Chapter 1

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. 8.1b.

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.

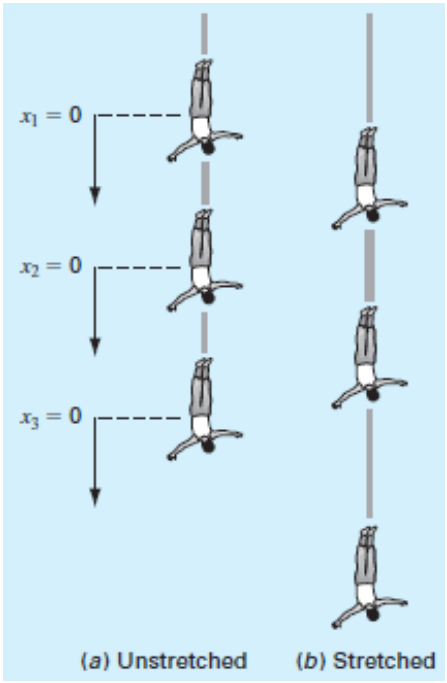

Figure 1.1: Three individuals connected by bungee cords.

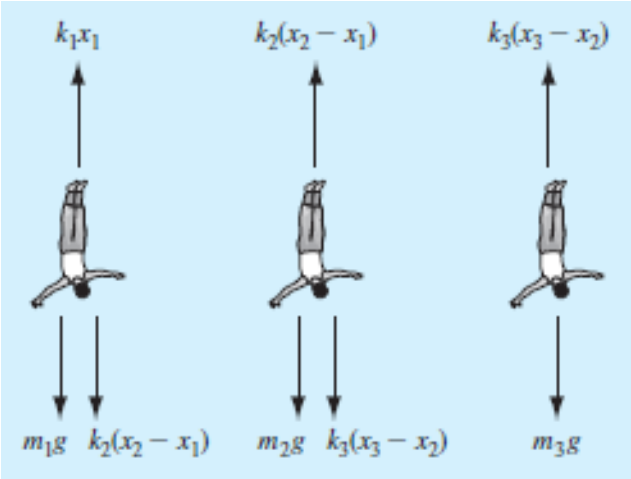

Figure 1.2: Free-body diagrams.

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 $(9.81\ \text{m/s}^2)$. Because we are interested in the steady-state solution, the second derivatives can be set to zero. Collecting terms gives

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

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.

1.1. 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.

1.1.1. 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

$$[b] = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$$

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 $\lfloor b \rfloor$.

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

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

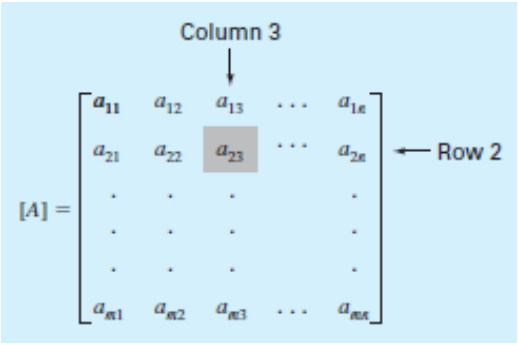

Figure 1.3: A matrix.

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.

1.1.2. 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:

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

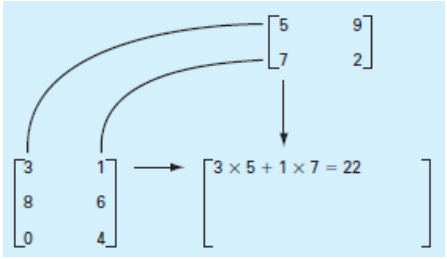

Figure 1.4: Visual depiction of how the rows and columns line up in matrix multiplication.

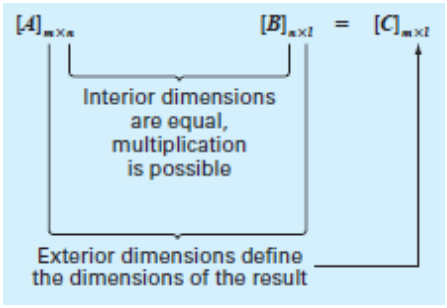

Figure 1.5: Matrix multiplication can be performed only if the inner dimensions are equal.

