8

Linear Algebraic Equations and Matrices

CHAPTER OBJECTIVES

The primary objective of this chapter is to acquaint you with linear algebraic equations and their relationship to matrices and matrix algebra. Specific objectives and topics covered are

- Understanding matrix notation.
- Being able to identify the following types of matrices: identity, diagonal, symmetric, triangular, and tridiagonal.
- Knowing how to perform matrix multiplication and being able to assess when it is feasible.
- Knowing how to represent a system of linear algebraic equations in matrix form.
- Knowing how to solve linear algebraic equations with left division and matrix inversion in MATLAB.

YOU'VE GOT A PROBLEM

Suppose that three jumpers are connected by bungee cords. Figure 8.1*a* shows them being held in place vertically so that each cord is fully extended but unstretched. We can define three distances, x_1, x_2, and x_3, as measured downward from each of their unstretched positions. After they are released, gravity takes hold and the jumpers will eventually come to the equilibrium positions shown in Fig. 8.1*b*.

Suppose that you are asked to compute the displacement of each of the jumpers. If we assume that each cord behaves as a linear spring and follows Hooke's law, free-body diagrams can be developed for each jumper as depicted in Fig. 8.2.

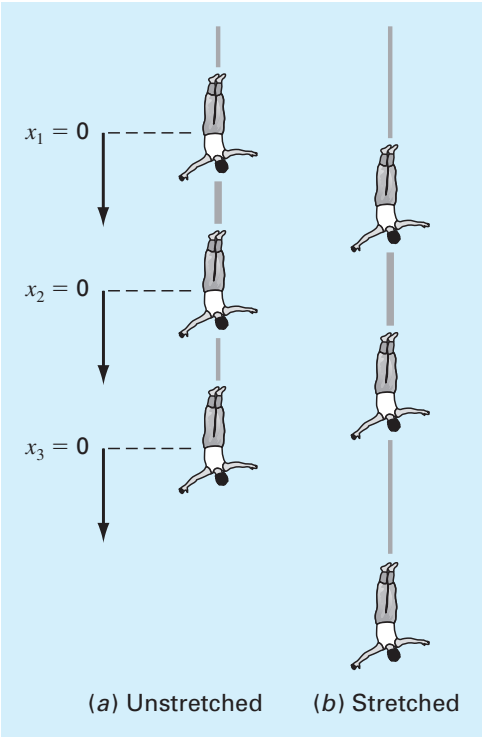


FIGURE 8.1
Three individuals connected by bungee cords.

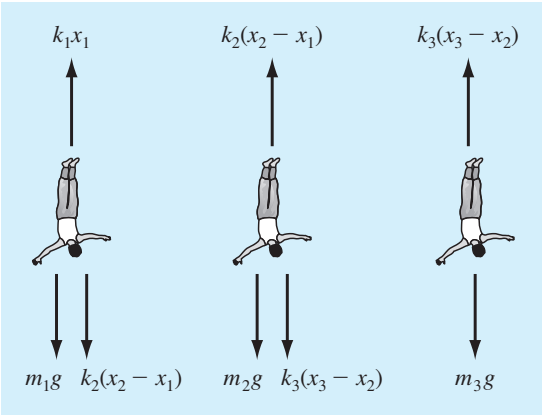


FIGURE 8.2
Free-body diagrams.

Using Newton's second law, force balances can be written for each jumper:

$$\begin{aligned} m_1\frac{d^2x_1}{dt^2} &= m_1g + k_2(x_2 - x_1) - k_1x_1 \\ m_2\frac{d^2x_2}{dt^2} &= m_2g + k_3(x_3 - x_2) + k_2(x_1 - x_2) \\ m_3\frac{d^2x_3}{dt^2} &= m_3g + k_3(x_2 - x_3) \end{aligned} \tag{8.1}$$

where m_i = the mass of jumper i (kg), t = time (s), k_j = the spring constant for cord j (N/m), x_i = the displacement of jumper i measured downward from the equilibrium position (m), and g = gravitational acceleration (9.81 m/s^2). Because we are interested in the steady-state solution, the second derivatives can be set to zero. Collecting terms gives

$$\begin{aligned} (k_1 + k_2)x_1 \quad - k_2x_2 \quad &= m_1g \\ -k_2x_1 + (k_2 + k_3)x_2 - k_3x_3 &= m_2g \\ -k_3x_2 + k_3x_3 &= m_3g \end{aligned} \tag{8.2}$$

Thus, the problem reduces to solving a system of three simultaneous equations for the three unknown displacements. Because we have used a linear law for the cords, these equations are linear algebraic equations. Chapters 8 through 12 will introduce you to how MATLAB is used to solve such systems of equations.

