Chapter 2

Linear Algebra: A Constructive Approach

In Section 1.4 we sketched a geometric interpretation of the simplex method. In this chapter, we describe the basis of an algebraic interpretation that allows it to be implemented on a computer. The fundamental building block for the simplex method from linear algebra is the *Jordan exchange*. In this chapter, we describe the Jordan exchange and its implementation in MATLAB. We use it in a constructive derivation of several key results in linear algebra concerning linear independence and the solution of linear systems of equations. In the latter part of the chapter, we discuss the LU factorization, another linear algebra tool that is important in implementations of the simplex method.

In this chapter and the rest of the book, we assume basic familiarity with MATLAB. There are many books and web sites that will get you started in MATLAB; we recommend the MATLAB Primer by Sigmon & Davis (2004).

We first describe the Jordan exchange, a fundamental building block of linear algebra and the simplex algorithm for linear programming.

2.1 Jordan Exchange

Consider the following simple linear equation in the one-dimensional variables x and y:

$$y = ax.$$

The form of the equation indicates that x is the *independent* variable and y is the *dependent* variable: Given a value of x, the equation tells us how to determine the corresponding value of y. Thus we can think of the dependent variable as a function of the independent variable; that is, $y(x) := ax$. If we assume that $a \neq 0$, we can reverse the roles of y and x as follows:

$$x = \tilde{a}y, \qquad \text{where } \tilde{a} = \frac{1}{a}.$$

Note that now we have a function $x(y)$ in which x is determined as a function of y. This exchange in roles between dependent and independent variables gives a very simple procedure for solving either the *equation* $ax - b = 0$ or the *inequality* $ax - b \geq 0$, using the

following simple equivalences:

$$\begin{array}{lllllll} ax - b = 0 & \Longleftrightarrow & ax = y, & y = b & \Longleftrightarrow & x = \tilde{a}y, & y = b, \\ ax - b \geq 0 & \Longleftrightarrow & ax = y, & y \geq b & \Longleftrightarrow & x = \tilde{a}y, & y \geq b. \end{array} \tag{2.1}$$

In particular, the second formula gives an explicit characterization of the values of x that satisfy the inequality $ax - b \geq 0$, in terms of an independent variable y for which $y \geq b$.

The *Jordan exchange* is a generalization of the process above. It deals with the case in which $x \in \mathbf{R}^n$ is a *vector* of independent variables and $y \in \mathbf{R}^m$ is a *vector* of dependent variables, and we wish to exchange one of the independent variables with one of the dependent variables. The Jordan exchange plays a crucial role in solving systems of equations, linear inequalities, and linear programs. In addition, it can be used to derive fundamental results of linear algebra and linear programming.

We now demonstrate a Jordan exchange on the system $y = Ax$ by exchanging the roles of a component y_r of y and a component x_s of x. First, we write this system equation-wise as

$$y_i = A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n, \qquad i = 1, 2, \ldots, m, \tag{2.2}$$

where the independent variables are $x_1, x_2, \ldots, x_n$ and the dependent variables are $y_1, y_2, \ldots, y_m$, and the A_{ij}'s are the coefficients. We can think of the y_i's as (linear) functions of x_j's, that is,

$$y_i(x) := A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n, \qquad i = 1, 2, \ldots, m, \tag{2.3}$$

or, more succinctly, $y(x) := Ax$. This system can also be represented in the following *tableau* form:

$$\begin{array}{c|c|cccccc} & & x_1 & \cdots & x_s & \cdots & x_n \\ \hline y_1 & = & A_{11} & \cdots & A_{1s} & \cdots & A_{1n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_r & = & A_{r1} & \cdots & A_{rs} & \cdots & A_{rn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_m & = & A_{m1} & \cdots & A_{ms} & \cdots & A_{mn} \end{array}$$

Note that the tableau is nothing more than a compact representation of the system of equations (2.2) or the functions determining the dependent variables from the independent variables (2.3). All the operations that we perform on the tableau are just simple algebraic operations on the system of equations, rewritten to conform with the tableau representation.

We now describe the *Jordan exchange* or *pivot operation* with regard to the tableau representation. The dependent variable y_r will become independent, while x_s changes from being independent to being dependent. The process is carried out by the following three steps.

(a) Solve the rth equation

$$y_r = A_{r1}x_1 + \cdots + A_{rs}x_s + \cdots + A_{rn}x_n$$

for x_s in terms of $x_1, x_2, \ldots, x_{s-1}, y_r, x_{s+1}, \ldots, x_n$. Note that this is possible if and only if $A_{rs} \neq 0$. (A_{rs} is known as the *pivot element*, or simply the *pivot*.)

(b) Substitute for x_s in all the remaining equations.

(c) Write the new system in a new tableau form as follows:

$$\begin{array}{cc|ccccccc} & & x_1 & \cdots & x_{s-1} & y_r & x_{s+1} & \cdots & x_n \\ y_1 & = & B_{11} & \cdots & & B_{1s} & & \cdots & B_{1n} \\ \vdots & & \vdots & & & \vdots & & & \vdots \\ y_{r-1} & = & & & & & & & \\ x_s & = & B_{r1} & \cdots & & B_{rs} & & \cdots & B_{rn} \\ y_{r+1} & = & & & & & & & \\ \vdots & & \vdots & & & \vdots & & & \vdots \\ y_m & = & B_{m1} & \cdots & & B_{ms} & & \cdots & B_{mn} \end{array}$$

To determine the elements B_{ij} in terms of the elements A_{ij}, let us carry out the algebra specified by the Jordan exchange. As will be our custom in this book, we will describe and produce corresponding MATLAB m-files for the important algebraic operations that we perform. Solution of the rth equation

$$y_r = \sum_{\substack{j=1 \\ j \neq s}}^{n} A_{rj} x_j + A_{rs} x_s$$

for x_s gives

$$x_s = \frac{1}{A_{rs}} y_r + \sum_{\substack{j=1 \\ j \neq s}}^{n} \frac{-A_{rj}}{A_{rs}} x_j = B_{rs} y_r + \sum_{\substack{j=1 \\ j \neq s}}^{n} B_{rj} x_j, \tag{2.4}$$

where

$$B_{rs} = \frac{1}{A_{rs}}, \qquad B_{rj} = \frac{-A_{rj}}{A_{rs}} \qquad \forall j \neq s. \tag{2.5}$$

These formulae define the rth row $B_{r\cdot}$ of the transformed tableau. We can express them in MATLAB by first defining `J` to represent the columns of the tableau excluding the sth column, that is,

```
≫ J = [1:s-1,s+1:n];
```

and then writing

```
≫ B(r,s) = 1.0/A(r,s); B(r,J) = -A(r,J)/A(r,s);
```

This "vector index" facility is an important feature of MATLAB which enables terse coding of expressions such as those given in (2.5).

We can now proceed with part (b) of the Jordan exchange. Substituting of the expression for x_s from (2.4) in the ith equation of the tableau ($i \neq r$), we have

$$\begin{aligned} y_i &= \sum_{\substack{j=1 \\ j\neq s}}^{n} A_{ij}x_j + A_{is}\left(\frac{1}{A_{rs}}y_r + \sum_{\substack{j=1 \\ j\neq s}}^{n} \frac{-A_{rj}}{A_{rs}}x_j\right) \\ &= \sum_{\substack{j=1 \\ j\neq s}}^{n} B_{ij}x_j + B_{is}y_r, \end{aligned}$$

where

$$B_{is} = \frac{A_{is}}{A_{rs}}, \qquad B_{ij} = \left(A_{ij} - \frac{A_{is}}{A_{rs}}A_{rj}\right) = \left(A_{ij} - B_{is}A_{rj}\right) \qquad \forall i \neq r,\ j \neq s. \tag{2.6}$$

These formulae define rows $B_{i.}$, $i = 1, 2, \ldots, m, i \neq r$, of the transformed tableau. We can also write these equations succinctly in MATLAB notation by defining `J` as above and defining `I` to represent all the rows of the tableau except the rth row, that is,

```
≫ I = [1:r-1,r+1:m];
```

and writing

```
≫ B(I,s) = A(I,s)/A(r,s);
≫ B(I,J) = A(I,J) - B(I,s)*A(r,J);
```

The complete description of one step of the Jordan exchange with pivot A_{rs} is coded in `jx.m`—the function `jx` in MATLAB.

Note that we have introduced the "function" facility of MATLAB. Any function can be defined in a file of the same name as the function, but with suffix `.m`, just as the function `jx` is stored in `jx.m`. It can then be invoked from within MATLAB—either from the command window or from within other functions—by simply typing the function name together with its arguments. The following example shows a call to the function `jx`.

Example 2-1-1. Solve the following system of equations for x_1, x_2 in terms of y_1, y_2:

$$\begin{aligned} y_1 &= 2x_1 + x_2, \\ y_2 &= 3x_1 + x_2. \end{aligned}$$

Working from the MATLAB command window, one first loads the data file containing the matrix `A` of Example 2-1-1 and then invokes `jx` twice to perform the two required Jordan exchanges:

```
≫ load ex2-1-1
≫ B = jx(A,1,1)
≫ B = jx(B,2,2)
```

MATLAB file jx.m: Jordan exchange

```
function B = jx(A,r,s)
% syntax: B = jx(A,r,s)
% input: matrix A, integers r,s
% perform a Jordan exchange with pivot A(r,s)

[m,n] = size(A); B = zeros(m,n);
I = [1:r-1,r+1:m]; J = [1:s-1,s+1:n];

% update pivot row
B(r,s) = 1.0/A(r,s); B(r,J) = -A(r,J)/A(r,s);

% update pivot column
B(I,s) = A(I,s)/A(r,s);

% update remainder of tableau
B(I,J) = A(I,J)-B(I,s)*A(r,J);
return;
```

Note that we overwrite B at each step to hold the cumulative effects of the sequence of exchanges.

We now introduce a MATLAB structure that stores a complete tableau—the row and column labels corresponding to the dependent and independent variables, along with the matrix of coefficients that defines the relationship between these quantities. The `totbl` command can be used to construct a tableau as follows:

```
≫ load ex2-1-1
≫ T = totbl(A);
```

$$\begin{array}{ccc} & x_1 & x_2 \\ y_1 & 2 & 1 \\ y_2 & 3 & 1 \end{array}$$

The row labels (dependent variables) are assigned the default values y_1 and y_2 and the column labels (independent variables) the default values x_1 and x_2; other forms of the `totbl` command, discussed later, will allow the user to define their own labels. The command `tbl` can be used to print out the tableau along with its associated labels.

To perform Jordan exchanges on the tableau representation (rather than on just the matrix), we use the labeled Jordan exchange function `ljx` in place of `jx` as follows:

```
≫ T = ljx(T,1,1);
```

$$\begin{array}{ccc} & y_1 & x_2 \\ x_1 & 0.5 & -0.5 \\ y_2 & 1.5 & -0.5 \end{array}$$

```
≫ T = ljx(T,2,2);
```

$$\begin{array}{cc|cc} & & y_1 & y_2 \\ \hline x_1 & = & -1 & 1 \\ x_2 & = & 3 & -2 \end{array}$$

In addition to making the algebraic changes to the matrix, `ljx` swaps the row and column labels as required by the exchange and prints the modified tableau using `tbl`. The trailing semicolon should not be omitted after the call to `ljx` since it results in the printing of additional unnecessary information about the structure `T`. ∎

The following simple theorem provides a formal justification of the Jordan exchange formulae (2.5) and (2.6) as well as their extension to multiple pivots in succession. The result will enable us to use the Jordan exchange to give some constructive derivations of key results in linear algebra and linear programming.

Theorem 2.1.1. *Consider the linear function y defined by $y(x) := Ax$, where $A \in \mathbf{R}^{m\times n}$. After k pivots (with appropriate reordering of rows and columns) denote the initial and kth tableaus as follows:*

$$\begin{array}{cc|cc} & & x_{\mathrm{J}_1} & x_{\mathrm{J}_2} \\ \hline y_{\mathrm{I}_1} & = & A_{\mathrm{I}_1\mathrm{J}_1} & A_{\mathrm{I}_1\mathrm{J}_2} \\ y_{\mathrm{I}_2} & = & A_{\mathrm{I}_2\mathrm{J}_1} & A_{\mathrm{I}_2\mathrm{J}_2} \end{array} \qquad \begin{array}{cc|cc} & & y_{\mathrm{I}_1} & x_{\mathrm{J}_2} \\ \hline x_{\mathrm{J}_1} & = & B_{\mathrm{I}_1\mathrm{J}_1} & B_{\mathrm{I}_1\mathrm{J}_2} \\ y_{\mathrm{I}_2} & = & B_{\mathrm{I}_2\mathrm{J}_1} & B_{\mathrm{I}_2\mathrm{J}_2} \end{array}$$

Here I_1, I_2 is a partition of $\{1, 2, \ldots, m\}$ and J_1, J_2 is a partition of $\{1, 2, \ldots, n\}$, with I_1 and J_1 containing the same number of elements. Then for all values of $x \in \mathbf{R}^n$

$$\begin{aligned} x_{\mathrm{J}_1} &= B_{\mathrm{I}_1\mathrm{J}_1} y_{\mathrm{I}_1}(x) + B_{\mathrm{I}_1\mathrm{J}_2} x_{\mathrm{J}_2}, \\ y_{\mathrm{I}_2}(x) &= B_{\mathrm{I}_2\mathrm{J}_1} y_{\mathrm{I}_1}(x) + B_{\mathrm{I}_2\mathrm{J}_2} x_{\mathrm{J}_2}. \end{aligned}$$

That is, the original linear functions y satisfy the new linear relationships given by the kth tableau.