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 i th row of the first matrix, in this case $[A]$, by the j th 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

$$\{c\}^T = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}$$

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 1.1. 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
```

■

1.1.3. 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:

$$\{b\}^T = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$$

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

$$\{x\}^T = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

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.

1.2. 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.

Example 1.2. Solving the Bungee Jumper Problem with MATLAB

Problem Statement. Use MATLAB to solve the bungee jumper problem described at the beginning of this chapter. The parameters for the problem are

Jumper	Mass (**kg**)	Spring Constant ($\mathbf{N/m}$)	Unstretched Cord Length (**m**)
Top (1)	60	50	20
Middle (2)	70	100	20
Bottom (3)	80	50	20

Solution. Substituting these parameter values into Eq. (8.2) gives:

$$\begin{bmatrix} 150 & -100 & 0 \\ -100 & 150 & -50 \\ 0 & -50 & 50 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 588.6 \\ 686.7 \\ 784.8 \end{Bmatrix}$$

Start up MATLAB and enter the coefficient matrix and the right-hand-side vector:

```
>> K = [150 -100 0;-100 150 -50;0 -50 50]
K =
     150 -100 0
     -100 150 -50
     0 -50 50
>> mg = [588.6; 686.7; 784.8]
mg =
     588.6000
     686.7000
     784.8000
```

Employing left division yields

```
>> x = K\mg
x =
     41.2020
     55.9170
     71.6130
```

Alternatively, multiplying the inverse of the coefficient matrix by the right-hand-side vector gives the same result:

```
>> x = inv(K)*mg
x =
     41.2020
     55.9170
     71.6130
```

Because the jumpers were connected by 20 – m cords, their initial positions relative to the platform is