8.1 MATRIX ALGEBRA OVERVIEW

Knowledge of matrices is essential for understanding the solution of linear algebraic equations. The following sections outline how matrices provide a concise way to represent and manipulate linear algebraic equations.

8.1.1 Matrix Notation

A *matrix* consists of a rectangular array of elements represented by a single symbol. As depicted in Fig. 8.3, $[A]$ is the shorthand notation for the matrix and a_{ij} designates an individual *element* of the matrix.

A horizontal set of elements is called a *row* and a vertical set is called a *column*. The first subscript i always designates the number of the row in which the element lies. The second subscript j designates the column. For example, element a_{23} is in row 2 and column 3.

The matrix in Fig. 8.3 has m rows and n columns and is said to have a dimension of m by n (or $m \times n$). It is referred to as an m by n matrix.

Matrices with row dimension $m = 1$, such as

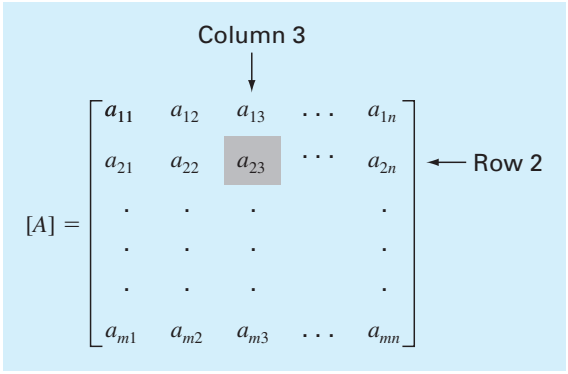
$$[b] = [b_1 \quad b_2 \quad \cdots \quad b_n]$$

are called *row vectors*. Note that for simplicity, the first subscript of each element is dropped. Also, it should be mentioned that there are times when it is desirable to employ a special shorthand notation to distinguish a row matrix from other types of matrices. One way to accomplish this is to employ special open-topped brackets, as in $\lfloor b \rfloor$.[1]

Matrices with column dimension $n = 1$, such as

$$[c] = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \tag{8.3}$$

FIGURE 8.3
A matrix.



[1] In addition to special brackets, we will use case to distinguish between vectors (lowercase) and matrices (uppercase).

are referred to as *column vectors*. For simplicity, the second subscript is dropped. As with the row vector, there are occasions when it is desirable to employ a special shorthand notation to distinguish a column matrix from other types of matrices. One way to accomplish this is to employ special brackets, as in $\{c\}$.

Matrices where $m = n$ are called *square matrices*. For example, a 3×3 matrix is

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The diagonal consisting of the elements a_{11}, a_{22}, and a_{33} is termed the *principal* or *main diagonal* of the matrix.

Square matrices are particularly important when solving sets of simultaneous linear equations. For such systems, the number of equations (corresponding to rows) and the number of unknowns (corresponding to columns) must be equal for a unique solution to be possible. Consequently, square matrices of coefficients are encountered when dealing with such systems.

There are a number of special forms of square matrices that are important and should be noted:

A *symmetric matrix* is one where the rows equal the columns—that is, $a_{ij} = a_{ji}$ for all i's and j's. For example,

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

is a 3×3 symmetric matrix.

A *diagonal matrix* is a square matrix where all elements off the main diagonal are equal to zero, as in

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

Note that where large blocks of elements are zero, they are left blank.

An *identity matrix* is a diagonal matrix where all elements on the main diagonal are equal to 1, as in

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

The identity matrix has properties similar to unity. That is,

$$[A][I] = [I][A] = [A]$$

An *upper triangular matrix* is one where all the elements below the main diagonal are zero, as in

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{bmatrix}$$

A *lower triangular matrix* is one where all elements above the main diagonal are zero, as in

$$[A] = \begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

A *banded matrix* has all elements equal to zero, with the exception of a band centered on the main diagonal:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{bmatrix}$$

The preceding matrix has a bandwidth of 3 and is given a special name—the *tridiagonal matrix.*

8.1.2 Matrix Operating Rules

Now that we have specified what we mean by a matrix, we can define some operating rules that govern its use. Two m by n matrices are equal if, and only if, every element in the first is equal to every element in the second—that is, $[A] = [B]$ if $a_{ij} = b_{ij}$ for all i and j.

