial variables and the corresponding columns of the tableau can be dropped.

The procedure we have just described is called *driving the artificial variables out of the basis*, and depends crucially on the assumption that the matrix $\mathbf{A}$ has rank m. After all, if $\mathbf{A}$ has rank less than m, constructing a basis for $\Re^m$ using the columns of $\mathbf{A}$ is impossible and there exist redundant equality constraints that must be eliminated, as described by Theorem 2.5 in Section 2.3. All of the above can be carried out mechanically, in terms of the simplex tableau, in the following manner.

Suppose that the ℓth basic variable is an artificial variable, which is in the basis at zero level. We examine the ℓth row of the tableau and find some j such that the ℓth entry of $\mathbf{B}^{-1}\mathbf{A}_j$ is nonzero. We claim that $\mathbf{A}_j$ is linearly independent from the columns $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(k)}$. To see this, note that $\mathbf{B}^{-1}\mathbf{A}_{B(i)} = \mathbf{e}_i$, $i = 1, \ldots, k$, and since $k < l$, the ℓth entry of these vectors is zero. It follows that the ℓth entry of any linear combination of the vectors $\mathbf{B}^{-1}\mathbf{A}_{B(1)}, \ldots, \mathbf{B}^{-1}\mathbf{A}_{B(k)}$ is also equal to zero. Since the ℓth entry of $\mathbf{B}^{-1}\mathbf{A}_j$ is nonzero, this vector is not a linear combination of the vectors $\mathbf{B}^{-1}\mathbf{A}_{B(1)}, \ldots, \mathbf{B}^{-1}\mathbf{A}_{B(k)}$. Equivalently, $\mathbf{A}_j$ is not a linear combination of the vectors $\mathbf{A}_{B(1)}, \ldots, \mathbf{A}_{B(k)}$, which proves our claim. We now bring $\mathbf{A}_j$ into the basis and have the ℓth basic variable exit the basis. This is accomplished in the usual manner: perform those elementary row operations that replace $\mathbf{B}^{-1}\mathbf{A}_j$ by the ℓth unit vector. The only difference from the usual mechanics of the simplex method is that the pivot element (the ℓth entry of $\mathbf{B}^{-1}\mathbf{A}_j$) could be negative. Because the ℓth basic variable was zero, adding a multiple of the ℓth row to the other rows does not change the values of the basic variables. This means that after the change of basis, we are still at the same basic feasible solution to the auxiliary problem, but we have reduced the number of basic artificial variables by one. We repeat this procedure as many times as needed until all artificial variables are driven out of the basis.

Let us now assume that the ℓth row of $\mathbf{B}^{-1}\mathbf{A}$ is zero, in which case the above described procedure fails. Note that the ℓth row of $\mathbf{B}^{-1}\mathbf{A}$ is equal to $\mathbf{g}'\mathbf{A}$, where $\mathbf{g}'$ is the ℓth row of $\mathbf{B}^{-1}$. Hence, $\mathbf{g}'\mathbf{A} = \mathbf{0}'$ for some nonzero vector $\mathbf{g}$, and the matrix $\mathbf{A}$ has linearly dependent rows. Since we are dealing with a feasible problem, we must also have $\mathbf{g}'\mathbf{b} = 0$. Thus, the constraint $\mathbf{g}'\mathbf{A}\mathbf{x} = \mathbf{g}'\mathbf{b}$ is redundant and can be eliminated (cf. Theorem 2.5 in Section 2.3). Since this constraint is the information provided by the ℓth row of the tableau, we can eliminate that row and continue from there.