```
>> xi = [20;40;60];
```

Thus, their final positions can be calculated as

```
>> xf = x+xi
xf =
     61.2020
     95.9170
     131.6130
```

The results, which are displayed in Fig. 8.6, make sense. The first cord is extended the longest because it has a lower spring constant and is subject to the most weight (all three jumpers). Notice that the second and third cords are extended about the same amount. Because it is subject to the weight of two jumpers, one might expect the second cord to be extended longer than the third. However, because it is stiffer (i.e., it has a higher spring constant), it stretches less than expected based on the weight it carries.

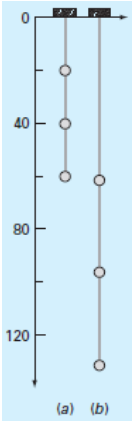

Figure 1.6: Positions of three individuals connected by bungee cords. (a) Unstretched and (b) stretched.

■

1.3. CASE STUDY - CURRENTS AND VOLTAGES IN CIRCUITS

Background. Recall that in Chap. 1 (Table 1.1), we summarized some models and associated conservation laws that figure prominently in engineering. As in Fig. 8.7, each model represents a system of interacting elements. Consequently, steady-state balances derived from the conservation laws yield systems of simultaneous equations. In many cases, such systems are linear and hence can be expressed in matrix form. The present case study focuses on one such application: circuit analysis.

A common problem in electrical engineering involves determining the currents and voltages at various locations in resistor circuits. These problems are solved using Kirchhoff's current and voltage rules. The current (or point) rule states that the algebraic sum of all currents entering a node must be zero (Fig. 8.8*a*), or

$$\sum i = 0 \tag{8.8}$$

where all current entering the node is considered positive in sign. The current rule is an application of the principle of conservation of charge (recall Table 1.1).

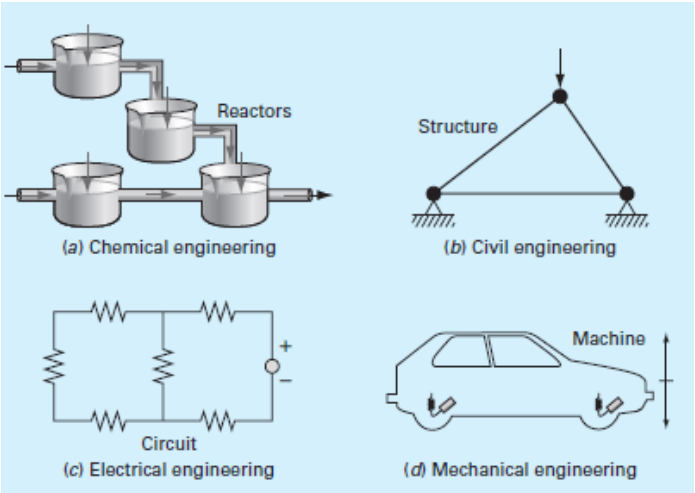

Figure 1.7: Engineering systems which, at steady state, can be modeled with linear algebraic equations.

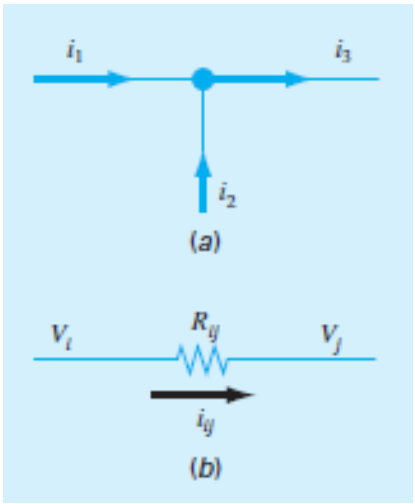

Figure 1.8: Schematic representations of (a) Kirchhoff's current rule and (b) Ohm's law.

The voltage (or loop) rule specifies that the algebraic sum of the potential differences (i.e., voltage changes) in any loop must equal zero. For a resistor circuit, this is expressed as

$$\sum \xi - \sum iR = 0 \tag{8.9}$$

where ξ is the emf (electromotive force) of the voltage sources, and R is the resistance of any resistors on the loop. Note that the second term derives from Ohm's law (Fig. 8.8 b), which states that the voltage drop across an ideal resistor is equal to the product of the current and the resistance. Kirchhoff's voltage rule is an expression of the conservation of energy.

Solution. Application of these rules results in systems of simultaneous linear algebraic equations because the various loops within a circuit are interconnected. For example, consider the circuit shown in Fig. 8.9. The currents associated with this circuit are unknown both in magnitude and direction. This presents no great difficulty because one simply assumes a direction for each current. If the resultant solution from Kirchhoff's laws is negative, then the assumed direction was incorrect. For example, Fig. 8.10 shows some assumed currents.

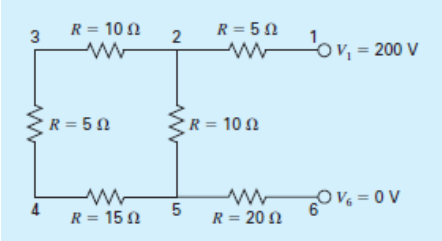

Figure 1.9: A resistor circuit to be solved using simultaneous linear algebraic equations.

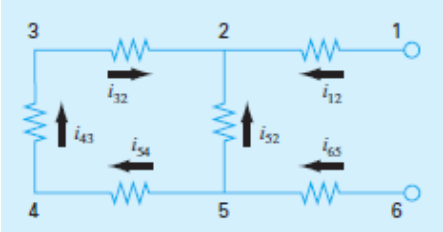

Figure 1.10: Assumed current directions.

Given these assumptions, Kirchhoff's current rule is applied at each node to yield

$$\begin{aligned} i_{12} + i_{52} + i_{32} &= 0 \\ i_{65} - i_{52} - i_{54} &= 0 \\ i_{43} - i_{32} &= 0 \\ i_{54} - i_{43} &= 0 \end{aligned}$$

Application of the voltage rule to each of the two loops gives

$$\begin{aligned} -i_{54}R_{54} - i_{43}R_{43} - i_{32}R_{32} + i_{52}R_{52} &= 0 \\ -i_{65}R_{65} - i_{52}R_{52} + i_{12}R_{12} - 200 &= 0 \end{aligned}$$

or, substituting the resistances from Fig. 8.9 and bringing constants to the right-hand side,

$$\begin{aligned} -15i_{54} - 5i_{43} - 10i_{32} + 10i_{52} &= 0 \\ -20i_{65} - 10i_{52} + 5i_{12} &= 200 \end{aligned}$$

Therefore, the problem amounts to solving six equations with six unknown currents. These equations can be expressed in matrix form as

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 10 & -10 & 0 & -15 & -5 \\ 5 & -10 & 0 & -20 & 0 & 0 \end{bmatrix} \begin{Bmatrix} i_{12} \\ i_{52} \\ i_{32} \\ i_{65} \\ i_{54} \\ i_{43} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 200 \end{Bmatrix}$$

Although impractical to solve by hand, this system is easily handled by MATLAB. The solution is