Addition of two matrices, say, $[A]$ and $[B]$, is accomplished by adding corresponding terms in each matrix. The elements of the resulting matrix $[C]$ are computed as

$$c_{ij} = a_{ij} + b_{ij}$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Similarly, the subtraction of two matrices, say, $[E]$ minus $[F]$, is obtained by subtracting corresponding terms, as in

$$d_{ij} = e_{ij} - f_{ij}$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. It follows directly from the preceding definitions that addition and subtraction can be performed only between matrices having the same dimensions.

Both addition and subtraction are commutative:

$$[A] + [B] = [B] + [A]$$

and associative:

$$([A] + [B]) + [C] = [A] + ([B] + [C])$$

The multiplication of a matrix $[A]$ by a scalar g is obtained by multiplying every element of $[A]$ by g. For example, for a 3×3 matrix:

$$[D] = g[A] = \begin{bmatrix} ga_{11} & ga_{12} & ga_{13} \\ ga_{21} & ga_{22} & ga_{23} \\ ga_{31} & ga_{32} & ga_{33} \end{bmatrix}$$

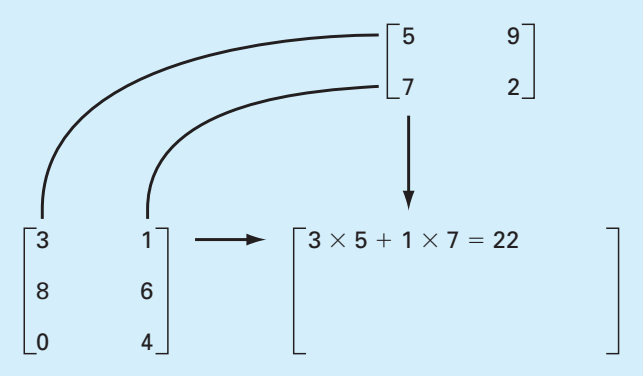


FIGURE 8.4
Visual depiction of how the rows and columns line up in matrix multiplication.

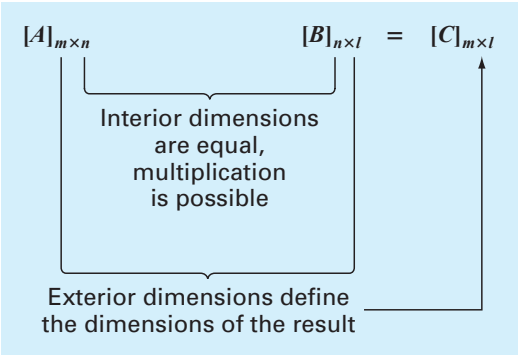


FIGURE 8.5
Matrix multiplication can be performed only if the inner dimensions are equal.

The product of two matrices is represented as $[C] = [A][B]$, where the elements of $[C]$ are defined as

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{8.4}$$

where $n =$ the column dimension of $[A]$ and the row dimension of $[B]$. That is, the c_{ij} element is obtained by adding the product of individual elements from the ith row of the first matrix, in this case $[A]$, by the jth column of the second matrix $[B]$. Figure 8.4 depicts how the rows and columns line up in matrix multiplication.

According to this definition, matrix multiplication can be performed only if the first matrix has as many columns as the number of rows in the second matrix. Thus, if $[A]$ is an m by n matrix, $[B]$ could be an n by l matrix. For this case, the resulting $[C]$ matrix would have the dimension of m by l. However, if $[B]$ were an m by l matrix, the multiplication could not be performed. Figure 8.5 provides an easy way to check whether two matrices can be multiplied.

If the dimensions of the matrices are suitable, matrix multiplication is *associative:*

$$([A][B])\,[C] = [A]([B][C])$$

and *distributive:*

$$[A]([B] + [C]) = [A][B] + [A][C]$$

or

$$([A] + [B])[C] = [A][C] + [B][C]$$

However, multiplication is not generally *commutative:*

$$[A][B] \neq [B][A]$$

That is, the order of matrix multiplication is important.