Example 3.8 Consider the linear programming problem:

$$\begin{array}{lrcl} \text{minimize} & x_1 + x_2 + x_3 & & \\ \text{subject to} & x_1 + 2x_2 + 3x_3 & = & 3 \\ & -x_1 + 2x_2 + 6x_3 & = & 2 \\ & 4x_2 + 9x_3 & = & 5 \\ & 3x_3 + x_4 & = & 1 \\ & x_1, \ldots, x_4 \geq 0. & & \end{array}$$

In order to find a feasible solution, we form the auxiliary problem

$$\begin{array}{lrcl} \text{minimize} & x_5 + x_6 + x_7 + x_8 & & \\ \text{subject to} & x_1 + 2x_2 + 3x_3 + x_5 & = & 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 & = & 2 \\ & 4x_2 + 9x_3 + x_7 & = & 5 \\ & 3x_3 + x_4 + x_8 & = & 1 \\ & x_1, \ldots, x_8 \geq 0. & & \end{array}$$

A basic feasible solution to the auxiliary problem is obtained by letting $(x_5, x_6, x_7, x_8) = \mathbf{b} = (3, 2, 5, 1)$. The corresponding basis matrix is the identity. Furthermore, we have $\mathbf{c}_B = (1, 1, 1, 1)$. We evaluate the reduced cost of each one of the original variables x_i, which is $-\mathbf{c}_B'\mathbf{A}_i$, and form the initial tableau:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-11	0	-8	-21	-1	0	0	0	0
$x_5 =$	3	1	2	3	0	1	0	0	0
$x_6 =$	2	-1	2	6	0	0	1	0	0
$x_7 =$	5	0	4	9	0	0	0	1	0
$x_8 =$	1	0	0	3	1*	0	0	0	1

We bring x_4 into the basis and have x_8 exit the basis. The basis matrix $\mathbf{B}$ is still the identity and only the zeroth row of the tableau changes. We obtain:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-10	0	-8	-18	0	0	0	0	1
$x_5 =$	3	1	2	3	0	1	0	0	0
$x_6 =$	2	-1	2	6	0	0	1	0	0
$x_7 =$	5	0	4	9	0	0	0	1	0
$x_4 =$	1	0	0	3*	1	0	0	0	1

We now bring x_3 into the basis and have x_4 exit the basis. The new tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-4	0	-8	0	6	0	0	0	7
$x_5 =$	2	1	2	0	-1	1	0	0	-1
$x_6 =$	0	-1	2*	0	-2	0	1	0	-2
$x_7 =$	2	0	4	0	-3	0	0	1	-3
$x_3 =$	1/3	0	0	1	1/3	0	0	0	1/3

We now bring x_2 into the basis and x_6 exits. Note that this is a degenerate pivot with $\theta^* = 0$. The new tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-4	-4	0	0	-2	0	4	0	-1
$x_5 =$	2	2*	0	0	1	1	-1	0	1
$x_2 =$	0	$-1/2$	1	0	-1	0	1/2	0	-1
$x_7 =$	2	2	0	0	1	0	-2	1	1
$x_3 =$	1/3	0	0	1	1/3	0	0	0	1/3

We now have x_1 enter the basis and x_5 exit the basis. We obtain the following tableau:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	0	0	0	0	0	2	2	0	1
$x_1 =$	1	1	0	0	1/2	1/2	$-1/2$	0	1/2
$x_2 =$	1/2	0	1	0	$-3/4$	1/4	1/4	0	$-3/4$
$x_7 =$	0	0	0	0	0	-1	-1	1	0
$x_3 =$	1/3	0	0	1	1/3	0	0	0	1/3

Note that the cost in the auxiliary problem has dropped to zero, indicating that we have a feasible solution to the original problem. However, the artificial variable x_7 is still in the basis, at zero level. In order to obtain a basic feasible solution to the original problem, we need to drive x_7 out of the basis. Note that x_7 is the third basic variable and that the third entry of the columns $\mathbf{B}^{-1}\mathbf{A}_j$, $j = 1, \ldots, 4$, associated with the original variables, is zero. This indicates that the matrix $\mathbf{A}$ has linearly dependent rows. At this point, we remove the third row of the tableau, because it corresponds to a redundant constraint, and also remove all of the artificial variables. This leaves us with the following initial tableau for the

original problem:

		x_1	x_2	x_3	x_4
	*	*	*	*	*
$x_1 =$	1	1	0	0	1/2
$x_2 =$	1/2	0	1	0	−3/4
$x_3 =$	1/3	0	0	1	1/3

We may now compute the reduced costs of the original variables, fill in the zeroth row of the tableau, and start executing the simplex method on the original problem.