```
>> A=[1 1 1 0 0 0
0 -1 0 1 -1 0
0 0 -1 0 0 1
0 0 0 0 1 -1
0 10 -10 0 -15 -5
5 -10 0 -20 0 0];
>> b=[0 0 0 0 0 200]';
>> current=A\b
current =
    6.1538
    -4.6154
    -1.5385
    -6.1538
    -1.5385
    -1.5385
```

Thus, with proper interpretation of the signs of the result, the circuit currents and voltages are as shown in Fig. 8.11. The advantages of using MATLAB for problems of this type should be evident.

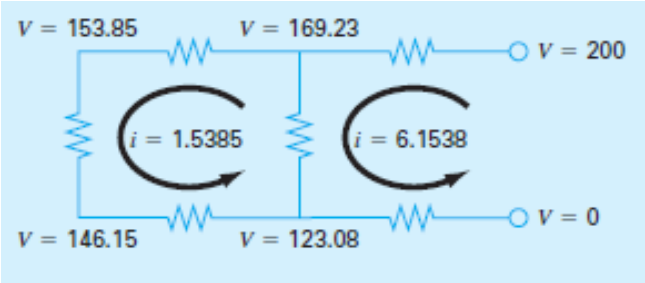

Figure 1.11: The solution for currents and voltages obtained using MATLAB.

PROBLEMS

8.1 Given a square matrix [A], write a single line MATLAB command that will create a new matrix $[Au_g]$ that consists of the original matrix $[A]$ augmented by an identity matrix $[I]$.

8.2 A number of matrices are defined as

$$[A] = \begin{bmatrix} 4 & 5 \\ 1 & 2 \\ 5 & 6 \end{bmatrix} \quad [B] = \begin{bmatrix} 4 & 3 & 7 \\ 1 & 2 & 6 \\ 2 & 0 & 4 \end{bmatrix}$$

$$\{C\} = \begin{Bmatrix} 2 \\ 6 \\ 1 \end{Bmatrix} \quad [D] = \begin{bmatrix} 5 & 4 & 3 & -7 \\ 2 & 1 & 7 & 5 \end{bmatrix}$$

$$[E] = \begin{bmatrix} 1 & 5 & 6 \\ 7 & 1 & 3 \\ 4 & 0 & 6 \end{bmatrix}$$

$$[F] = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 7 & 4 \end{bmatrix} \quad \lfloor G \rfloor = \begin{bmatrix} 8 & 6 & 4 \end{bmatrix}$$

Answer the following questions regarding these matrices:
(a) What are the dimensions of the matrices?
(b) Identify the square, column, and row matrices.
(c) What are the values of the elements: $a_{12}, b_{23}, d_{32}, e_{22}, f_{12}, g_{12}$?
(d) Perform the following operations:
(1) $[E]+[B]$
(2) $[A]+[F]$
(3) $[B]-[E]$
(4) $7\times[B]$
(5) $\{C\}^T$
(6) $[E]\times[B]$
(7) $[B]\times[E]$
(8) $[D]^T$
(9) $[G]\times\{C\}$
(10) $[I]\times[B]$
(11) $[E]^T\times[E]$
(12) $\{C\}^T\times\{C\}$

8.3 Write the following set of equations in matrix form:

$$50 = 5x_3 - 6x_2$$
$$2x_2 + 7x_3 + 30 = 0$$
$$x_1 - 7x_3 = 50 - 3x_2 + 5x_1$$

Use MATLAB to solve for the unknowns. In addition, use it to compute the transpose and the inverse of the coefficient matrix.

8.4 Three matrices are defined as

$$[A] = \begin{bmatrix} 6 & -1 \\ 12 & 7 \\ -5 & 3 \end{bmatrix} \quad [B] = \begin{bmatrix} 4 & 0 \\ 0.6 & 8 \end{bmatrix} \quad [C] = \begin{bmatrix} 1 & -2 \\ -6 & 1 \end{bmatrix}$$

(a) Perform all possible multiplications that can be computed between pairs of these matrices.
(b) Justify why the remaining pairs cannot be multiplied.
(c) Use the results of (a) to illustrate why the order of multiplication is important.

8.5 Solve the following system with MATLAB:

$$\begin{bmatrix} 3+2i & 4 \\ -i & 1 \end{bmatrix} \begin{Bmatrix} z_1 \\ z_2 \end{Bmatrix} = \begin{Bmatrix} 2+i \\ 3 \end{Bmatrix}$$

8.6 Develop, debug, and test your own M-file to multiply two matrices that is, $[X] = [Y][Z]$, where $[Y]$ is m by n and $[Z]$ is n by p. Employ `for...  end` loops to implement the multiplication and include error traps to flag bad cases. Test the program using the matrices from Prob. 8.4.

8.7 Develop, debug, and test your own M-file to generate the transpose of a matrix. Employ `for...  end` loops to implement the transpose. Test it on the matrices from Prob. 8.4.

8.8 Develop, debug, and test your own M-file function to switch the rows of a matrix using a permutation matrix. The first lines of the function should be as follows:

```
function B = permut(A,r1,r2)
% Permut: Switch rows of matrix
A
% with a permutation matrix
% B = permut(A,r1,r2)
% input:
% A = original matrix
% r1, r2 = rows to be switched
% output:
% B = matrix with rows switched
```

Include error traps for erroneous inputs (e.g., user specifies rows that exceed the dimensions of the original matrix).

8.9 Five reactors linked by pipes are shown in Fig. P8.9. The rate of mass flow through each pipe is computed as the product of flow (Q) and concentration (c). At steady state, the mass flow into and out of each reactor must be equal. For example, for the first reactor, a mass balance can be written as

$$Q_{01}c_{01} + Q_{31}c_3 = Q_{15}c_1 + Q_{12}c_1$$

Write mass balances for the remaining reactors in Fig. P8.9 and express the equations in matrix form. Then use MATLAB to solve for the concentrations in each reactor.

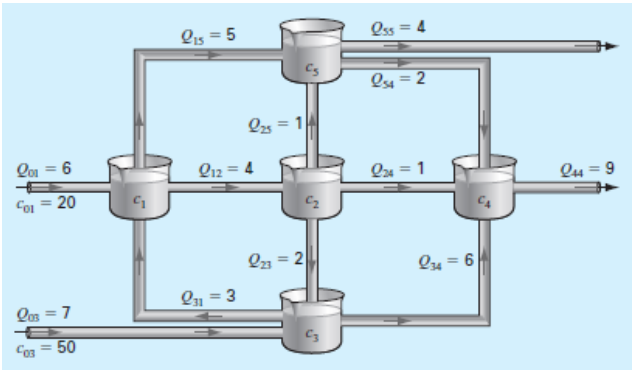

Figure 1.12: FIGURE P8.9

8.10 An important problem in structural engineering is that of finding the forces in a statically detemminate truss (Fig. P8.10). This type of structure can be described as a system of coupled linear algebraic equations derived from force balances. The sum of the forces in both horizontal and vertical directions must be zero at each node, because the system is at rest. Therefore, for node 1:

$$\sum F_H = 0 = -F_1\cos 30^\circ + F_3\cos 60^\circ + F_{1,h}$$
$$\sum F_V = 0 = -F_1\sin 30^\circ - F_3\sin 60^\circ + F_{1,v}$$

for node 2:

$$\sum F_H = 0 = F_2 + F_1\cos 30^\circ + F_{2,h} + H_2$$
$$\sum F_V = 0 = F_1\sin 30^\circ + F_{2,v} + V_2$$

for node 3 :

$$\sum F_H = 0 = -F_2 - F_3 \cos 60^\circ + F_{3,h}$$
$$\sum F_V = 0 = F_3 \sin 60^\circ + F_{3,V} + V_3$$

where $F_{i,h}$ is the external horizontal force applied to node i (where a positive force is from left to right) and $F_{i,v}$ is the external vertical force applied to node i (where a positive force is upward). Thus, in this problem, the $2000-$N downward force on node 1 corresponds to $F_{i,v} = -2000$. For this case, all other $F_{i,b}$'s and $F_{i\lambda h}$'s are zero. Express this set of linear algebraic equations in matrix form and then use MATLAB to solve for the unknowns.