Although multiplication is possible, matrix division is not a defined operation. However, if a matrix $[A]$ is square and nonsingular, there is another matrix $[A]^{-1}$, called the *inverse* of $[A]$, for which

$$[A][A]^{-1} = [A]^{-1}[A] = [I]$$

Thus, the multiplication of a matrix by the inverse is analogous to division, in the sense that a number divided by itself is equal to 1. That is, multiplication of a matrix by its inverse leads to the identity matrix.

The inverse of a 2×2 matrix can be represented simply by

$$[A]^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

Similar formulas for higher-dimensional matrices are much more involved. Chapter 11 will deal with techniques for using numerical methods and the computer to calculate the inverse for such systems.

The *transpose* of a matrix involves transforming its rows into columns and its columns into rows. For example, for the 3×3 matrix:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

the transpose, designated $[A]^T$, is defined as

$$[A]^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

In other words, the element a_{ij} of the transpose is equal to the a_{ji} element of the original matrix.

The transpose has a variety of functions in matrix algebra. One simple advantage is that it allows a column vector to be written as a row, and vice versa. For example, if

$$\{c\} = \begin{Bmatrix} c_1 \\ c_1 \\ c_1 \end{Bmatrix}$$

then

$$\{c\}^T = \lfloor c_1 \quad c_2 \quad c_3 \rfloor$$

In addition, the transpose has numerous mathematical applications.

A *permutation matrix* (also called a *transposition matrix*) is an identity matrix with rows and columns interchanged. For example, here is a permutation matrix that is constructed by switching the first and third rows and columns of a 3×3 identity matrix:

$$[P] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Left multiplying a matrix $[A]$ by this matrix, as in $[P][A]$, will switch the corresponding rows of $[A]$. Right multiplying, as in $[A][P]$, will switch the corresponding columns. Here is an example of left multiplication:

$$[P][A] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -7 & 4 \\ 8 & 3 & -6 \\ 5 & 1 & 9 \end{bmatrix} = \begin{bmatrix} 5 & 1 & 9 \\ 8 & 3 & -6 \\ 2 & -7 & 4 \end{bmatrix}$$

The final matrix manipulation that will have utility in our discussion is *augmentation*. A matrix is augmented by the addition of a column (or columns) to the original matrix. For example, suppose we have a 3×3 matrix of coefficients. We might wish to augment this matrix $[A]$ with a 3×3 identity matrix to yield a 3×6 dimensional matrix:

$$\left[\begin{array}{ccc|ccc} a_{11} & a_{11} & a_{11} & 1 & 0 & 0 \\ a_{21} & a_{21} & a_{21} & 0 & 1 & 0 \\ a_{31} & a_{31} & a_{31} & 0 & 0 & 1 \end{array}\right]$$

Such an expression has utility when we must perform a set of identical operations on the rows of two matrices. Thus, we can perform the operations on the single augmented matrix rather than on the two individual matrices.

EXAMPLE 8.1 MATLAB Matrix Manipulations

Problem Statement. The following example illustrates how a variety of matrix manipulations are implemented with MATLAB. It is best approached as a hands-on exercise on the computer.

Solution. Create a 3×3 matrix:

```
>> A = [1 5 6;7 4 2;-3 6 7]

A =
     1     5     6
     7     4     2
    -3     6     7
```

The transpose of $[A]$ can be obtained using the ' operator:

```
>> A'

ans =
     1     7    -3
     5     4     6
     6     2     7
```

Next we will create another 3×3 matrix on a row basis. First create three row vectors:

```
>> x = [8 6 9];
>> y = [-5 8 1];
>> z = [4 8 2];
```

Then we can combine these to form the matrix:

```
>> B = [x; y; z]

B =
     8     6     9
    -5     8     1
     4     8     2
```

We can add [*A*] and [*B*] together:

```
>> C = A+B

C =
     9    11    15
     2    12     3
     1    14     9
```

Further, we can subtract [*B*] from [*C*] to arrive back at [*A*]:

```
>> A = C-B

A =
     1     5     6
     7     4     2
    -3     6     7
```

Because their inner dimensions are equal, [*A*] and [*B*] can be multiplied

```
>> A*B

ans =
     7    94    26
    44    90    71
   -26    86    -7
```