We observe that in this example, the artificial variable x_8 was unnecessary. Instead of starting with $x_8 = 1$, we could have started with $x_4 = 1$ thus eliminating the need for the first pivot. More generally, whenever there is a variable that appears in a single constraint and with a positive coefficient (slack variables being the typical example), we can always set that variable be in the initial basis and we do not have to associate an artificial variable with that constraint.

The two-phase simplex method

We can now summarize a complete algorithm for linear programming problems in standard form.

Phase I:

1. By multiplying some of the constraints by -1, change the problem so that $\mathbf{b} \geq \mathbf{0}$.
2. Introduce artificial variables $y_1, \ldots, y_m$, if necessary, and apply the simplex method to the auxiliary problem with cost $\sum_{i=1}^{m} y_i$.
3. If the optimal cost in the auxiliary problem is positive, the original problem is infeasible and the algorithm terminates.
4. If the optimal cost in the auxiliary problem is zero, a feasible solution to the original problem has been found. If no artificial variable is in the final basis, the artificial variables and the corresponding columns are eliminated, and a feasible basis for the original problem is available.
5. If the ℓth basic variable is an artificial one, examine the ℓth entry of the columns $\mathbf{B}^{-1}\mathbf{A}_j$, $j = 1, \ldots, n$. If all of these entries are zero, the ℓth row represents a redundant constraint and is eliminated. Otherwise, if the ℓth entry of the jth column is nonzero, apply a change of basis (with this entry serving as the pivot element): the ℓth basic variable exits and x_j enters the basis. Repeat this operation until all artificial variables are driven out of the basis.

Phase II:

1. Let the final basis and tableau obtained from Phase I be the initial basis and tableau for Phase II.
2. Compute the reduced costs of all variables for this initial basis, using the cost coefficients of the original problem.
3. Apply the simplex method to the original problem.

The above two-phase algorithm is a complete method, in the sense that it can handle all possible outcomes. As long as cycling is avoided (due to either nondegeneracy, an anticycling rule, or luck), one of the following possibilities will materialize:

(a) If the problem is infeasible, this is detected at the end of Phase I.

(b) If the problem is feasible but the rows of $\mathbf{A}$ are linearly dependent, this is detected and corrected at the end of Phase I, by eliminating redundant equality constraints.

(c) If the optimal cost is equal to $-\infty$, this is detected while running Phase II.

(d) Else, Phase II terminates with an optimal solution.

The big-M method

We close by mentioning an alternative approach, the *big-M method*, that combines the two phases into a single one. The idea is to introduce a cost function of the form

$$\sum_{j=1}^{n} c_j x_j + M \sum_{i=1}^{m} y_i,$$

where M is a large positive constant, and where y_i are the same artificial variables as in Phase I simplex. For a sufficiently large choice of M, if the original problem is feasible and its optimal cost is finite, all of the artificial variables are eventually driven to zero (Exercise 3.26), which takes us back to the minimization of the original cost function. In fact, there is no reason for fixing a numerical value for M. We can leave M as an undetermined parameter and let the reduced costs be functions of M. Whenever M is compared to another number (in order to determine whether a reduced cost is negative), M will be always treated as being larger.

Example 3.9 We consider the same linear programming problem as in Example 3.8:

$$\begin{array}{lrcl} \text{minimize} & x_1 + x_2 + x_3 & & \\ \text{subject to} & x_1 + 2x_2 + 3x_3 & = & 3 \\ & -x_1 + 2x_2 + 6x_3 & = & 2 \\ & 4x_2 + 9x_3 & = & 5 \\ & 3x_3 + x_4 & = & 1 \\ & x_1, \ldots, x_4 \geq 0. & & \end{array}$$

We use the big-M method in conjunction with the following auxiliary problem, in which the unnecessary artificial variable x_8 is omitted.