Proof. We show the result for one pivot. The result for k pivots follows by induction.

For a pivot on the (r, s) element, we have $\mathrm{I}_1 = \{r\}$, $\mathrm{I}_2 = \{1, \ldots, r-1, r+1, \ldots, m\}$, $\mathrm{J}_1 = \{s\}$, and $\mathrm{J}_2 = \{1, \ldots, s-1, s+1, \ldots, n\}$. Then for all x, we have

$$\begin{aligned} x_{\mathrm{J}_1} - B_{\mathrm{I}_1\mathrm{J}_1} y_{\mathrm{I}_1}(x) - B_{\mathrm{I}_1\mathrm{J}_2} x_{\mathrm{J}_2} &= x_s - B_{rs} y_r(x) - \sum_{\substack{j=1 \\ j\neq s}}^{n} B_{rj} x_j \\ &= x_s - \frac{1}{A_{rs}} \left(\sum_{j=1}^{n} A_{rj} x_j \right) - \sum_{\substack{j=1 \\ j\neq s}}^{n} \frac{-A_{rj}}{A_{rs}} x_j \\ &= 0 \end{aligned}$$

and

$$
\begin{aligned}
& y_{I_2}(x) - B_{I_2J_1} y_{I_1}(x) - B_{I_2J_2} x_{I_2} \\
&= \left[y_i(x) - B_{is} y_r(x) - \sum_{\substack{j=1 \\ j \neq s}}^{n} B_{ij} x_j \right], \qquad \begin{matrix} i = 1, 2, \ldots, m, \\ i \neq r \end{matrix} \\
&= \left[\sum_{j=1}^{n} A_{ij} x_j - \frac{A_{is}}{A_{rs}} \sum_{j=1}^{n} A_{rj} x_j - \sum_{\substack{j=1 \\ j \neq s}}^{n} \left(A_{ij} - \frac{A_{is} A_{rj}}{A_{rs}} \right) x_j \right], \qquad \begin{matrix} i = 1, 2, \ldots, m, \\ i \neq r \end{matrix} \\
&= 0,
\end{aligned}
$$

verifying the claims. □

The following result shows that if two tableaus have identical dependent variables for all possible values of the independent variables, then the tableau entries are also identical.

Proposition 2.1.2. *If the linear function y is defined by $y(x) = Ax$ and also by $y(x) = Bx$, then $A = B$.*

Proof. Since $(A - B)x = 0$ for all $x \in \mathbf{R}^n$, we can choose $x = I_{\cdot i}$, where $I_{\cdot i}$ is the ith column of the identity matrix. We deduce that the ith columns of A and B are identical. Since this fact is true for all $i = 1, 2, \ldots, n$, we conclude that $A = B$. □

2.2 Linear Independence

A simple geometric way to solve a system of two equations in two unknowns is to plot the corresponding lines and determine the point where they intersect. Of course, this technique fails when the lines are parallel to one another. A key idea in linear algebra is that of *linear dependence*, which is a generalization of the idea of parallel lines. Given a matrix $A \in \mathbf{R}^{m \times n}$, we may ask if any of its rows are redundant. In other words, is there a row $A_{k\cdot}$ that can be expressed as a linear combination of the other rows? That is,

$$A_{k\cdot} = \sum_{\substack{i=1 \\ i \neq k}}^{m} \lambda_i A_{i\cdot}. \tag{2.7}$$

If so, then the rows of A are said to be *linearly dependent*.

As a concrete illustration, consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 4 & 8 \\ 5 & 6 & 12 \end{bmatrix}.$$

MATLAB file bjx.m: Labeled block Jordan exchange

```
function A = bjx(A,R,S)
% syntax: B = bjx(A,R,S)
% input: tableau A, integer vectors R,S
% perform a block Jordan exchange with pivot A(R,S)

R = R(:); S = S(:);
[m,n] = size(A.val);

% setdiff(1:m,R) := {1,...,m}\R
I = setdiff(1:m,R); J = setdiff(1:n,S);

% note that values are updated in place
% update pivot column
A.val(R,S) = inv(A.val(R,S));
A.val(I,S) = A.val(I,S)*A.val(R,S);

% update remainder of tableau
A.val(I,J) = A.val(I,J)-A.val(I,S)*A.val(R,J);

% update pivot row
A.val(R,J) = -A.val(R,S)*A.val(R,J);

% now update the labels
swap = A.bas(R);
A.bas(R) = A.nonbas(S);
A.nonbas(S) = swap;

if isfield(A,'dualbas')
  swap = A.dualbas(S);
  A.dualbas(S) = A.dualnonbas(R);
  A.dualnonbas(S) = swap;
end

tbl(A);

return;
```

2.4 Exact Solution of m Equations in n Unknowns

At this stage, we have considered only square systems where the number of variables is the same as the number of equations. In optimization applications, it is more likely that the systems under consideration have different numbers of variables than equations. Such

systems pose additional concerns. For example, the system

$$x_1 + x_2 = 1$$

having one equation and two variables has infinitely many solutions, whereas the system

$$x_1 = 1, \qquad x_1 = 2$$

clearly has no solution.

Suppose we wish to solve $Ax = b$ (or determine that no solution exists), where $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, with m and n not necessarily equal. The Jordan exchange gives a simple method for solving this problem under no assumption whatsoever on A.

1. Write the system in the following tableau form:

$$\begin{array}{c|cc} & x & 1 \\ \hline y & A & -b \end{array}$$

   Our aim is to seek x and y related by this tableau *such that* $y = 0$.

2. Pivot as many of the y_i's to the top of the tableau, say y_{I_1}, until no more can be pivoted, in which case we are blocked by a tableau as follows (with row and column reordering):

$$\begin{array}{c|cc} & y_{\mathrm{I}_1} & x_{\mathrm{J}_2} & 1 \\ \hline x_{\mathrm{J}_1} & B_{\mathrm{I}_1\mathrm{J}_1} & B_{\mathrm{I}_1\mathrm{J}_2} & d_{\mathrm{I}_1} \\ y_{\mathrm{I}_2} & B_{\mathrm{I}_2\mathrm{J}_1} & 0 & d_{\mathrm{I}_2} \end{array}$$

   We now ask the question: Is it possible to find x and y related by this tableau such that $y = 0$?

3. The system is solvable if and only if $d_{\mathrm{I}_2} = 0$, since *we require* $y_{\mathrm{I}_1} = 0$ *and* $y_{\mathrm{I}_2} = 0$. When $d_{\mathrm{I}_2} = 0$, we obtain by writing out the relationships in the tableau explicitly that

$$\begin{aligned} y_{\mathrm{I}_1} &= 0, \\ y_{\mathrm{I}_2} &= B_{\mathrm{I}_2\mathrm{J}_1} y_{\mathrm{I}_1} = 0, \\ x_{\mathrm{J}_2} &\quad \text{is arbitrary}, \\ x_{\mathrm{J}_1} &= B_{\mathrm{I}_1\mathrm{J}_2} x_{\mathrm{I}_2} + d_{\mathrm{I}_1}. \end{aligned}$$

Note that m could be less than or greater than n.

Example 2-4-1. Solve the following system:

$$\begin{aligned} x_1 - x_2 + x_3 &= 2, \\ -x_1 + 2x_2 + x_3 &= 3, \\ x_1 - x_2 - x_3 &= -2. \end{aligned}$$

Remember that our technique is to pivot as many of the y_i's to the top of the tableau as possible and then check that the resulting tableau is still valid. In particular, the elements in the last column corresponding to the y_i's that we cannot pivot to the top—the subvector d_{I_2} in the notation above—should be zero.

Rewrite the problem as follows:

$$\begin{array}{rcrcrcrcl} y_1 & = & x_1 & - & x_2 & + & x_3 & - & 2, \\ y_2 & = & -x_1 & + & 2x_2 & + & x_3 & - & 3, \\ y_3 & = & x_1 & - & x_2 & - & x_3 & + & 2, \end{array}$$

and carry out the following pivot operations on its tableau representation:

```
≫ load ex2-4-1
≫ T = totbl(A,b);
```

		x_1	x_2	x_3	1
y_1	=	1	−1	1	−2
y_2	=	−1	2	1	−3
y_3	=	1	−1	−1	2

This gives rise to the following sequence of tableaus:

```
≫ T = ljx(T,1,1);
```

		y_1	x_2	x_3	1
x_1	=	1	1	−1	2
y_2	=	−1	1	2	−5
y_3	=	1	0	−2	4

```
≫ T = ljx(T,2,2);
```

		y_1	y_2	x_3	1
x_1	=	2	1	−3	7
x_2	=	1	1	−2	5
y_3	=	1	0	−2	4

```
≫ T = ljx(T,3,3);
```

		y_1	y_2	y_3	1
x_1	=	0.5	1	1.5	1
x_2	=	0	1	1	1
x_3	=	0.5	0	−0.5	2

The final solution can be read off the tableau by setting $y_1 = y_2 = y_3 = 0$. We get $x_1 = 1$, $x_2 = 1$, and $x_3 = 2$. Note that if the calculations are being carried out by hand, the columns of the tableau that are labeled with a y_i can be suppressed since their values are never needed. This can result in a significant saving in computation time, particularly if you are performing the steps by hand.

```
≫ load ex2-4-1
≫ T = ljx(T,1,1);
≫ T = delcol(T,'y1');
```

		x_2	x_3	1
x_1	=	1	−1	2
y_2	=	1	2	−5
y_3	=	0	−2	4

```
≫ T = ljx(T,2,1);
≫ T = delcol(T,'y2');
```

$$\begin{array}{cc|c} & & x_3 & 1 \\ \hline x_1 & = & -3 & 7 \\ x_2 & = & -2 & 5 \\ y_3 & = & -2 & 4 \end{array}$$

```
≫ T = ljx(T,3,1);
≫ T = delcol(T,'y3');
```

$$\begin{array}{cc|c} & & 1 \\ \hline x_1 & = & 1 \\ x_2 & = & 1 \\ x_3 & = & 2 \end{array}$$

Of course, by deleting the y_i columns, we lose access to the linear dependence relationships between these variables. ■

Exercise 2-4-2. Test yourself on the following example:

$$\begin{aligned} x_1 + x_2 + x_3 &= 1, \\ x_1 - x_2 - x_3 &= 1, \\ x_1 - x_2 + x_3 &= 3, \end{aligned}$$

which has solution $x = (1, -1, 1)$ using

```
≫ load ex2-4-2
≫ T = totbl(A,b);
≫ ...
```

Exercise 2-4-3. Solve the following systems of equations:

1.
$$\begin{array}{rcrcrcl} 2u & + & 3v & + & 3w & = & 2, \\ & & 5v & + & 7w & = & 2, \\ 6u & + & 9v & + & 8w & = & 5. \end{array}$$

2.
$$\begin{array}{rcrcrcl} u & + & 4v & + & 2w & = & -2, \\ -2u & - & 8v & + & 3w & = & 32, \\ & & v & + & w & = & 1. \end{array}$$

Example 2-4-4. Solve the following system of 3 equations in 4 unknowns:

$$\begin{array}{rcrcrcrcl} x_1 & - & x_2 & & & + & x_4 & = & 1, \\ x_1 & & & + & x_3 & & & = & 1, \\ x_1 & + & x_2 & + & 2x_3 & - & x_4 & = & 0. \end{array}$$