Note that [*A*] and [*B*] can also be multiplied on an element-by-element basis by including a period with the multiplication operator as in

```
>> A.*B

ans =
     8    30    54
   -35    32     2
   -12    48    14
```

A 2×3 matrix can be set up

```
>> D = [1 4 3;5 8 1];
```

If [*A*] is multiplied times [*D*], an error message will occur

```
>> A*D

??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

However, if we reverse the order of multiplication so that the inner dimensions match, matrix multiplication works

```
>> D*A

ans =

    20    39    35
    58    63    53
```

The matrix inverse can be computed with the `inv` function:

```
>> AI = inv(A)

AI =

    0.2462    0.0154   -0.2154
   -0.8462    0.3846    0.6154
    0.8308   -0.3231   -0.4769
```

To test that this is the correct result, the inverse can be multiplied by the original matrix to give the identity matrix:

```
>> A*AI

ans =

    1.0000   -0.0000   -0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

The `eye` function can be used to generate an identity matrix:

```
>> I = eye(3)

I =

     1     0     0
     0     1     0
     0     0     1
```

We can set up a permutation matrix to switch the first and third rows and columns of a 3×3 matrix as

```
>> P=[0 0 1;0 1 0;1 0 0]

P =

     0     0     1
     0     1     0
     1     0     0
```

We can then either switch the rows:

```
>> PA=P*A

PA =

    -3     6     7
     7     4     2
     1     5     6
```

or the columns:

```
>> AP=A*P

AP =

     6     5     1
     2     4     7
     7     6    -3
```

Finally, matrices can be augmented simply as in

```
>> Aug = [A I]

Aug =

     1     5     6     1     0     0
     7     4     2     0     1     0
    -3     6     7     0     0     1
```

Note that the dimensions of a matrix can be determined by the `size` function:

```
>> [n,m] = size(Aug)

n =
     3

m =
     6
```

8.1.3 Representing Linear Algebraic Equations in Matrix Form

It should be clear that matrices provide a concise notation for representing simultaneous linear equations. For example, a 3×3 set of linear equations,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \tag{8.5}$$

can be expressed as

$$[A]\{x\} = \{b\} \tag{8.6}$$

where $[A]$ is the matrix of coefficients:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$\{b\}$ is the column vector of constants:

$$\{b\}^T = \lfloor b_1 \quad b_2 \quad b_3 \rfloor$$

and $\{x\}$ is the column vector of unknowns:

$$\{x\}^T = \lfloor x_1 \quad x_2 \quad x_3 \rfloor$$

Recall the definition of matrix multiplication [Eq. (8.4)] to convince yourself that Eqs. (8.5) and (8.6) are equivalent. Also, realize that Eq. (8.6) is a valid matrix multiplication because the number of columns n of the first matrix $[A]$ is equal to the number of rows n of the second matrix $\{x\}$.

This part of the book is devoted to solving Eq. (8.6) for $\{x\}$. A formal way to obtain a solution using matrix algebra is to multiply each side of the equation by the inverse of $[A]$ to yield

$$[A]^{-1}[A]\{x\} = [A]^{-1}\{b\}$$

Because $[A]^{-1}[A]$ equals the identity matrix, the equation becomes

$$\{x\} = [A]^{-1}\{b\} \tag{8.7}$$

Therefore, the equation has been solved for $\{x\}$. This is another example of how the inverse plays a role in matrix algebra that is similar to division. It should be noted that this is not a very efficient way to solve a system of equations. Thus, other approaches are employed in numerical algorithms. However, as discussed in Section 11.1.2, the matrix inverse itself has great value in the engineering analyses of such systems.

It should be noted that systems with more equations (rows) than unknowns (columns), $m > n$, are said to be *overdetermined.* A typical example is least-squares regression where an equation with n coefficients is fit to m data points (x, y). Conversely, systems with less equations than unknowns, $m < n$, are said to be *underdetermined.* A typical example of underdetermined systems is numerical optimization.

8.2 SOLVING LINEAR ALGEBRAIC EQUATIONS WITH MATLAB

MATLAB provides two direct ways to solve systems of linear algebraic equations. The most efficient way is to employ the backslash, or "left-division," operator as in

```
>> x = A\b
```

The second is to use matrix inversion:

```
>> x = inv(A)*b
```

As stated at the end of Section 8.1.3, the matrix inverse solution is less efficient than using the backslash. Both options are illustrated in the following example.