$$\begin{array}{lrcl} \text{minimize} & x_1 + x_2 + x_3 + Mx_5 + Mx_6 + Mx_7 & & \\ \text{subject to} & x_1 + 2x_2 + 3x_3 + x_5 & = & 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 & = & 2 \\ & 4x_2 + 9x_3 + x_7 & = & 5 \\ & 3x_3 + x_4 & = & 1 \\ & x_1, \ldots, x_7 \geq 0. & & \end{array}$$

A basic feasible solution to the auxiliary problem is obtained by letting $(x_5, x_6, x_7, x_4) = \mathbf{b} = (3, 2, 5, 1)$. The corresponding basis matrix is the identity. Furthermore, we have $\mathbf{c}_B = (M, M, M, 0)$. We evaluate the reduced cost of each one of the original variables x_i, which is $c_i - \mathbf{c}_B'\mathbf{A}_i$, and form the initial tableau:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$-10M$	1	$-8M+1$	$-18M+1$	0	0	0	0
$x_5 =$	3	1	2	3	0	1	0	0
$x_6 =$	2	-1	2	6	0	0	1	0
$x_7 =$	5	0	4	9	0	0	0	1
$x_4 =$	1	0	0	3*	1	0	0	0

The reduced cost of x_3 is negative when M is large enough. We therefore bring x_3 into the basis and have x_4 exit. Note that in order to set the reduced cost of x_3 to zero, we need to multiply the pivot row by $6M - 1/3$ and add it to the zeroth row. The new tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$-4M-1/3$	1	$-8M+1$	0	$6M-1/3$	0	0	0
$x_5 =$	2	1	2	0	-1	1	0	0
$x_6 =$	0	-1	2*	0	-2	0	1	0
$x_7 =$	2	0	4	0	-3	0	0	1
$x_3 =$	1/3	0	0	1	1/3	0	0	0

The reduced cost of x_2 is negative when M is large enough. We therefore bring x_2 into the basis and x_6 exits. Note that this is a degenerate pivot with $\theta^* = 0$.

The new tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$-4M-\frac{1}{3}$	$-4M+\frac{3}{2}$	0	0	$-2M+\frac{2}{3}$	0	$4M-\frac{1}{2}$	0
$x_5 =$	2	2*	0	0	1	1	-1	0
$x_2 =$	0	$-1/2$	1	0	-1	0	1/2	0
$x_7 =$	2	2	0	0	1	0	-2	1
$x_3 =$	1/3	0	0	1	1/3	0	0	0

We now have x_1 enter and x_5 exit the basis. We obtain the following tableau:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$-11/6$	0	0	0	$-1/12$	$2M-3/4$	$2M+1/4$	0
$x_1 =$	1	1	0	0	1/2	1/2	$-1/2$	0
$x_2 =$	1/2	0	1	0	$-3/4$	1/4	1/4	0
$x_7 =$	0	0	0	0	0	-1	-1	1
$x_3 =$	1/3	0	0	1	1/3*	0	0	0

We now bring x_4 into the basis and x_3 exits. The new tableau is:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	$-7/4$	0	0	1/4	0	$2M-3/4$	$2M+1/4$	0
$x_1 =$	1/2	1	0	$-3/2$	0	1/2	$-1/2$	0
$x_2 =$	5/4	0	1	9/4	0	1/4	1/4	0
$x_7 =$	0	0	0	0	0	-1	-1	1
$x_4 =$	1	0	0	3	1	0	0	0

With M large enough, all of the reduced costs are nonnegative and we have an optimal solution to the auxiliary problem. In addition, all of the artificial variables have been driven to zero, and we have an optimal solution to the original problem.

3.6 Column geometry and the simplex method

In this section, we introduce an alternative way of visualizing the workings of the simplex method. This approach provides some insights into why the

simplex method appears to be efficient in practice.