The data file `ex2-4-4.mat` can be loaded into MATLAB enabling the following sequence of tableaus to be constructed:

```
≫ load ex2-4-4
≫ T = totbl(A,b);
```

		x_1	x_2	x_3	x_4	1
y_1	=	1	−1	0	1	−1
y_2	=	1	0	1	0	−1
y_3	=	1	1	2	−1	0

```
≫ T =ljx(T,2,1);
```

		y_2	x_2	x_3	x_4	1
y_1	=	1	−1	−1	1	0
x_1	=	1	0	−1	0	1
y_3	=	1	1	1	−1	1

```
≫ T =ljx(T,3,2);
```

		y_2	y_3	x_3	x_4	1
y_1	=	2	−1	0	0	1
x_1	=	1	0	−1	0	1
x_2	=	−1	1	−1	1	−1

At this point we are blocked—we cannot pivot y_1 to the top of the tableau because the pivot elements corresponding to x_3 and x_4 are both zero. (It makes no sense to exchange y_1 with either y_2 or y_3 since such a move would not increase the number of y_i's at the top of the tableau.) Since the element in the final column of the row labeled y_1 is nonzero, the system has *no solution* because, whenever $y_2 = 0$ and $y_3 = 0$, it follows that $y_1 = 1$. We are unable to set $y_1 = y_2 = y_3 = 0$ in the final tableau. In fact, we have

$$y_1(x) = 2y_2(x) - y_3(x) + 1.$$

Note that the tableau also shows that the rows of the coefficient matrix A of the original linear system are related by $A_{1\cdot} = 2A_{2\cdot} - A_{3\cdot}$ and thus are linearly dependent. ∎

Example 2-4-5. We now consider a modification of Example 2-4-4 in which the right-hand side of the first equation is changed. Our system is now as follows:

$$\begin{aligned} x_1 - x_2 + x_4 &= 2, \\ x_1 + x_3 &= 1, \\ x_1 + x_2 + 2x_3 - x_4 &= 0. \end{aligned}$$

The data file `ex2-4-5.mat` can be loaded into MATLAB enabling the following sequence of tableaus to be constructed:

```
≫ load ex2-4-5
≫ T = totbl(A,b);
```

		x_1	x_2	x_3	x_4	1
y_1	=	1	−1	0	1	−2
y_2	=	1	0	1	0	−1
y_3	=	1	1	2	−1	0

```
≫ T =ljx(T,2,1);
```

		y_2	x_2	x_3	x_4	1
y_1	=	1	−1	−1	1	−1
x_1	=	1	0	−1	0	1
y_3	=	1	1	1	−1	1

```
≫ T =ljx(T,3,2);
```

		y_2	y_3	x_3	x_4	1
y_1	=	2	−1	0	0	0
x_1	=	1	0	−1	0	1
x_2	=	−1	1	−1	1	−1

This system has *infinitely many solutions* because the final column of the tableau contains a zero in the location corresponding to the y_i column label *and* some of the x_j's are still independent variables, and therefore their values can be chosen arbitrarily. Following the procedure above, we characterize the solution set by allowing x_3 and x_4 to be arbitrary and defining

$$\begin{aligned} x_1 &= -x_3 + 1, \\ x_2 &= -x_3 + x_4 - 1. \end{aligned}$$

Another way to express this result is to introduce arbitrary variables λ_1 and λ_2 (representing the arbitrary values x_3 and x_4, respectively) and writing the solution as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix} \lambda_1 + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \lambda_2. \quad \blacksquare$$

Exercise 2-4-6. Using the MATLAB function `ljx`, solve the following systems of equations by carrying out all pivot operations. If a system has no solution, give a reason. If a system has infinitely many solutions, describe the solution set as in Example 2-4-5. If the rows of the matrix are linearly dependent, write down the actual linear dependence relations.

1. $Ax = a$, where
$$A = \begin{bmatrix} 2 & -1 & 1 & 1 \\ -1 & 2 & -1 & -2 \\ 4 & 1 & 1 & -1 \end{bmatrix}, \qquad a = \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}.$$

2. $Bx = b$, where
$$B = \begin{bmatrix} 1 & -1 & 1 & 2 \\ 1 & 1 & 0 & -1 \\ 1 & -3 & 2 & 5 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}.$$

3. $Cx = c$, where
$$C = \begin{bmatrix} 1 & -1 & 1 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \\ 1 & 1 & -1 \end{bmatrix}, \qquad c = \begin{bmatrix} 3 \\ 2 \\ 2 \\ -1 \end{bmatrix}.$$

Exercise 2-4-7. Find all solutions of $Ax = b$, where

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Describe all solutions to the following set of equations and inequalities: $Ax = b$, $x_3 \geq 0$.

We end this section by deriving a fundamental theorem of linear algebra by means of the Jordan exchange. In what follows we shall refer to the initial tableau as the $y = Ax$ tableau and (if it is possible) the $x = By$ tableau as a tableau where all the y's have been pivoted to the top of the tableau.

Theorem 2.4.1. *Let $A \in \mathbf{R}^{n\times n}$. The following are equivalent:*

(a) *A is nonsingular.*

(b) $Ax = 0 \implies x = 0$, *that is,* $\ker A := \{x \mid Ax = 0\} = \{0\}$.

(c) *$Ax = b$ has a unique solution for* each $b \in \mathbf{R}^n$.

(d) *$Ax = b$ has a unique solution for* some $b \in \mathbf{R}^n$.

Proof.

(a) $\implies$ (c) By Theorem 2.3.1, A^{-1} exists and $A^{-1}b$ solves $Ax = b$ for each b. Furthermore, if x^1 and x^2 both solve $Ax = b$, then $A(x^1 - x^2) = 0$. This implies that $x^1 - x^2 = A^{-1}0 = 0$, and so solution is unique.

(c) $\implies$ (b) If we take $b = 0$, then (c) implies that $Ax = 0$ has a unique solution. Clearly, $x = 0$ is that unique solution.

(b) $\implies$ (d) (d) is a special case of (b).

(d) $\implies$ (a) We actually prove the contrapositive $\sim$(a) $\implies$ $\sim$(d). Suppose that A is singular. Then, by definition, its rows are linearly dependent, and so by the Steinitz theorem (Theorem 2.2.3), it will not be possible to pivot all the y_i's to the top of the tableau. By carrying out Jordan exchanges until we are blocked, we obtain a tableau of the form

$$\begin{array}{ccc|cc|c} & & & y_{\mathrm{I}_1} & x_{\mathrm{J}_2} & 1 \\ \hline x_{\mathrm{J}_1} & = & & B_{\mathrm{I}_1\mathrm{J}_1} & B_{\mathrm{I}_1\mathrm{J}_2} & d_{\mathrm{I}_1} \\ y_{\mathrm{I}_2} & = & & B_{\mathrm{I}_2\mathrm{J}_1} & 0 & d_{\mathrm{I}_2} \end{array}$$

(after a possible rearrangement). Since A is square, some x_j's will remain as column labels in this final tableau; that is, J_2 is not empty. If $d_{\mathrm{I}_2} = 0$, this final tableau indicates infinitely many solutions, since x_{J_2} is arbitrary. Alternatively, if $d_{\mathrm{I}_2} \neq 0$, there are no solutions. In no case is it possible to have a *unique* solution, and so (d) cannot hold. □

Exercise 2-4-8. Find matrices A, not necessarily square, for which the number of solutions to $Ax = b$ has the following properties. Explain your answers.

1. one solution, regardless of b;

Chapter 3

The Simplex Method

All linear programs can be reduced to the following *standard form*:

$$\begin{array}{ll} \min\limits_{x} & z = p'x \\ \text{subject to} & Ax \geq b, \quad x \geq 0, \end{array} \tag{3.1}$$

where $p \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, and $A \in \mathbf{R}^{m \times n}$. To create the initial tableau for the simplex method, we rewrite the problem in the following *canonical form*:

$$\begin{array}{ll} \min\limits_{x_{\mathrm{B}}, x_{\mathrm{N}}} & z = p'x_{\mathrm{N}} + 0'x_{\mathrm{B}} \\ \text{subject to} & x_{\mathrm{B}} = Ax_{\mathrm{N}} - b, \quad x_{\mathrm{B}}, x_{\mathrm{N}} \geq 0, \end{array} \tag{3.2}$$

where the index sets N and B are defined initially as $\mathrm{N} = \{1, 2, \ldots, n\}$ and $\mathrm{B} = \{n + 1, \ldots, n + m\}$. The variables $x_{n+1}, \ldots, x_{n+m}$ are introduced to represent the slack in the inequalities $Ax \geq b$ (the difference between left- and right-hand sides of these inequalities) and are called *slack variables*. We shall represent this canonical linear program by the following tableau:

$$\begin{array}{c|c|ccc|c} & & x_1 & \cdots & x_n & 1 \\ \hline x_{n+1} & = & A_{11} & \cdots & A_{1n} & -b_1 \\ \vdots & & \vdots & \ddots & \vdots & \vdots \\ x_{n+m} & = & A_{m1} & \cdots & A_{mn} & -b_m \\ \hline z & = & p_1 & \cdots & p_n & 0 \end{array} \tag{3.3}$$

In this tableau, the slack variables $x_{n+1}, \ldots, x_{n+m}$ (the variables that make up x_{B}) are the dependent variables, while the original problem variables $x_1, \ldots, x_n$ (the variables that make up x_{N}) are independent variables. It is customary in the linear programming literature to call the dependent variables *basic* and the independent variables *nonbasic*, and we will adopt this terminology for the remainder of the book. A more succinct form of the initial tableau is known as the *condensed tableau*, which is written as follows:

$$\begin{array}{c|c|cc} & & x_{\mathrm{N}} & 1 \\ \hline x_{\mathrm{B}} & = & A & -b \\ \hline z & = & p' & 0 \end{array} \tag{3.4}$$

We "read" a tableau by setting the nonbasic variables x_N to zero, thus assigning the basic variables x_B and the objective variable z the values in the last column of the tableau. The tableau above represents the point $x_N = 0$ and $x_B = -b$ (that is, $x_{n+i} = -b_i$ for $i = 1, 2, \ldots, m$), with an objective of $z = 0$. The tableau is said to be *feasible* if the values assigned to the basic variables by this procedure are nonnegative. In the above, the tableau will be feasible if $b \leq 0$.

At each iteration of the simplex method, we exchange one element between B and N, performing the corresponding Jordan exchange on the tableau representation, much as we did in Chapter 2 in solving systems of linear equations. We ensure that the tableau remains feasible at every iteration, and we try to choose the exchanged elements so that the objective function z decreases at every iteration. We continue in this fashion until either

1. a solution is found, or
2. we discover that the objective function is unbounded below on the feasible region, or
3. we determine that the feasible region is empty.

The simplex method can be examined from two viewpoints, which must be understood separately and jointly in order to fully comprehend the method:

1. an algebraic viewpoint represented by tableaus;
2. a geometric viewpoint obtained by plotting the constraints and the contours of the objective function in the space of original variables $\mathbf{R}^n$.

Later, we show that the points represented by each feasible tableau correspond to *vertices* of the feasible region.

3.1 A Simple Example

We now illustrate how the simplex method moves from a feasible tableau to an optimal tableau, one pivot at a time, by means of the following two-dimensional example.

Example 3-1-1.

$$\begin{array}{lrcrcrl} \min\limits_{x_1,x_2} & 3x_1 - 6x_2 & & & & & \\ \text{subject to} & x_1 & + & 2x_2 & \geq & -1, \\ & 2x_1 & + & x_2 & \geq & 0, \\ & x_1 & - & x_2 & \geq & -1, \\ & x_1 & - & 4x_2 & \geq & -13, \\ & -4x_1 & + & x_2 & \geq & -23, \\ & & & x_1, x_2 & \geq & 0. \end{array}$$ ∎

The first step is to add slack variables to convert the constraints into a set of general equalities combined with nonnegativity requirements on all the variables. The slacks are defined as follows:

$$\begin{array}{rcrcrcr} x_3 & = & x_1 & + & 2x_2 & + & 1, \\ x_4 & = & 2x_1 & + & x_2, & & \\ x_5 & = & x_1 & - & x_2 & + & 1, \\ x_6 & = & x_1 & - & 4x_2 & + & 13, \\ x_7 & = & -4x_1 & + & x_2 & + & 23. \end{array}$$

(When we use MATLAB to form the initial tableau, it adds the slacks automatically; there is no need to define them explicitly as above.) We formulate the initial tableau by assembling the data for the problem (that is, the matrix A and the vectors p and b) as indicated in the condensed tableau (3.4). The MATLAB command `totbl` performs this task:

```
≫ load ex3-1-1
≫ T = totbl(A,b,p);
```

		x_1	x_2	1
x_3	=	1	2	1
x_4	=	2	1	0
x_5	=	1	−1	1
x_6	=	1	−4	13
x_7	=	−4	1	23
z	=	3	−6	0

The labels associated with the original and slack variables are stored in the MATLAB structure `T`. The point represented by the tableau above can be deduced by setting the nonbasic variables x_1 and x_2 both to zero. The resulting point is feasible, since the corresponding values of the basic variables, which initially are the same as the slack variables $x_3, x_4, \ldots, x_7$, are all nonnegative. The value of the objective in this tableau, $z = 0$, is obtained from the bottom right element.