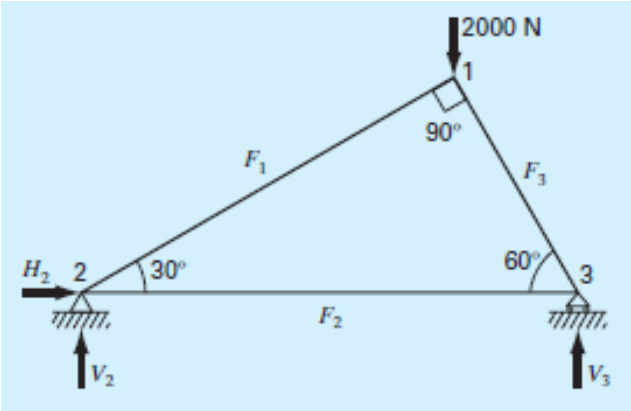

Figure 1.13: FIGURE P8.10

8.11 Consider the three mass-four spring system in Fig. P8.11. Determining the equations of motion from $\Sigma F_x = ma_x$ for each mass using its free-body diagram results in the following differential equations:

$$\ddot{x}_1 + \left(\frac{k_1+k_2}{m_1}\right)x_1 - \left(\frac{k_2}{m_1}\right)x_2 = 0$$
$$\ddot{x}_2 - \left(\frac{k_2}{m_2}\right)x_1 + \left(\frac{k_2+k_3}{m_2}\right)x_2 - \left(\frac{k_3}{m_2}\right)x_3 = 0$$
$$\ddot{x}_3 - \left(\frac{k_3}{m_3}\right)x_2 + \left(\frac{k_3+k_4}{m_3}\right)x_3 = 0$$

where $k_1 = k_4 = 10$ N/m, $k_2 = k_3 = 40$ N/m, and $m_1 = m_2 = m_3 = 1$ kg. The three equations can be written in matrix form: $0 = \{$ Acceleration vector $\} + [k/m$ matrix $]$ displacement vector $x\}$ At a specific time where $x_1 = 0.05$ m, $x_2 = 0.04$ m, and $x_3 = 0.03$ m, this forms a tridiagonal matrix. Use MATLAB to solve for the acceleration of each mass.

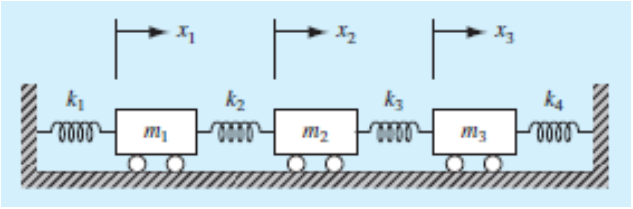

Figure 1.14: FIGURE P8.11

8.12 Perfom the same computation as in Example 8.2, but use five jumpers with the following characteristics:

Jumper	Mass (**kg**)	Spring Constant (**N/m**)	Unstretched Cord Length (**m**)
1	65	80	10
2	75	40	10
3	60	70	10
4	75	100	10
5	90	20	10

8.13 Three masses are suspended vertically by a series of identical springs where mass 1 is at the top and mass 3 is at the bottom. If $g = 9.81$ m/s^2, $m_1 = 2$ kg, $m_2 = 2.5$ kg, $m_3 = 3$ kg, and the k's $= 15$ kg/s^2, use MATLAB to solve for the displacements x.

8.14 Perform the same computation as in Sec. 8.3, but for the circuit in Fig. P8.14.

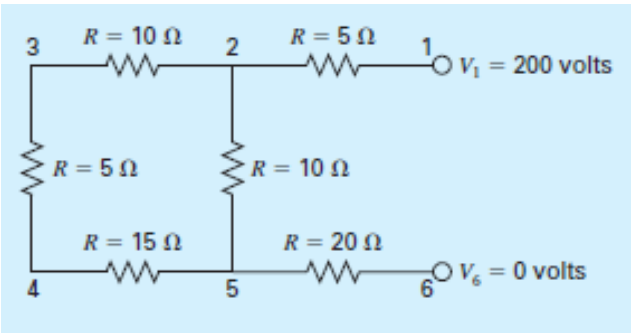

Figure 1.15: FIGURE P8.14

8.15 Perform the same computation as in Sec. 8.3, but for the circuit in Fig. P8.15.

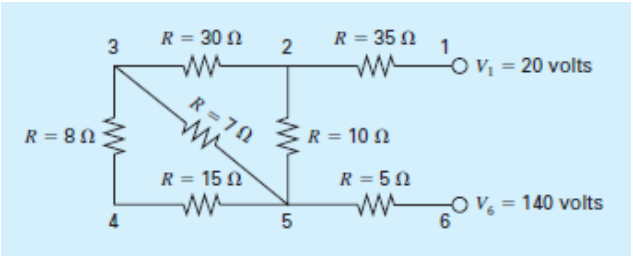

Figure 1.16: FIGURE P8.15