We consider the problem

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{e}'\mathbf{x} = 1 \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.6}$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{e}$ is the n-dimensional vector with all components equal to one. Although this might appear to be a special type of a linear programming problem, it turns out that every problem with a bounded feasible set can be brought into this form (Exercise 3.28). The constraint $\mathbf{e}'\mathbf{x} = 1$ is called the *convexity constraint*. We also introduce an auxiliary variable z defined by $z = \mathbf{c}'\mathbf{x}$. If $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ are the n columns of $\mathbf{A}$, we are dealing with the problem of minimizing z subject to the nonnegativity constraints $\mathbf{x} \geq \mathbf{0}$, the convexity constraint $\sum_{i=1}^{n} x_i = 1$, and the constraint

$$x_1 \begin{bmatrix} \mathbf{A}_1 \\ c_1 \end{bmatrix} + x_2 \begin{bmatrix} \mathbf{A}_2 \\ c_2 \end{bmatrix} + \cdots + x_n \begin{bmatrix} \mathbf{A}_n \\ c_n \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ z \end{bmatrix}.$$

In order to capture this problem geometrically, we view the horizontal plane as an m-dimensional space containing the columns of $\mathbf{A}$, and we view the vertical axis as the one-dimensional space associated with the cost components c_i. Then, each point in the resulting three-dimensional space corresponds to a point $(\mathbf{A}_i, c_i)$; see Figure 3.5.

In this geometry, our objective is to construct a vector $(\mathbf{b}, z)$, which is a convex combination of the vectors $(\mathbf{A}_i, c_i)$, such that z is as small as possible. Note that the vectors of the form $(\mathbf{b}, z)$ lie on a vertical line, which we call the *requirement line*, and which intersects the horizontal plane at $\mathbf{b}$. If the requirement line does not intersect the convex hull of the points $(\mathbf{A}_i, c_i)$, the problem is infeasible. If it does intersect it, the problem is feasible and an optimal solution corresponds to the lowest point in the intersection of the convex hull and the requirement line. For example, in Figure 3.6, the requirement line intersects the convex hull of the points $(\mathbf{A}_i, c_i)$; the point G corresponds to an optimal solution, and its height is the optimal cost.

We now need some terminology.

Definition 3.6

(a) *A collection of vectors $\mathbf{y}^1, \ldots, \mathbf{y}^{k+1}$ in $\Re^n$ are said to be* **affinely independent** *if the vectors $\mathbf{y}^1 - \mathbf{y}^{k+1}, \mathbf{y}^2 - \mathbf{y}^{k+1}, \ldots, \mathbf{y}^k - \mathbf{y}^{k+1}$ are linearly independent. (Note that we must have $k \leq n$.)*

(b) *The convex hull of $k+1$ affinely independent vectors in $\Re^n$ is called a k-dimensional* **simplex**.

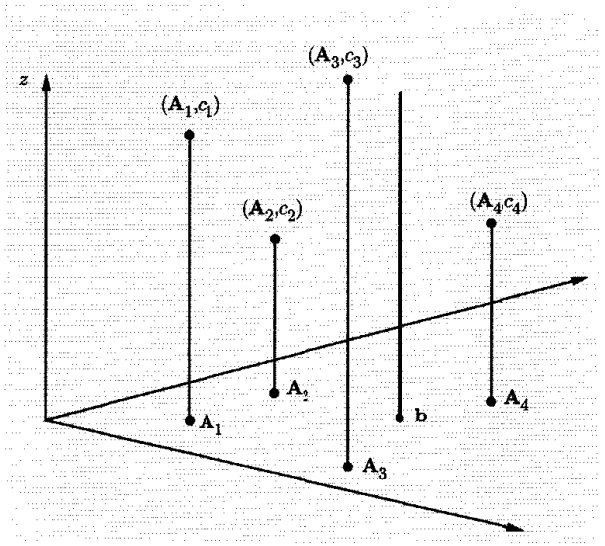

Figure 3.5: The column geometry.

Thus, three points are either collinear or they are affinely independent and determine a two-dimensional simplex (a triangle). Similarly, four points either lie on the same plane, or they are affinely independent and determine a three-dimensional simplex (a pyramid).