We now seek a pivot—a Jordan exchange of a basic variable with a nonbasic variable—that yields a decrease in the objective z. The first issue is to choose the nonbasic variable which is to become basic, that is, to choose a pivot column in the tableau. In allowing a nonbasic variable to become basic, we are allowing its value to possibly increase from 0 to some positive value. What effect will this increase have on z and on the dependent (basic) variables? In the given example, let us try increasing x_1 from 0. We assign x_1 the (nonnegative) value λ while holding the other nonbasic variable x_2 at zero; that is,

$$x_1 = \lambda, \qquad x_2 = 0.$$

The tableau tells us how the objective z depends on x_1 and x_2, and so for the values given above we have

$$z = 3(\lambda) - 6(0) = 3\lambda > 0 \qquad \text{for } \lambda > 0.$$

This expression tells us that z *increases* as λ increases—the opposite of what we want. Let us try instead choosing x_2 as the variable to increase, and set

$$x_1 = 0, \qquad x_2 = \lambda > 0. \tag{3.5}$$

For this choice, we have

$$z = 3(0) - 6\lambda = -6\lambda < 0 \qquad \text{for } \lambda > 0,$$

thus decreasing z, as we wished. The general rule is to choose the pivot column to have a *negative* value in the last row, as this indicates that z will decrease as the variable corresponding to that column increases away from 0. We use the term *pricing* to indicate selection of the pivot column. We call the label of the pivot column the *entering variable*, as this variable is the one that "enters" the basis at this step of the simplex method.

To determine which of the basic variables is to change places with the entering variable, we examine the effect of increasing the entering variable on each of the basic variables. Given (3.5), we have the following relationships:

$$\begin{aligned} x_3 &= 2\lambda + 1, \\ x_4 &= \lambda, \\ x_5 &= -\lambda + 1, \\ x_6 &= -4\lambda + 13, \\ x_7 &= \lambda + 23. \end{aligned}$$

Since $z = -6\lambda$, we clearly would like to make λ as large as possible to obtain the largest possible decrease in z. On the other hand, we cannot allow λ to become *too* large, as this would force some of the basic variables to become negative. By enforcing the nonnegativity restrictions on the variables above, we obtain the following restrictions on the value of λ:

$$\begin{aligned} x_3 &= 2\lambda + 1 \geq 0 &&\Longrightarrow \quad \lambda \geq -1/2, \\ x_4 &= \lambda \geq 0 &&\Longrightarrow \quad \lambda \geq 0, \\ x_5 &= -\lambda + 1 \geq 0 &&\Longrightarrow \quad \lambda \leq 1, \\ x_6 &= -4\lambda + 13 \geq 0 &&\Longrightarrow \quad \lambda \leq 13/4, \\ x_7 &= \lambda + 23 \geq 0 &&\Longrightarrow \quad \lambda \geq -23. \end{aligned}$$

We see that the largest nonnegative value that λ can take without violating any of these constraints is $\lambda = 1$. Moreover, we observe that the *blocking variable*—the one that will become negative if we increase λ above its limit of 1—is x_5. We choose the row for which x_5 is the label as the pivot row and refer to x_5 as the *leaving variable*—the one that changes from being basic to being nonbasic. The pivot row selection process just outlined is called the *ratio test*.

By setting $\lambda = 1$, we have that x_1 and x_5 are zero, while the other variables remain nonnegative. We obtain the tableau corresponding to this point by performing the Jordan exchange of the row labeled x_5 (row 3) with the column labeled x_2 (column 2). The new tableau is as follows:

```
≫ T = ljx(T,3,2);
```

		x_1	x_5	1
x_3	=	3	−2	3
x_4	=	3	−1	1
x_2	=	1	−1	1
x_6	=	−3	4	9
x_7	=	−3	−1	24
z	=	−3	6	−6

Note that z has decreased from 0 to −6.

Before proceeding with this example, let us review the procedure above for a single step of the simplex method, indicating the general rules for selecting pivot columns and rows. Given the tableau

		x_N	1
x_B	=	H	h
z	=	c'	α

(3.6)

where B represents the current set of basic variables and N represents the current set of nonbasic variables, a pivot step of the simplex method is a Jordan exchange between a basic and nonbasic variable according to the following pivot selection rules:

1. *Pricing* (selection of pivot column s): The pivot column is a column s with a negative element in the bottom row. These elements are called *reduced costs*.

2. *Ratio Test* (selection of pivot row r): The pivot row is a row r such that

$$-h_r/H_{rs} = \min_i \{-h_i/H_{is} \mid H_{is} < 0\}.$$

Note that there is considerable flexibility in selection of the pivot column, as it is often the case that many of the reduced costs are negative. One simple rule is to choose the column with the most negative reduced cost. This gives the biggest decrease in z *per unit* increase in the entering variable. However, since we cannot tell *how much* we can increase the entering variable until we perform the ratio test, it is not generally true that this choice leads to the best decrease in z on this step, among all possible pivot columns.

Returning to the example, we see that column 1, the one labeled x_1, is the only possible choice for pivot column. The ratio test indicates that row 4, labeled by x_6, should be the pivot row. We thus obtain

```
≫ T = ljx(T,4,1);
```

		x_6	x_5	1
x_3	=	−1	2	12
x_4	=	−1	3	10
x_2	=	−0.33	0.33	4
x_1	=	−0.33	1.33	3
x_7	=	1	−5	15
z	=	1	2	−15

In this tableau, all reduced costs are positive, and so the pivot column selection procedure does not identify an appropriate column. This is as it should be, because this tableau is optimal! For any other feasible point than the one indicated by this tableau, we would have $x_6 \geq 0$ and $x_5 \geq 0$, giving an objective $z = x_6 + 2x_5 - 15 \geq -15$. Hence, we cannot improve z over its current value of -15 by allowing either x_5 or x_6 to enter the basis, and so the tableau is optimal. The values of the basic variables can be read from the last column of the optimal tableau. We are particularly interested in the values of the two variables x_1 and x_2 from the original standard formulation of the problem; they are $x_1 = 3$ and $x_2 = 4$. In general, we have an *optimal tableau* when both the last column and the bottom row are nonnegative. (Note: when talking about the last row or last column, we do not include in our considerations the bottom right element of the tableau, the one indicating the current value of the objective. Its sign is irrelevant to the optimization process.)

Figure 3.1 illustrates Example 3-1-1.

The point labeled "Vertex 1" corresponds to the initial tableau, while "Vertex 2" is represented by the second tableau and "Vertex 3" is represented by the final tableau.

Exercise 3-1-2. Consider the problem

$$\begin{array}{ll} \min & z = p'x \\ \text{subject to} & Ax \geq b, \quad x \geq 0, \end{array}$$

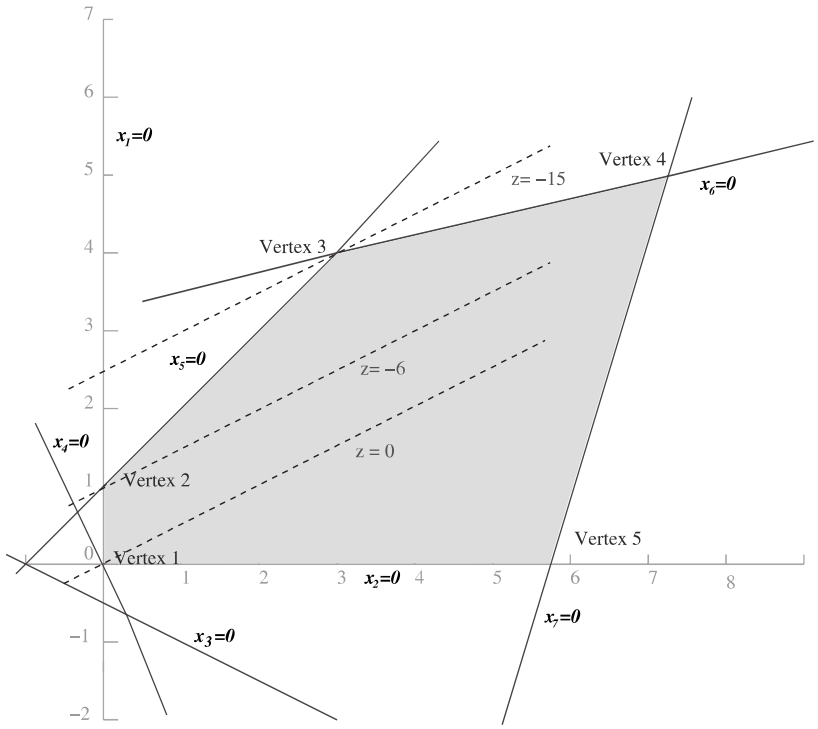

Figure 3.1. *Simplex method applied to Example* $3-1-1$.

where

$$A = \begin{bmatrix} 0 & -1 \\ -1 & -1 \\ -1 & 2 \\ 1 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} -5 \\ -9 \\ 0 \\ -3 \end{bmatrix}, \qquad p = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

(i) Draw the feasible region in $\mathbf{R}^2$.

(ii) Draw the contours of $z = -12$, $z = -14$, $z = -16$ and determine the solution graphically.

(iii) Solve the problem in MATLAB using Example 3-1-1 as a template. In addition, trace the path in contrasting color that the simplex method takes on your figure.

3.2 Vertices

The concept of a *vertex* of the feasible region plays an important role in the geometric interpretation of the simplex method. Given the feasible set for (3.1) defined by

$$\mathrm{S} := \left\{ x \in \mathbf{R}^n \mid Ax \geq b, x \geq 0 \right\}, \tag{3.7}$$

a point $x \in \mathrm{S}$ is a *vertex* of S if it is not possible to define a line segment lying entirely in S that contains x in its interior. Each vertex can be represented as the intersection of the n hyperplanes defined by $x_i = 0$, for all $i \in \mathrm{N}$, where N is the set of nonbasic variables for some feasible tableau. This representation is apparent from Figure 3.1, where for instance Vertex 2 is at the intersection of the lines defined by $x_1 = 0$ and $x_5 = 0$, where x_1 and x_5 are the two nonbasic variables in the tableau corresponding to this vertex.

Definition 3.2.1. For the feasible region S of (3.1) defined by (3.7), let $x_{n+i} := A_{i\cdot}x - b_i$, $i = 1, 2, \ldots, m$. A *vertex* of S is any point in $(x_1, x_2, \ldots, x_n)' \in \mathrm{S}$ that satisfies

$$x_{\mathrm{N}} = 0,$$

where N is any subset of $\{1, 2, \ldots, n + m\}$ containing n elements such that the linear functions defined by x_j, $j \in \mathrm{N}$, are linearly independent.

It is important for the n functions in this definition to be linearly independent. If not, then the equation $x_{\mathrm{N}} = 0$ has either zero solutions or infinitely many solutions.

Theorem 3.2.2. *Suppose that $\bar{x}$ is a vertex of* S *with corresponding index set* N. *Then if we define*

$$\mathcal{A} := [A \quad -I], \qquad \mathrm{B} := \{1, 2, \ldots, n + m\} \backslash \mathrm{N},$$

then $\bar{x}$ satisfies the relationships

$$\mathcal{A}_{\cdot\mathrm{B}}x_{\mathrm{B}} + \mathcal{A}_{\cdot\mathrm{N}}x_{\mathrm{N}} = b, \qquad x_{\mathrm{B}} \geq 0, \qquad x_{\mathrm{N}} = 0, \tag{3.8}$$

where $\mathcal{A}_{\cdot\mathrm{B}}$ is invertible. Moreover, $\bar{x}$ can be represented by a tableau of the form

$$\begin{array}{ccc} & x_{\mathrm{N}} & 1 \\ x_{\mathrm{B}} & H & h \\ z & c' & \alpha \end{array} \tag{3.9}$$

with $h \geq 0$.

Proof. By the definition of a vertex it follows that

$$\mathcal{A}_{\cdot\mathrm{B}}\bar{x}_{\mathrm{B}} + \mathcal{A}_{\cdot\mathrm{N}}\bar{x}_{\mathrm{N}} = b, \qquad \bar{x}_{\mathrm{B}} \geq 0, \qquad \bar{x}_{\mathrm{N}} = 0. \tag{3.10}$$

It remains to prove that $\mathcal{A}_{\cdot\mathrm{B}}$ is invertible. Suppose there exists a vector z such that $z'\mathcal{A}_{\cdot\mathrm{B}} = 0$. It follows from (3.10) that $z'b = 0$. By definition, the functions x_{N} satisfy

$$\mathcal{A}_{\cdot\mathrm{B}}x_{\mathrm{B}} + \mathcal{A}_{\cdot\mathrm{N}}x_{\mathrm{N}} = b, \tag{3.11}$$

and so $z'\mathcal{A}_{\cdot N}x_N = 0$. Since x_N are linearly independent, it follows that $z'\mathcal{A}_{\cdot N} = 0$ and hence that $z'\mathcal{A} = 0$. Since $\mathcal{A}$ has the (negative) identity matrix in its columns, this implies that $z = 0$, and thus $\mathcal{A}_{\cdot B}$ is invertible.