Let us now give an interpretation of basic feasible solutions to problem (3.6) in this geometry. Since we have added the convexity constraint, we have a total of $m+1$ equality constraints. Thus, a basic feasible solution is associated with a collection of $m+1$ linearly independent columns $(\mathbf{A}_i, 1)$ of the linear programming problem (3.6). These are in turn associated with $m+1$ of the points $(\mathbf{A}_i, c_i)$, which we call *basic points*; the remaining points $(\mathbf{A}_i, c_i)$ are called the *nonbasic points*. It is not hard to show that the $m+1$ basic points are affinely independent (Exercise 3.29) and, therefore, their convex hull is an m-dimensional simplex, which we call the *basic simplex*. Let the requirement line intersect the m-dimensional basic simplex at some point $(\mathbf{b}, z)$. The vector of weights x_i used in expressing $(\mathbf{b}, z)$ as a convex combination of the basic points, is the current basic feasible solution, and z represents its cost. For example, in Figure 3.6, the shaded triangle CDF is the basic simplex, and the point H corresponds to a basic feasible solution associated with the basic points C, D, and F.

Let us now interpret a change of basis geometrically. In a change of basis, a new point $(\mathbf{A}_j, c_j)$ becomes basic, and one of the currently basic points is to become nonbasic. For example, in Figure 3.6, if C, D, F, are the current basic points, we could make point B basic, replacing F

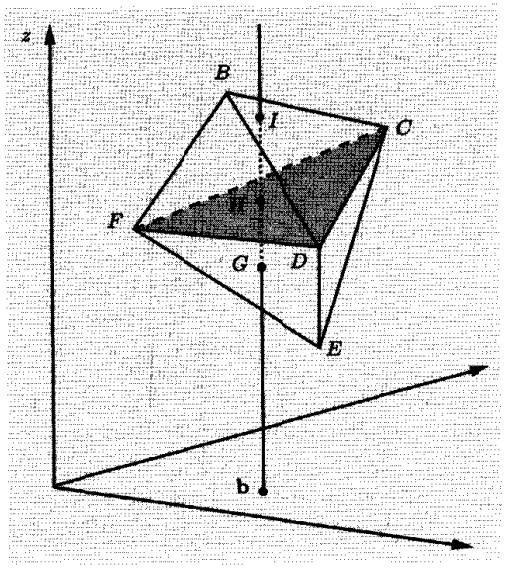

Figure 3.6: Feasibility and optimality in the column geometry.

(even though this turns out not to be profitable). The new basic simplex would be the convex hull of B, C, D, and the new basic feasible solution would correspond to point I. Alternatively, we could make point E basic, replacing C, and the new basic feasible solution would now correspond to point G. After a change of basis, the intercept of the requirement line with the new basic simplex is lower, and hence the cost decreases, if and only if the new basic point is below the plane that passes through the old basic points; we refer to the latter plane as the *dual plane*. For example, point E is below the dual plane and having it enter the basis is profitable; this is not the case for point B. In fact, the vertical distance from the dual plane to a point $(\mathbf{A}_j, c_j)$ is equal to the reduced cost of the associated variable x_j (Exercise 3.30); requiring the new basic point to be below the dual plane is therefore equivalent to requiring the entering column to have negative reduced cost.

We discuss next the selection of the basic point that will exit the basis. Each possible choice of the exiting point leads to a different basic simplex. These m basic simplices, together with the original basic simplex (before the change of basis) form the boundary (the faces) of an $(m+1)$-dimensional simplex. The requirement line exits this $(m+1)$-dimensional simplex through its top face and must therefore enter it by crossing some other face. This determines which one of the potential basic simplices will be obtained after the change of basis. In reference to Figure 3.6, the basic

points C, D, F, determine a two-dimensional basic simplex. If point E is to become basic, we obtain a three-dimensional simplex (pyramid) with vertices C, D, E, F. The requirement line exits the pyramid through its top face with vertices C, D, F. It enters the pyramid through the face with vertices D, E, F; this is the new basic simplex.