Finally, premultiplying (3.11) by $\mathcal{A}_{\cdot B}^{-1}$ and rearranging, we see that

$$x_B = -\mathcal{A}_{\cdot B}^{-1}\mathcal{A}_{\cdot N}x_N + \mathcal{A}_{\cdot B}^{-1}b, \tag{3.12}$$

which can be written in tableau form (3.9) with $H = -\mathcal{A}_{\cdot B}^{-1}\mathcal{A}_{\cdot N}$ and $h = \mathcal{A}_{\cdot B}^{-1}b$. The last row of the tableau can be generated by substituting for x_B from (3.12) in the expression $z = p_B'x_B + p_N'x_N$, obtaining $c' = p_N' - p_B'\mathcal{A}_{\cdot B}^{-1}\mathcal{A}_{\cdot N}$ and $\alpha = p_B'\mathcal{A}_{\cdot B}^{-1}b$. Note that $h \geq 0$ since $\bar{x}$ satisfies (3.12) and hence $h = \bar{x}_B$. □

The proof shows that corresponding to each vertex of S there is an invertible square matrix, which we denote by $\mathcal{A}_{\cdot B}$ above. This matrix is frequently called the *basis matrix*. It plays an important role in the revised simplex method of Chapter 5.

We illustrate this theorem by considering one of the vertices for the problem in Example 3-1-1. Here we have $n = 2$ and $m = 5$. The point $x = (0, 1)'$ is a vertex of the feasible region, occurring at the intersection of the lines defined by $x_1 = 0$ and $x_5 = 0$. The sets N and B corresponding to this vertex are therefore $N = \{1, 5\}$ and $B = \{2, 3, 4, 6, 7\}$. Thus

$$\mathcal{A}_{\cdot B} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ -4 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix}, \qquad \mathcal{A}_{\cdot N} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & -1 \\ 1 & 0 \\ -4 & 0 \end{bmatrix}$$

and simple (MATLAB) calculations show that

$$H = -\mathcal{A}_{\cdot B}^{-1}\mathcal{A}_{\cdot N} = \begin{bmatrix} 1 & -1 \\ 3 & -2 \\ 3 & -1 \\ -3 & 4 \\ -3 & -1 \end{bmatrix}, \qquad h = \mathcal{A}_{\cdot B}^{-1}b = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 9 \\ 24 \end{bmatrix}, \tag{3.13}$$

$$c' = p_N' - p_B'\mathcal{A}_{\cdot B}^{-1}\mathcal{A}_{\cdot N} = \begin{bmatrix} -3 & 6 \end{bmatrix}, \qquad \alpha = p_B'\mathcal{A}_{\cdot B}^{-1}b = -6,$$

which can be checked against the second tableau given in Example 3-1-1.

To complete our discussion of vertices, we note two more facts.

- The set N in Definition 3.2.1 that corresponds to a particular vertex may not be uniquely defined; that is, the same vertex may be specified by more than one set N. Looking at Figure 3.1 again, we see that Vertex 3 is defined by the unique set $N = \{5, 6\}$ and Vertex 5 is defined uniquely by $N = \{2, 7\}$. However, Vertex 1 can be specified by three possible choices of N: $N = \{1, 2\}$, $N = \{1, 4\}$, or $N = \{2, 4\}$. A vertex that can be specified by more than one set N is sometimes called a *degenerate vertex*.

- Given any set $N \subset \{1, 2, \ldots, n+m\}$ of n indices such that x_N is linearly independent, the point defined by $x_N = 0$ is *not* necessarily a vertex. This is because the point may lie outside the feasible region S. In Figure 3.1, for instance, the point defined by $N = \{3, 5\}$ is not a vertex, since the lines $x_3 = 0$ and $x_5 = 0$ intersect outside the feasible region.

Exercise 3-2-1. Consider the feasible set defined by the following constraints:

$$\begin{array}{rcrcr} x_1 & + & 2x_2 & \geq & 2, \\ -2x_1 & - & x_2 & \geq & -4, \\ & & x_1, x_2 & \geq & 0. \end{array}$$

1. Plot the feasible region, indicating the vertices on your plot. Indicate all possible choices for the N associated with each vertex.

2. For the vertex $x = (0, 4)'$, write down the sets N and B defined in Theorem 3.2.2, and calculate the quantities H and h in the tableau (3.9) from the formulae (3.13). (Do not use a Jordan exchange.)

3.3 The Phase II Procedure

The simplex method is generally split into two phases. *Phase I* finds a starting point that satisfies the constraints. That is, it finds a tableau of the general form (3.6) such that the last column h is nonnegative. *Phase II* starts with a feasible tableau and applies the pivots needed to move to an optimal tableau, thus solving the linear program.

In the following high-level description of the simplex method, $\mathrm{B}(r)$ denotes the label on x corresponding to the rth row of the tableau, and $\mathrm{N}(s)$ denotes the label on x corresponding to the sth column of the tableau.

Algorithm 3.1 (Simplex Method).

1. *Construct an initial tableau. If the problem is in standard form* (3.1)*, this process amounts to simply adding slack variables.*

2. *If the tableau is not feasible, apply a Phase I procedure to generate a feasible tableau, if one exists (see Section* 3.4*). For now we shall assume the origin $x_{\mathrm{N}} = 0$ is feasible.*

3. *Use the pricing rule to determine the pivot column s. If none exists,* **stop**; (a)*: tableau is optimal.*

4. *Use the ratio test to determine the pivot row r. If none exists,* **stop**; (b)*: tableau is unbounded.*

5. *Exchange $x_{\mathrm{B}(r)}$ and $x_{\mathrm{N}(s)}$ using a Jordan exchange on H_{rs}.*

6. *Go to Step* 3.

Phase II comprises Steps 3 through 6 of the method above—that part of the algorithm that occurs after an initial feasible tableau has been identified.

The method terminates in one of two ways. Stop (a) indicates optimality. This occurs when the last row is nonnegative. In this case, there is no benefit to be obtained by letting any of the nonbasic variables x_{N} increase away from zero. We can verify this claim mathematically by writing out the last row of the tableau, which indicates that the objective function is

$$z = c' x_{\mathrm{N}} + \alpha.$$

When $c \geq 0$ and $x_N \geq 0$, we have $z \geq \alpha$. Therefore, the point corresponding to the tableau (3.6)—$x_B = h$ and $x_N = 0$—is optimal, with objective function value α.

The second way that the method above terminates, Stop (b), occurs when a column with a negative cost c_s has been identified, but the ratio test fails to identify a pivot row. This situation can occur only when all the entries in the pivot column $H_{\cdot s}$ are nonnegative. In this case, by allowing $x_{N(s)}$ to grow larger, without limit, we will be decreasing the objective function to $-\infty$ without violating feasibility. In other words, by setting $x_{N(s)} = \lambda$ for any positive value λ, we have for the basic variables x_B that

$$x_B(\lambda) = H_{\cdot s}\lambda + h \geq h \geq 0,$$

so that the full set of variables $x(\lambda) \in \mathbf{R}^{m+n}$ is defined by the formula

$$x_j(\lambda) = \begin{cases} \lambda & \text{if } j = N(s), \\ H_{is}\lambda + h_i & \text{if } j = B(i), \\ 0 & \text{if } j \in N \setminus \{N(s)\}, \end{cases}$$

which is feasible for all $\lambda \geq 0$. The objective function for $x(\lambda)$ is

$$z = c' x_N(\lambda) + \alpha = c_s\lambda + \alpha,$$

which tends to $-\infty$ as $\lambda \to \infty$. Thus the set of points $x(\lambda)$ for $\lambda \geq 0$ identifies a ray of feasible points along which the objective function approaches $-\infty$. Another way to write this ray is to separate $x(\lambda)$ into a constant vector u and another vector that depends on λ as follows:

$$x(\lambda) = u + \lambda v,$$

where the elements of u and v are defined as follows:

$$u_j = \begin{cases} 0 & \text{if } j \in N, \\ h_i & \text{if } j = B(i), \end{cases} \qquad v_j = \begin{cases} 1 & \text{if } j = N(s), \\ H_{is} & \text{if } j = B(i), \\ 0 & \text{if } j \in N \setminus \{N(s)\}. \end{cases}$$

Example 3-3-1. Show that the following linear program is unbounded below, and find vectors u and v such that $u + \lambda v$ is feasible for all $\lambda \geq 0$. Find a feasible point of this form with objective value -98.

$$\begin{array}{lrcrcrcl} \text{min} & z = -2x_1 - 3x_2 + x_3 & & & & & & \\ \text{subject to} & x_1 & + & x_2 & + & x_3 & \geq & -3, \\ & -x_1 & + & x_2 & - & x_3 & \geq & -4, \\ & x_1 & - & x_2 & - & 2x_3 & \geq & -1, \\ & & & & x_1, x_2, x_3 & & \geq & 0. \end{array}$$

```
≫ load ex3-3-1
≫ T = totbl(A,b,p);
```

		x_1	x_2	x_3	1
x_4	=	1	1	1	3
x_5	=	−1	1	−1	4
x_6	=	1	−1	−2	1
z	=	−2	−3	1	0

Since this tableau is feasible, we can start immediately on Phase II (Step 3 of the algorithm above). Selecting column 2 as the pivot column, we find that the ratio test chooses row 3 as the pivot row.

```
≫ T = ljx(T,3,2);
```

		x_1	x_6	x_3	1
x_4	=	2	−1	−1	4
x_5	=	0	−1	−3	5
x_2	=	1	−1	−2	1
z	=	−5	3	7	−3

The next pivot column selected must be column 1, but we find that the ratio test fails to select a pivot row, since all the elements in column 1 (except, of course, in the last row) are nonnegative. The tableau is unbounded, and the method therefore terminates at Step 4. By setting $x_1 = \lambda \geq 0$, we have from the tableau that the dependent variables satisfy

$$x_4(\lambda) = 2\lambda + 4, \qquad x_5(\lambda) = 5, \qquad x_2(\lambda) = \lambda + 1,$$

whereas the nonbasic variables x_6 and x_3 are both 0. For the original three variables of the problem x_1, x_2, x_3, we have

$$x(\lambda) = \begin{bmatrix} \lambda \\ \lambda + 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \lambda \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},$$

so that by setting $u = (0, 1, 0)'$ and $v = (1, 1, 0)'$ we obtain the direction of unboundedness in the specified form. From the final tableau, we also have $z = -5\lambda - 3$, so that the value $z = -98$ is obtained by setting $\lambda = 19$. The corresponding value of x is then

$$x = u + 19v = \begin{bmatrix} 19 \\ 20 \\ 0 \end{bmatrix}. \quad \blacksquare$$

Exercise 3-3-2. Solve the following linear program. If it is unbounded below, find vectors u and v such that $u + \lambda v$ is feasible for all $\lambda \geq 0$. Find a feasible point of this form with objective value −415.

$$\begin{array}{lrcrcrcrcl}
\min & z = x_1 & - & 2x_2 & - & 4x_3 & + & 4x_4 & & \\
\text{subject to} & & & x_2 & - & 2x_3 & - & x_4 & \geq & -4, \\
& 2x_1 & - & x_2 & - & x_3 & + & 4x_4 & \geq & -5, \\
& -x_1 & + & x_2 & & & - & 2x_4 & \geq & -3, \\
& & & & & x_1, x_2, x_3, x_4 & & & \geq & 0.
\end{array}$$

Linear programs may have more than one solution. In fact, given any collection of solutions $x^1, x^2, \ldots, x^K$, any other point in the *convex hull* of these solutions, defined by

$$\left\{ x \,|\, x = \sum_{i=1}^{K} \alpha_i x^i, \ \sum_{i=1}^{K} \alpha_i = 1, \ \alpha_i \geq 0, \ i = 1, 2, \ldots, K \right\}, \tag{3.14}$$

is also a solution. To prove this claim, we need to verify that any such x is feasible with respect to the constraints $Ax \geq b, x \geq 0$ in (3.1) and also that x achieves the same objective value as each of the solutions x^i, $i = 1, 2, \ldots, K$. First, note that

$$Ax = \sum_{i=1}^{K} \alpha_i A x^i \geq \left(\sum_{i=1}^{K} \alpha_i \right) b = b,$$

and so the inequality constraint is satisfied. Since x is a nonnegative combination of the nonnegative vectors x, it is also nonnegative, and so the constraint $x \geq 0$ is also satisfied. Finally, since each x^i is a solution, we have that $p'x^i = z$ for some scalar z_{opt} and all $i = 1, 2, \ldots, K$. Hence,

$$p'x = \sum_{i=1}^{K} \alpha_i p' x^i = \left(\sum_{i=1}^{K} \alpha_i \right) z_{\text{opt}} = z_{\text{opt}}.$$

Since x is feasible for (3.1) and attains the optimal objective value z_{opt}, we conclude that x is a solution, as claimed.

Phase II can be extended to identify multiple solutions by performing additional pivots on columns with zero reduced costs after an optimal tableau has been identified. We illustrate the technique with the following simple example.

Example 3-3-3.

$$\begin{array}{lrcrcrcl} \min\limits_{x_1,x_2,x_3} & -x_1 & - & x_2 & - & x_3 & & \\ \text{subject to} & x_1 & - & x_2 & + & x_3 & \geq & -2, \\ & -x_1 & + & x_2 & + & x_3 & \geq & -3, \\ & x_1 & + & x_2 & - & x_3 & \geq & -1, \\ & -x_1 & - & x_2 & - & x_3 & \geq & -4, \\ & & & x_1, x_2, x_3 & & & \geq & 0. \end{array}$$

```
≫ load ex3-3-3
≫ T = totbl(A,b,p);
```

		x_1	x_2	x_3	1
x_4	=	1	−1	1	2
x_5	=	−1	1	1	3
x_6	=	1	1	−1	1
x_7	=	−1	−1	−1	4
z	=	−1	−1	−1	0

Since the problem is in standard form, the initial tableau can be created using `totbl`. The right-hand column is nonnegative, and so the resulting tableau is feasible, and no Phase I procedure is needed.

```
≫ T = ljx(T,3,3);
```

		x_1	x_2	x_6	1
x_4	=	2	0	−1	3
x_5	=	0	2	−1	4
x_3	=	1	1	−1	1
x_7	=	−2	−2	1	3
z	=	−2	−2	1	−1

```
≫ T = ljx(T,4,1);
```

		x_7	x_2	x_6	1
x_4	=	−1	−2	0	6
x_5	=	0	2	−1	4
x_3	=	−0.5	0	−0.5	2.5
x_1	=	−0.5	−1	0.5	1.5
z	=	1	0	0	−4

At this point we have found a solution to the problem, namely,

$$x = \begin{bmatrix} 1.5 \\ 0 \\ 2.5 \end{bmatrix}.$$

However, it is interesting to note that this is not the only possible solution. If we had chosen a different sequence of pivots, we would have obtained a different solution (with, of course, the same objective value $z = -4$), as we now show.

```
≫ load ex3-3-3
≫ T = totbl(A,b,p);
≫ T = ljx(T,2,1);
```

		x_5	x_2	x_3	1
x_4	=	−1	0	2	5
x_1	=	−1	1	1	3
x_6	=	−1	2	0	4
x_7	=	1	−2	−2	1
z	=	1	−2	−2	−3

```
≫ T = ljx(T,4,3);
```

		x_5	x_2	x_7	1
x_4	=	0	−2	−1	6
x_1	=	−0.5	0	−0.5	3.5
x_6	=	−1	2	0	4
x_3	=	0.5	−1	−0.5	0.5
z	=	0	0	1	−4

giving the solution

$$x = \begin{bmatrix} 3.5 \\ 0 \\ 0.5 \end{bmatrix}.$$

We can find additional solutions by performing pivots on the columns for which there is a zero in the last row. If we were to take the last tableau above and choose the first column as a pivot column, the ratio test would select row 3 as the pivot row, and the resulting pivot would yield the tableau for the solution that we found first. If, on the other hand, we were to choose the second column as the pivot column, the ratio test would select row 4 as the pivot row as follows:

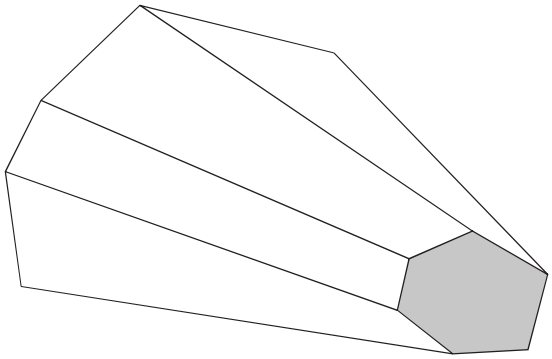

Figure 3.2. *Feasible set in* $\mathbf{R}^3$*, showing an optimal face that contains* 6 *vertices .*

```
≫ T = ljx(T,4,2);
```

		x_5	x_3	x_7	1
x_4	=	−1	2	0	5
x_1	=	−0.5	0	−0.5	3.5
x_6	=	0	−2	−1	5
x_2	=	0.5	−1	−0.5	0.5
z	=	0	0	1	−4

yielding the solution $x = (3.5, 0.5, 0)'$. Continuing this process, if we were to choose the second column as the pivot column again, we would return to the previous tableau and hence recover the previous solution. If, on the other hand, we were to select the first column, then the ratio test would select row 1 as the pivot row, and we would obtain

```
≫ T = ljx(T,1,1);
```

		x_4	x_3	x_7	1
x_5	=	−1	2	0	5
x_1	=	0.5	−1	−0.5	1
x_6	=	0	−2	−1	5
x_2	=	−0.5	0	−0.5	3
z	=	0	0	1	−4

yielding another solution $x = (1, 3, 0)'$. Proceeding in this fashion, taking care not to double back to a previous solution by choosing the same pivot column twice, we can identify two more distinct solutions: $x = (0, 3, 1)'$ and $x = (3.5, 0, 0.5)'$. These six solutions represent the vertices of an *optimal face*, defined by the convex hull (3.14), like the one illustrated in Figure 3.2. Every point in the optimal face can be associated with a particular choice of coefficients α_i in the formula (3.14).

Note that an algebraic description of the solution set can be derived from any optimal tableau. A solution is a feasible point whose objective value is equal to the optimal value. The equations represented in the tableau, along with $x_i \geq 0$, determine feasibility, while the last row determines the objective value. For example, in the optimal tableau given above,

the last row represents the equation

$$z = x_7 - 4.$$

Since $x_7 \geq 0$, it follows that in any solution, we have $x_7 = 0$. The feasibility constraints then amount to $x_3 = \lambda \geq 0$, $x_4 = \mu \geq 0$, and

$$\begin{array}{rcrcrcr} 0 \leq x_5 & = & -\mu & + & 2\lambda & + & 5, \\ 0 \leq x_1 & = & 0.5\mu & - & \lambda & + & 1, \\ 0 \leq x_6 & = & & - & 2\lambda & + & 5, \\ 0 \leq x_2 & = & -0.5\mu & & & + & 3. \end{array}$$

Thus the solution set in this example is

$$\left\{x = (0.5\mu - \lambda + 1, 3 - 0.5\mu, \lambda)' \geq 0 \mid \lambda, \mu \geq 0, 2\lambda + 5 \geq \mu, 2\lambda \leq 5\right\}.$$

Note that the solutions generated by each of the six optimal tableaus are all in this set (for particular choice of λ and μ). For concreteness, note that the first optimal tableau found generates the solution given by $\lambda = 2.5$, $\mu = 6$. ∎

Exercise 3-3-4. Solve the following problem using MATLAB:

$$\begin{array}{ll} \min & z = p'x \\ \text{subject to} & Ax \geq b, \quad x \geq 0, \end{array}$$

where

$$A = -\begin{bmatrix} 1 & 3 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}, \qquad b = -\begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix}, \qquad p = -\begin{bmatrix} 2 \\ 4 \\ 1 \\ 1 \end{bmatrix}.$$

Exercise 3-3-5. 1. Use the simplex procedure to solve

$$\begin{array}{lrcrcr} \min & z = x - y & & & & \\ \text{subject to} & -x & + & y & \geq & -2, \\ & -x & - & y & \geq & -6, \\ & & & x, y & \geq & 0. \end{array}$$

2. Draw a graphical representation of the problem in x, y space and indicate the path of the simplex steps.

3. Repeat the problem above but using the new objective function $z = -x + y$. This problem has multiple solutions, and so find all the vertex solutions and write down an expression for the full set of solutions.

4. Solve the following problem, and graph the path followed by the simplex method:

$$\begin{array}{lrcrcr} \min & z = -x - y & & & & \\ \text{subject to} & 2x & - & y & \geq & -1, \\ & -x & + & y & \geq & -1, \\ & & & x, y & \geq & 0. \end{array}$$

Exercise 3-3-6. Solve the following linear program. Is the feasible region unbounded? Is the solution set unbounded? If so, find vectors u and v such that $u + \lambda v$ is optimal for all $\lambda \geq 0$.

$$\begin{array}{lrcrcrcrcl} \min & z = 3x_1 & - & 3x_2 & - & 2x_3 & + & 5x_4 & & \\ \text{subject to} & x_1 & - & x_2 & - & x_3 & - & 0.5x_4 & \geq & -2, \\ & 2.5x_1 & + & 2.25x_2 & - & 0.5x_3 & + & 4.25x_4 & \geq & -4, \\ & -x_1 & + & x_2 & & & - & 2x_4 & \geq & -3, \\ & & & & & & & x_1, x_2, x_3, x_4 & \geq & 0. \end{array}$$

Exercise 3-3-7. Consider the linear program in three variables given by

$$\begin{array}{ll} \min & z = p'x \\ \text{subject to} & x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 1. \end{array}$$

1. Is the feasible region unbounded?

2. When $p = [1\ 1\ 1]'$, does the problem have a solution? Is the solution unique?

3. When $p = [1\ 0\ 1]'$, does the problem have a solution? Is the solution unique? If not, determine the solution set, and indicate whether the solution set is an unbounded set or not.

4. When $p = [-1\ 0\ 1]'$, does the problem have a solution? Is the solution unique?

In all three cases, indicate what properties the final tableau generated by an application of the simplex method to the problem would have. Can you construct an example of a linear program that has multiple solutions but such that the solution set is a bounded set?

3.4 The Phase I Procedure

In all the problems we have examined to date, the linear program has been stated in standard form, and the tableau constructed from the problem data has been feasible. This situation occurs when $x = 0$ is feasible with respect to the constraints, that is, when the right-hand side of all the inequality constraints is nonpositive. In general, however, this need not be the case, and we are often faced with the task of identifying a feasible initial point (that is, a feasible tableau), so that we can go ahead and apply the Phase II procedure described in Section 3.3. The process of identifying an initial feasible tableau is called *Phase I*.

Phase I entails the solution of a linear program that is different from, though closely related to, the problem we actually wish to solve. It is easy to identify an initial feasible tableau for the modified problem, and its eventual solution tells us whether the *original* problem has a feasible tableau or not. If the original problem has a feasible tableau, it can be easily constructed from the tableau resulting from Phase I.

The Phase I problem contains one additional variable x_0, a set of constraints that is the same as the original problem except for the addition of x_0 to some of them, and an objective

function of x_0 itself. It can be stated as follows:

$$\begin{array}{lll} \min\limits_{x_0,x} & z_0 = x_0 & \\ \text{subject to} & x_{n+i} = A_{i\cdot}x - b_i + x_0 & \text{if } b_i > 0, \\ & x_{n+i} = A_{i\cdot}x - b_i & \text{if } b_i \leq 0, \\ & x_0, x \geq 0. & \end{array} \tag{3.15}$$

The variable x_0 is an *artificial variable*. Note that the objective of (3.15) is bounded below by 0, since x_0 is constrained to be nonnegative. We note a number of important facts about this problem:

- We can obtain a feasible point for (3.15) by setting $x_0 = \max\left(\max_{1\leq i\leq m} b_i, 0\right)$ and $x_{\mathrm{N}} = 0$ for $\mathrm{N} = \{1, \ldots, n\}$. The dependent variables x_{B}, where $\mathrm{B} = \{n+1, n+2, \ldots, n+m\}$, then take the following initial values:

  $$\begin{aligned} b_i > 0 &\implies x_{n+i} = A_{i\cdot}x - b_i + x_0 = -b_i + \max_{1\leq j\leq m,\ b_j>0} b_j \geq -b_i + b_i = 0, \\ b_i \leq 0 &\implies x_{n+i} = A_{i\cdot}x - b_i = -b_i \geq 0, \end{aligned}$$

  so that $x_{\mathrm{B}} \geq 0$, and these components are also feasible.

- If there exists a point $\bar{x}$ that is feasible for the original problem, then the point $(x_0, x) = (0, \bar{x})$ is feasible for the Phase I problem. (It is easy to check this fact by verifying that $x_{n+i} \geq 0$ for $i = 1, 2, \ldots, m$.)

- Conversely, if $(0, x)$ is a solution of the Phase I problem, then x is feasible for the original problem. We see this by examining the constraint set for the Phase I problem and noting that

  $$\begin{aligned} b_i > 0 &\implies 0 \leq x_{n+i} = A_{i\cdot}x - b_i + x_0 = A_{i\cdot}x - b_i, \\ b_i \leq 0 &\implies 0 \leq x_{n+i} = A_{i\cdot}x - b_i, \end{aligned}$$

  so that $Ax \geq b$.