We can now visualize pivoting through the following physical analogy. Think of the original basic simplex with vertices C, D, F, as a solid object anchored at its vertices. Grasp the corner of the basic simplex at the vertex C leaving the basis, and pull the corner down to the new basic point E. While so moving, the simplex will hinge, or *pivot*, on its anchor and stretch down to the lower position. The somewhat peculiar terms (e.g., "simplex", "pivot") associated with the simplex method have their roots in this column geometry.

Example 3.10 Consider the problem illustrated in Figure 3.7, in which $m = 1$, and the following pivoting rule: choose a point $(\mathbf{A}_i, c_i)$ below the dual plane to become basic, whose vertical distance from the dual plane is largest. According to Exercise 3.30, this is identical to the pivoting rule that selects an entering variable with the most negative reduced cost. Starting from the initial basic simplex consisting of the points $(\mathbf{A}_3, c_3)$, $(\mathbf{A}_6, c_6)$, the next basic simplex is determined by the points $(\mathbf{A}_3, c_3)$, $(\mathbf{A}_5, c_5)$, and the next one by the points $(\mathbf{A}_5, c_5)$, $(\mathbf{A}_8, c_8)$. In particular, the simplex method only takes two pivots in this case. This example indicates why the simplex method may require a rather small number of pivots, even when the number of underlying variables is large.

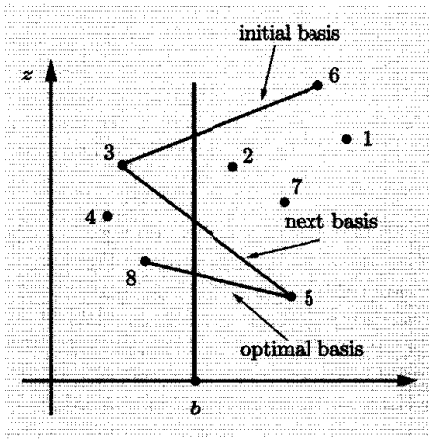

Figure 3.7: The simplex method finds the optimal basis after two iterations. Here, the point indicated by a number i corresponds to the vector $(\mathbf{A}_i, c_i)$.

required by the algorithm, when applied to a random problem drawn according to the postulated probability distribution. Unfortunately, there is no natural probability distribution over the set of linear programming problems. Nevertheless, a fair number of positive results have been obtained for a few different types of probability distributions. In one such result, a set of vectors $\mathbf{c}, \mathbf{a}_1, \ldots, \mathbf{a}_m \in \Re^n$ and scalars $b_1, \ldots, b_m$ is given. For $i = 1, \ldots, m$, we introduce either constraint $\mathbf{a}_i'\mathbf{x} \leq b_i$ or $\mathbf{a}_i'\mathbf{x} \geq b_i$, with equal probability. We then have 2^m possible linear programming problems, and suppose that L of them are feasible. Haimovich (1983) has established that under a rather special pivoting rule, the simplex method requires no more than $n/2$ iterations, on the average over those L feasible problems. This linear dependence on the size of the problem agrees with observed behavior; some empirical evidence is discussed in Chapter 12.

3.8 Summary

This chapter was centered on the development of the simplex method, which is a complete algorithm for solving linear programming problems in standard form. The cornerstones of the simplex method are:

(a) the optimality conditions (nonnegativity of the reduced costs) that allow us to test whether the current basis is optimal;

(b) a systematic method for performing basis changes whenever the optimality conditions are violated.

At a high level, the simplex method simply moves from one extreme point of the feasible set to another, each time reducing the cost, until an optimal solution is reached. However, the lower level details of the simplex method, relating to the organization of the required computations and the associated bookkeeping, play an important role. We have described three different implementations: the naive one, the revised simplex method, and the full tableau implementation. Abstractly, they are all equivalent, but their mechanics are quite different. Practical implementations of the simplex method follow our general description of the revised simplex method, but the details are different, because an explicit computation of the inverse basis matrix is usually avoided.