- If (x_0, x) is a solution of the Phase I problem and x_0 is *strictly positive*, then the original problem must be infeasible. This fact follows immediately from the observations above: If the original problem were feasible, it would be possible to find a feasible point for the Phase I problem with objective zero.

We can set up this starting point by forming the initial tableau for (3.15) in the usual way and performing a "special pivot." We select the x_0 column as the pivot column and choose the pivot row to be a row with the most negative entry in the last column of the tableau.

After the special pivot, the tableau contains only nonnegative entries in its last column, and the simplex method can proceed, using the usual rules for pivot column and row selection. Since the objective of (3.15) is bounded below (by zero), it can terminate only at an optimal tableau. Two possibilities then arise.

- The optimal objective z_0 is *strictly positive*. In this case, we conclude that the original problem (3.1) is *infeasible*, and so we terminate without going to Phase II.

- The optimal objective z_0 is *zero*. In this case, x_0 must also be zero, and the remaining components of x are a feasible initial point for the original problem. We can construct a feasible table for the initial problem from the optimal tableau for the Phase I problem as follows. First, if x_0 is still a dependent variable in the tableau (that is, one of the row labels), perform a Jordan exchange to make it an independent variable. (Since $x_0 = 0$, this pivot will be a degenerate pivot, and the values of the other variables will not change.) Next, delete the column labeled by x_0 and the row labeled by z_0 from the tableau. The tableau that remains is feasible for the original problem (3.1), and we can proceed with Phase II, as described in Section 3.3.

We summarize Phase I and then proceed with an example of the MATLAB implementation.

Algorithm 3.2 (Phase I).

1. *If $b \leq 0$, then $x_B = -b$, $x_N = 0$ is a feasible point corresponding to the initial tableau and no Phase I is required. Skip to Phase II.*

2. *If $b \not\leq 0$, introduce the artificial variable x_0 (and objective function $z_0 = x_0$) and set up the Phase I problem* (3.15) *and the corresponding tableau.*

3. *Perform the "special pivot" of the x_0 column with a row corresponding to the most negative entry of the last column to obtain a feasible tableau for Phase I.*

4. *Apply standard simplex pivot rules until an optimal tableau for the Phase I problem is attained. If the optimal value (for z_0) is positive,* **stop***: The original problem has no feasible point. Otherwise, perform an extra pivot (if needed) to move x_0 to the top of the tableau.*

5. *Strike out the column corresponding to x_0 and the row corresponding to z_0 and proceed to Phase II.*

The following example shows how to perform the two-phase simplex method.

Example 3-4-1.

$$\begin{array}{lrcrcr} \min & 4x_1 + 5x_2 & & & & \\ \text{subject to} & x_1 & + & x_2 & \geq & -1, \\ & x_1 & + & 2x_2 & \geq & 1, \\ & 4x_1 & + & 2x_2 & \geq & 8, \\ & -x_1 & - & x_2 & \geq & -3, \\ & -x_1 & + & x_2 & \geq & 1, \\ & & & x_1, x_2 & \geq & 0. \end{array}$$

We start by loading the data into a tableau and then adding a column for the artificial variable x_0 and the Phase I objective z_0. We use the MATLAB routines `addrow` and `addcol`. The last argument of each of these routines specifies the position that we wish the new row/column to occupy within the augmented tableau.

```
≫ load ex3-4-1
≫ T = totbl(A,b,p);
≫ neg = [0 1 1 0 1 0]';
≫ T = addcol(T,neg,'x0',3);
≫ T = addrow(T,[0 0 1 0],'z0',7);
```

		x_1	x_2	x_0	1
x_3	=	1	1	0	1
x_4	=	1	2	1	−1
x_5	=	4	2	1	−8
x_6	=	−1	−1	0	3
x_7	=	−1	1	1	−1
z	=	4	5	0	0
z_0	=	0	0	1	0

We now perform the "special" pivot, using the MATLAB `max` command to identify the largest element in the vector b (which corresponds to the most negative element in the final column of the tableau). This command also returns the position `r` occupied by this largest element. The variable `s` denotes the column position occupied by the x_0 column.

```
≫ [maxviol,r] = max(b);
≫ s = length(p)+1;
≫ T = ljx(T,r,s);
```

		x_1	x_2	x_5	1
x_3	=	1	1	0	1
x_4	=	−3	0	1	7
x_0	=	−4	−2	1	8
x_6	=	−1	−1	0	3
x_7	=	−5	−1	1	7
z	=	4	5	0	0
z_0	=	−4	−2	1	8

We now proceed with the simplex method, using the usual rules for selecting pivot columns and rows. Note that we use the last row in the tableau—the z_0 row—to select the pivot columns, since it is z_0 (not z) that defines the objective for Phase I. However, we modify the entries in row z along with the rest of the tableau.

```
≫ T = ljx(T,4,2);
```

		x_1	x_6	x_5	1
x_3	=	0	−1	0	4
x_4	=	−3	0	1	7
x_0	=	−2	2	1	2
x_2	=	−1	−1	0	3
x_7	=	−4	1	1	4
z	=	−1	−5	0	15
z_0	=	−2	2	1	2

```
≫ T = ljx(T,3,1);
```

		x_0	x_6	x_5	1
x_3	=	0	−1	0	4
x_4	=	1.5	−3	−0.5	4
x_1	=	−0.5	1	0.5	1
x_2	=	0.5	−2	−0.5	2
x_7	=	2	−3	−1	0
z	=	0.5	−6	−0.5	14
z_0	=	1	0	0	0

At this point, we have solved Phase I and x_0 appears again at the top of the tableau. We now delete the x_0 column and the z_0 row to obtain the following feasible tableau for Phase II:

```
≫ T = delrow(T,'z0');
≫ T = delcol(T,'x0');
```

		x_6	x_5	1
x_3	=	−1	0	4
x_4	=	−3	−0.5	4
x_1	=	1	0.5	1
x_2	=	−2	−0.5	2
x_7	=	−3	−1	0
z	=	−6	−0.5	14

Although feasible, this tableau is not optimal for Phase II. Simplex rules lead us to perform the following degenerate pivot:

```
≫ T = ljx(T,5,1);
```

		x_7	x_5	1
x_3	=	1/3	1/3	4
x_4	=	1	0.5	4
x_1	=	−1/3	1/6	1
x_2	=	2/3	1/6	2
x_6	=	−1/3	−1/3	0
z	=	2	1.5	14

We have now identified a solution to the original problem: $x_1 = 1, x_2 = 2, z = 14$. ∎

The Phase I technique is interesting not just as a way to identify a starting point for Phase II but also in its own right as a technique to find a feasible point for a system of inequalities. The approach is the same; the only difference is that the tableau does not have a row for the objective function of the original problem (since there *is* no objective function).

Exercise 3-4-2. 1. Demonstrate that

$$\begin{array}{lrcrcl} \min & z = -3x_1 + x_2 & & & & \\ \text{subject to} & -x_1 & - & x_2 & \geq & -2, \\ & 2x_1 & + & 2x_2 & \geq & 10, \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

is infeasible.

2. Demonstrate that

$$\begin{array}{lrcrcl} \min & z = -x_1 + x_2 & & & & \\ \text{subject to} & 2x_1 & - & x_2 & \geq & 1, \\ & x_1 & + & 2x_2 & \geq & 2, \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

is unbounded.

Exercise 3-4-3. Solve the following linear programs using MATLAB. Give an optimal vector and its objective value if solvable; give a direction of unboundedness if unbounded; or verify that the problem is infeasible. Each problem has the form

$$\begin{array}{ll} \min & z = p'x \\ \text{subject to} & Ax \geq b, \quad x \geq 0, \end{array}$$

where the data A, b, and p are given below.

1. $$A = \begin{bmatrix} -1 & -3 & 0 & -1 \\ -2 & -1 & 0 & 0 \\ 0 & -1 & -4 & -1 \\ 1 & 1 & 2 & 0 \\ -1 & 1 & 4 & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} -4 \\ -3 \\ -3 \\ 1 \\ 1 \end{bmatrix}, \qquad p = \begin{bmatrix} -2 \\ -4 \\ -1 \\ -1 \end{bmatrix}.$$

2. $$A = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}, \qquad p = \begin{bmatrix} -1 \\ -3 \\ 0 \end{bmatrix}.$$

Exercise 3-4-4. Using the simplex method, find *all* solutions of

$$\begin{array}{lrcrcrcrcl} \min & z = 2x_1 & + & 3x_2 & + & 6x_3 & + & 4x_4 & & \\ \text{subject to} & x_1 & + & 2x_2 & + & 3x_3 & + & x_4 & \geq & 5, \\ & x_1 & + & x_2 & + & 2x_3 & + & 3x_4 & \geq & 3, \\ & & & & & & & x_1, x_2, x_3, x_4 & \geq & 0. \end{array}$$

3.5 Finite Termination

3.5.1 The Nondegenerate Case

We now ask the question, Is the method described in the previous section guaranteed to terminate (either at a solution or by finding a direction of unboundedness), or can we cycle forever between Steps 3 and 6? In this section, we show that termination is guaranteed under certain rather restrictive assumptions. (We relax these assumptions and prove a more general result in Section 3.5.3.)

To set up our result, we define the concept of a *degenerate tableau*.

Definition 3.5.1. A feasible tableau is *degenerate* if the last column contains any zero elements. If the elements in the last column are all strictly positive, the tableau is *nondegenerate*. A linear program is said to be *nondegenerate* if all feasible tableaus for that linear program are nondegenerate.

Geometrically, a tableau is nondegenerate if the vertex it defines is at the intersection of exactly n hyperplanes of the form $x_j = 0$; namely, those hyperplanes defined by $j \in \mathrm{N}$. Vertices that lie at the intersection of more than n hyperplanes correspond to degenerate

tableaus. (See the examples at the end of Section 3.2, illustrated in Figure 3.1.) Consequently, a linear program is nondegenerate if each of the vertices of the feasible region for that linear program is defined uniquely by a set N.

We encounter degenerate tableaus during the simplex method when there is a tie in the ratio test for selection of the pivot row. After the pivot is performed, zeros appear in the last column of the row(s) that tied but were *not* selected as pivots.

The finite termination of the simplex method under these assumptions is now shown.

Theorem 3.5.2. *If a linear program is* feasible *and* nondegenerate*, then starting at any feasible tableau, the objective function strictly decreases at each pivot step. After a finite number of pivots the method terminates with an* optimal *point or else identifies a direction of* unboundedness.

Proof. At every iteration, we must have a nonoptimal, optimal, or unbounded tableau. In the latter two cases, termination occurs. In the first case, the following transformation occurs when we pivot on an element H_{rs}, for which $h_r > 0$ (nondegeneracy) and $c_s < 0$ (by pivot selection), and $H_{rs} < 0$ (by the ratio test):

$$\begin{array}{rcc|c} & & x_{\mathrm{N}} & 1 \\ \hline x_{\mathrm{B}} & = & H & h \\ z & = & c^T & \alpha \end{array} \longrightarrow \begin{array}{rcc|c} & & x_{\tilde{\mathrm{N}}} & 1 \\ \hline x_{\tilde{\mathrm{B}}} & = & \tilde{H} & \tilde{h} \\ z & = & \tilde{c}^T & \tilde{\alpha} \end{array}$$

Here

$$\tilde{\alpha} = \alpha - \frac{c_s h_r}{H_{rs}} < \alpha,$$

where the strict inequality follows from the properties of c_s, h_r, and H_{rs}. Hence, we can never return to the tableau with objective α, since this would require us to increase the objective at a later iteration, something the simplex method does not allow. Since we can only visit each possible tableau at most once, and since there are only a finite number of possible tableaus, the method must eventually terminate at either an optimal or an unbounded tableau. □

In fact, a bound on the number of possible tableaus is obtained by determining the number of ways to choose the nonbasic set N (with n indices) from the index set $\{1, 2, \ldots, m+n\}$ which, by elementary combinatorics, is

$$\binom{m+n}{n} = \frac{(m+n)!}{m!n!}.$$

3.5.2 Cycling

We start by giving a classic example due to Beale (1955), which shows that for a degenerate linear program, reasonable rules for selecting pivots can fail to produce finite termination. In this example, the simplex method repeats the same sequence of six pivots indefinitely, making no progress toward a solution.

Example 3-5-1 (Beale). Consider the initial tableau

```
≫ load beale
≫ T = totbl(A,b,p);
```

		x_1	x_2	x_3	x_4	1
x_5	=	−0.5	5.5	2.5	−9	0
x_6	=	−0.5	1.5	0.5	−1	0
x_7	=	−1	0	0	0	1
z	=	−10	57	9	24	0

Let us use the following pivot rules:

1. The pivot column is the one with the most negative entry in the bottom row.
2. If two or more rows tie for being a pivot row in the ratio test, choose the one with the smallest subscript.

Note that the example has zeros in the final column, and so it does not satisfy the nondegeneracy assumption of Section 3.5. Carrying out the simplex method, following the pivot rules above, we obtain the following tableau after six pivots:

```
≫ T = ljx(T,1,1);
≫ T = ljx(T,2,2);
≫ T = ljx(T,1,3);
≫ T = ljx(T,2,4);
≫ T = ljx(T,1,1);
≫ T = ljx(T,2,2);
```

		x_3	x_4	x_1	x_2	1
x_5	=	2.5	−9	−0.5	5.5	0
x_6	=	0.5	−1	−0.5	1.5	0
x_7	=	0	0	−1	0	1
z	=	9	24	−10	57	0

Notice that apart from a column reordering, this tableau is identical to the initial tableau. If we continue to apply the pivot rules above, we will continue to cycle through the same six pivots. The positions of the columns may change at the end of every cycle, but the objective function will never change and we will make no progress toward a solution. This phenomenon is known as *cycling*. Cycling can occur only for degenerate problems; a nondegenerate pivot cannot be part of a cycle since it yields a decrease in the objective function. ∎

3.5.3 The General Case

We now revisit the issue of finite termination of the simplex method. The main result shows that a particular variant of the simplex method, when started from a feasible tableau, is guaranteed to produce either a solution or a direction of unboundedness within a finite number of iterations—without any nondegeneracy assumption such as the one used in Section 3.5. Finite termination of the two-phase simplex method—that is, determination of infeasibility, optimality, or unboundedness within a finite number of simplex pivots—follows as a simple consequence. Finite termination depends crucially on the rule used to select pivot columns (in the event of more than one negative entry in the last row) and on the rule for selecting the

pivot row (in the event of a tie in the ratio test). As shown above, even apparently reasonable rules can fail to produce finite termination.

We now modify the pivot selection rule of the simplex method to overcome this problem. This rule was introduced by Bland (1977) and is commonly called Bland's rule or the *smallest-subscript rule*.

1. *Pricing* (pivot column selection): The pivot column is the smallest $\mathrm{N}(s)$ of nonbasic variable indices such that column s has a negative element in the bottom row (reduced cost).

2. *Ratio Test* (pivot row selection): The pivot row is the smallest $\mathrm{B}(r)$ of basic variable indices such that row r satisfies

$$-h_r/H_{rs} = \min_i \{-h_i/H_{is} \mid H_{is} < 0\}.$$

In other words, among all possible pivot columns (those with negative reduced costs), we choose the one whose label has the smallest subscript. Among all possible pivot rows (those that tie for the minimum in the ratio test), we again choose the one whose label has the smallest subscript.

Example 3-5-2. We apply the smallest-subscript rule to Beale's problem.

```
≫ load beale
≫ T = totbl(A,b,p);
```

		x_1	x_2	x_3	x_4	1
x_5	=	−0.5	5.5	2.5	−9	0
x_6	=	−0.5	1.5	0.5	−1	0
x_7	=	−1	0	0	0	1
z	=	−10	57	9	24	0

The pivot column must be the column labeled x_1, since it is the only one with a negative reduced cost. In the ratio tests, the rows labeled x_5 and x_6 tie for minimum ratio; the smallest-subscript rule chooses x_5. By continuing in this manner, we generate the following sequence of five pivots, ending with the tableau shown below.

```
≫ T = ljx(T,1,1);
≫ T = ljx(T,2,2);
≫ T = ljx(T,1,3);
≫ T = ljx(T,2,4);
≫ T = ljx(T,1,1);
```

		x_3	x_6	x_1	x_2	1
x_5	=	−2	9	4	−8	0
x_4	=	0.5	−1	−0.5	1.5	0
x_7	=	0	0	−1	0	1
z	=	21	−24	−22	93	0

At this point, the smallest-subscript rule chooses the column labeled x_1 as the pivot column, whereas the "most negative" rule described earlier would have chosen x_6. By continuing with the next pivots, we obtain the following tableau:

```
≫ T = ljx(T,2,3);
≫ T = ljx(T,3,1);
```

		x_7	x_6	x_4	x_2	1
x_5	=	−2	5	−4	−2	2
x_1	=	−1	0	0	0	1
x_3	=	−1	2	2	−3	1
z	=	1	18	42	30	−1

Finally, we have arrived at an optimal tableau. The optimal value is -1, achieved when $x = (1, 0, 1, 0, 2, 0, 0)$. ∎

The following theorem establishes finiteness of the simplex method using the smallest-subscript rule, without any nondegeneracy assumption. The proof closely follows the one given by Chvátal (1983).

Theorem 3.5.3. *If a linear program is* feasible*, then starting at any feasible tableau, and using the smallest-subscript anticycling rule, the simplex method terminates after a finite number of pivots at an* optimal *or* unbounded *tableau.*

Proof. Since there are at most $\binom{m+n}{m}$ ways of choosing m basic variables from $n + m$ variables, some choice of basic variables (and the tableau that corresponds to these variables) must repeat if the simplex method cycles. We will show, by contradiction, that the method cannot cycle when the smallest-subscript anticycling rule is used.

Suppose for contradiction that some tableau is repeated. Then there is a sequence of degenerate pivots that leads from some tableau T_0 back to itself. We denote this sequence of tableaus as follows:

$$T_0, T_1, \ldots, T_k = T_0.$$

If any one of these pivots were nondegenerate, the objective function would decrease and the cycle would be broken. Hence, all pivots in the above sequence must be degenerate.

A variable will be called *fickle* if it is nonbasic in some of the tableaus $T_0, T_1, \ldots, T_k = T_0$ and basic in others. *Every* fickle variable must become basic at least once in the cycle *and* become nonbasic at least once in the cycle since the set of basic and nonbasic variables is the same at the beginning and end of the cycle. Furthermore, the value of a fickle variable is always zero, since every pivot is degenerate.

Among all fickle variables, let x_l have the largest subscript. Then among $T_1, \ldots, T_k$ there is some T where x_l is the basic variable chosen to become nonbasic. Let x_s be the (fickle) nonbasic variable that is chosen to become basic at this pivot. Our logic will show that there is another fickle variable x_r, $r < l$, that is eligible to become nonbasic at this pivot step as well, contradicting our choice of x_l as the variable to become nonbasic. The tableau T looks like this so far when we order according to subscripts:

$$\begin{array}{lcc|c} T: & & x_s & 1 \\ & = & & \\ x_r & = & H_{rs} & \\ & = & & \\ x_l & = & H_{ls} < 0 & 0 \\ & = & & \\ z & = & \geq 0 \quad c_s < 0 & \alpha \end{array} \tag{3.16}$$

For the purposes of this proof, we abuse notation and assume that the entries of the tableau h, c, and H are indexed not by $1, 2, \ldots, m$ and $1, 2, \ldots, n$ but by the corresponding basic and nonbasic labels B and N. Note therefore that $c_s < 0$ since s is chosen to enter at this pivot step. Since x_l is fickle, it must also be chosen to become basic at some other tableau $\tilde{T}$ in the sequence $T_1, T_2, \ldots, T_k$. Since the objective value is the same for all tableaus, the bottom row of $\tilde{T}$ represents the equation

$$z = \tilde{c}'_{\tilde{\mathrm{N}}} x_{\tilde{\mathrm{N}}} + \alpha, \tag{3.17}$$

where $\tilde{\mathrm{N}}$ (and $\tilde{\mathrm{B}}$) correspond to the nonbasic (and basic) variables in $\tilde{T}$. By defining $\tilde{c}_{\tilde{\mathrm{B}}} = 0$, we obtain

$$z = \tilde{c}'_{\tilde{\mathrm{B}}} x_{\tilde{\mathrm{B}}} + \tilde{c}'_{\tilde{\mathrm{N}}} x_{\tilde{\mathrm{N}}} + \alpha = \tilde{c}' x + \alpha.$$

For any $\lambda \in \mathbf{R}$, the tableau T (3.16) above defines the following relationships between the variables:

$$\begin{aligned} x_i &= \begin{cases} \lambda & \text{if } i = s, \\ H_{js}\lambda + h_j & \text{if } i = \mathrm{B}(j), \\ 0 & \text{if } i \in \mathrm{N} \setminus \{s\}, \end{cases} \\ z &= c_s \lambda + \alpha. \end{aligned} \tag{3.18}$$

By Theorem 2.1.1, the value $z = c_s\lambda + \alpha$ must be the same as the value of z obtained by substituting (3.18) into (3.17). Hence,

$$c_s\lambda + \alpha = \tilde{c}_{\tilde{\mathrm{N}}} x_{\tilde{\mathrm{N}}} + \alpha = \tilde{c}_s\lambda + \sum_{i \in \mathrm{B}} \tilde{c}_i (H_{is}\lambda + h_i) + \alpha,$$

which implies that

$$\left(c_s - \tilde{c}_s - \sum_{i \in \mathrm{B}} \tilde{c}_i H_{is} \right) \lambda = \sum_{i \in \mathrm{B}} \tilde{c}_i h_i.$$

Since this relationship holds for *any* $\lambda \in \mathbf{R}$, we must have

$$c_s - \tilde{c}_s - \sum_{i \in \mathrm{B}} \tilde{c}_i H_{is} = 0. \tag{3.19}$$

Since x_s becomes basic in T, we have from (3.16) that $c_s < 0$. Also, x_s does not become basic in $\tilde{T}$ (x_l does), and so because $s < l$, we must have $\tilde{c}_s \geq 0$. Thus from (3.19), we have

$$\sum_{i \in \mathrm{B}} \tilde{c}_i H_{is} < 0,$$

and so there must be some $r \in \mathrm{B}$ such that

$$\tilde{c}_r H_{rs} < 0. \tag{3.20}$$

This is the r we require for our contradiction. Note the following properties of x_r:

1. x_r is fickle: From (3.20), $\tilde{c}_r \neq 0$, and so $\tilde{c}_{\tilde{\mathrm{B}}} = 0$ implies that $r \in \tilde{\mathrm{N}}$. On the other hand, $r \in \mathrm{B}$, and so x_r is fickle. Since l is the largest fickle index, we must have $r \leq l$.

2. $r < l$: Since x_l becomes nonbasic in T, $H_{ls} < 0$; see (3.16). Also $\tilde{c}_l < 0$ because x_l becomes basic in $\tilde{T}$. It follows from (3.20) that $r \neq l$.

3. $H_{rs} < 0$: Since (as established above) $r \in \tilde{\mathrm{N}}$, and since $r < l$, we must have that $\tilde{c}_r \geq 0$ (otherwise r, not l, would be chosen to become basic at $\tilde{T}$). It now follows from (3.20) that $H_{rs} < 0$.

By combining all this information, we conclude that T has the following structure:

		x_s	1
	=		
x_r	=	$H_{rs} < 0$	0
	=		
x_l	=	$H_{ls} < 0$	0
	=		
z	=	$\geq 0 \quad c_s < 0$	α

According to this tableau, x_r is chosen to become nonbasic, and $r < l$ contradicts our hypothesis that x_l is chosen to become nonbasic in T. □

We can now use Theorem 3.5.3 to show that the two-phase simplex method identifies a solution, or else indicates infeasibility or unboundedness, after a finite number of pivots.

Theorem 3.5.4. *For a linear program* (3.1)*, the two-phase simplex method with the smallest-subscript anticycling rule terminates after a finite number of pivots with a conclusion that the problem is* infeasible*, or at an* optimal *or* unbounded *tableau.*

Proof. Consider first Phase I. After the initial pivot, we have a feasible tableau for Phase I, and since the Phase I problem is bounded below, it must have an optimal solution. Theorem 3.5.3 indicates that this solution is found after a finite number of pivots. If $x_0 > 0$ at optimality, then the original problem is infeasible. If $x_0 = 0$, then we can construct a feasible tableau for Phase II. By applying Theorem 3.5.3 again, we find that an optimal or unbounded tableau is reached for Phase II after a finite number of pivots. □

Exercise 3-5-3. Consider the following tableau:

		x_2	x_3	1
x_1	=	1	-2	2
x_5	=	2	-1	1
x_4	=	2	-1	2
z	=	1	$\mu - 3$	-1

Note that the variables $x_1, x_2, \ldots, x_5$ are all nonnegative. Indicate for each of the three cases $\mu < 3$, $\mu > 3$, $\mu = 3$ whether the tableau is

- infeasible;
- feasible but not optimal (do not do any pivoting);
- optimal with a unique solution (write it down);
- optimal with a nonunique solution (write down at least two solutions).

Exercise 3-5-4. Read the following statements carefully to see whether it is *true* or *false*. Explain your answers briefly in each case.