We have seen that degeneracy can cause substantial difficulties, including the possibility of nonterminating behavior (cycling). This is because in the presence of degeneracy, a change of basis may keep us at the same basic feasible solution, with no cost improvement resulting. Cycling can be avoided if suitable rules for choosing the entering and exiting variables (pivoting rules) are applied (e.g., Bland's rule or the lexicographic pivoting rule).

Starting the simplex method requires an initial basic feasible solution, and an associated tableau. These are provided by the Phase I simplex algorithm, which is nothing but the simplex method applied to an auxiliary

problem. We saw that the changeover from Phase I to Phase II involves some delicate steps whenever some artificial variables are in the final basis constructed by the Phase I algorithm.

The simplex method is a rather efficient algorithm and is incorporated in most of the commercial codes for linear programming. While the number of pivots can be an exponential function of the number of variables and constraints in the worst case, its observed behavior is a lot better, hence the practical usefulness of the method.

3.9 Exercises

Exercise 3.1 (Local minima of convex functions) Let $f : \Re^n \mapsto \Re$ be a convex function and let $S \subset \Re^n$ be a convex set. Let $\mathbf{x}^*$ be an element of S. Suppose that $\mathbf{x}^*$ is a local optimum for the problem of minimizing $f(\mathbf{x})$ over S; that is, there exists some $\epsilon > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$ for which $\|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon$. Prove that $\mathbf{x}^*$ is globally optimal; that is, $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$.

Exercise 3.2 (Optimality conditions) Consider the problem of minimizing $\mathbf{c}'\mathbf{x}$ over a polyhedron P. Prove the following:

(a) A feasible solution $\mathbf{x}$ is optimal if and only if $\mathbf{c}'\mathbf{d} \geq 0$ for every feasible direction $\mathbf{d}$ at $\mathbf{x}$.

(b) A feasible solution $\mathbf{x}$ is the unique optimal solution if and only if $\mathbf{c}'\mathbf{d} > 0$ for every nonzero feasible direction $\mathbf{d}$ at $\mathbf{x}$.

Exercise 3.3 Let $\mathbf{x}$ be an element of the standard form polyhedron $P = \{\mathbf{x} \in \Re^n \mid \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}$. Prove that a vector $\mathbf{d} \in \Re^n$ is a feasible direction at $\mathbf{x}$ if and only if $\mathbf{A}\mathbf{d} = \mathbf{0}$ and $d_i \geq 0$ for every i such that $x_i = 0$.

Exercise 3.4 Consider the problem of minimizing $\mathbf{c}'\mathbf{x}$ over the set $P = \{\mathbf{x} \in \Re^n \mid \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{D}\mathbf{x} \leq \mathbf{f},\ \mathbf{E}\mathbf{x} \leq \mathbf{g}\}$. Let $\mathbf{x}^*$ be an element of P that satisfies $\mathbf{D}\mathbf{x}^* = \mathbf{f}$, $\mathbf{E}\mathbf{x}^* < \mathbf{g}$. Show that the set of feasible directions at the point $\mathbf{x}^*$ is the set

$$\{\mathbf{d} \in \Re^n \mid \mathbf{A}\mathbf{d} = \mathbf{0},\ \mathbf{D}\mathbf{d} \leq \mathbf{0}\}.$$

Exercise 3.5 Let $P = \{\mathbf{x} \in \Re^3 \mid x_1 + x_2 + x_3 = 1,\ \mathbf{x} \geq \mathbf{0}\}$ and consider the vector $\mathbf{x} = (0, 0, 1)$. Find the set of feasible directions at $\mathbf{x}$.

Exercise 3.6 (Conditions for a unique optimum) Let $\mathbf{x}$ be a basic feasible solution associated with some basis matrix $\mathbf{B}$. Prove the following:

(a) If the reduced cost of every nonbasic variable is positive, then $\mathbf{x}$ is the unique optimal solution.

(b) If $\mathbf{x}$ is the unique optimal solution and is nondegenerate, then the reduced cost of every nonbasic variable is positive.