PART I

MATRIX THEORY

Matrix theory is a powerful field of mathematics that has found applications in the solution of several real-world problems, ranging from the solution of algebraic equations to the solution of differential equations. Its importance has also been enhanced by the rapid development of several computer programs that have improved the efficiency of matrix analysis and the solution of matrix equations.

We have allotted three chapters to discussing matrix theory. Chapter 1 contains the basic notations and operations. These include conventions and notations for the various structural, algebraic, differential, and integral operations. As such, this chapter focuses on how to formulate problems in terms of matrix equations, the various approaches of matrix algebraic manipulations, and matrix partitions.

Chapter 2 then focuses on the solution of the linear equation given by $A\mathbf{x} = \mathbf{b}$, and it includes both direct and indirect methods. The most direct method is to find the inverse of A and then evaluate $\mathbf{x} = A^{-1}\mathbf{b}$. However, the major practical issue is that matrix inverses become unwieldy when the matrices are large. This chapter is concerned with finding the solutions by reformulating the problem to take advantage of available matrix properties. Direct methods use various factorizations of A based on matrices that are more easily invertible, whereas indirect methods use an iterative process starting with an initial guess of the solution. The methods can then be applied to linear least-squares problems, as well as to the solution of multivariable nonlinear equations.

Chapter 3 focuses on matrices as operators. In this case, the discussion is concerned with the analysis of matrices, for example, using eigenvalues and eigenvectors. This allows one to obtain diagonalized matrices or Jordan canonical forms. These forms provide efficient tools for evaluating matrix functions, which are also very useful for solving simultaneous differential equations. Other analysis tools such as singular values decomposition, matrix norms, and condition numbers are also included in the chapter.

The matrix theory topics are also used in the other parts of this book. In Part II, we can use matrices to represent vector coordinates and tensors. The operations and vector/tensor properties can also be evaluated and analyzed efficiently using matrix theory. For instance, the mutual orthogonalities among the principal axes of a symmetric tensor are immediate consequences of the properties of matrix eigenvectors. In Part III, matrices are also shown to be indispensable tools for solving ordinary differential equations. Specifically, the solution and analysis of a set of simultaneous

linear ordinary differential equations can be represented in terms of matrix exponential functions. Moreover, numerical solution methods can now be coded in matrix forms. Finally, in Part IV of the book, both the finite difference and finite elements methods reduce partial differential equations to linear algebraic equations. Thus the tools discussed in Chapter 2 are strongly applicable because the matrices resulting from either of these methods will likely be large and sparse.

1 Matrix Algebra

In this chapter, we review some definitions and operations of matrices. Matrices play very important roles in the computation and analysis of several mathematical problems. They allow for compact notations of large sets of linear algebraic equations. Various matrix operations such as addition, multiplication, and inverses can be combined to find the required solutions in a more tractable manner. The existence of several software tools, such as MATLAB, have also made it very efficient to approach the solution by posing several problems in the form of matrix equations. Moreover, the matrices possess internal properties such as determinant, rank, trace, eigenvalues, and eigenvectors, which can help characterize the systems under consideration.

We begin with the basic notation and definitions in Section 1.1. The matrix notations introduced in this chapter are used throughout the book. Then in Section 1.2, we discuss the various matrix operations. Several matrix operations should be familiar to most readers, but some may not be as familiar, such as Kronecker products. We have classified the operations as either structural or algebraic. The structural operations are those operations that involve only the collection and arrangement of the elements. On the other hand, the algebraic operations pertain to those in which algebraic operations are implemented among the elements of a matrix or group of matrices. The properties of the different matrix operations such as associativity, commutativity, and distributivity properties are summarized in Section 1.3. In addition, we discuss the properties of determinants and include some matrix inverse formulas. The properties and formulas allow for the manipulation and simplification of matrix equations. These will be important tools used throughout this book.

In Section 1.4, we explore various block matrix operations. These operations are very useful when the structure of the matrices can be partitioned into submatrices. These block operations will also prove to be very useful when solving large sets of equations that exhibit a specific pattern.

From algebraic operations, we then move to topics involving differential and integral calculus in Section 1.5. We first define and fix various notations for the derivatives and integrals of matrices. These notations are also used throughout the book. The various properties of the matrix calculus operations are also summarized in this section. One of the applications of matrix calculus is optimization, in which the concept of positive (and negative) definiteness is needed for sufficient conditions. We

devote Section A.5 in the appendix to explaining positive or negative definiteness in more detail.

Finally, in Section 1.6, we include a brief discussion on sparse matrices. These matrices often result when the problem involves a large collection of smaller elements that are connected with only few of the other elements, such as when we solve differential equations by numerical methods, for example, the finite difference methods or finite element methods.

1.1 Definitions and Notations

The primary application of matrices is in solving simultaneous linear equations. These equations can come from solving problems based on mass and energy balance of physical, chemical, and biological processes; Kirchhoff's laws in electric circuits; force and moment balances in engineering structures; and so forth. The size of the unknowns for these problems can be quite large, so the solution can become quite complicated. This is especially the case with modern engineering systems, which typically contain several stages (e.g., staged operations in chemical engineering), are highly integrated (e.g., large-scale integration in microelectronics), or are structurally large (e.g., large power grids and large buildings). Matrix methods offer techniques that allow for tractability and computational efficiency.

When solving large nonlinear problems, numerical methods become a necessary approach. The numerical computations often involve matrix formulations. For instance, several techniques for solving nonlinear equations and nonlinear optimization problems implement Newton's method and other gradient-based methods, in which the calculations include matrix operations. Matrix equations also result from finite approximations of systems of differential equations. For boundary value problems, the internal values are to be solved such that both the boundary conditions and the differential equations that describe the systems are satisfied. Here, the numerical techniques include finite element methods and finite difference methods, both of which translate the problem back to a linear set of equations.

Aside from calculating the unknowns or solving differential equations, matrix methods are also useful in operator analysis and design. In this case, matrix equations are analyzed in terms of operators, inputs, and outputs. The matrices associated with the operators can be formulated to obtain the desired behavior. For example, if we want to move a 3D point $a = (x, y, z)$ to another position, say, $b = (\widehat{x}, \widehat{y}, \widehat{z})$, in a particular way, for instance, to move it radially outward or rotate it at specified degrees counterclockwise, then we can build matrices that would produce the desired effects. Conversely, for a system (mechanical, chemical, electrical, biological, etc.) that can be written in matrix forms (both in differential equations and algebraic equations), we can often isolate the matrices associated with system operations and use matrix analysis to explore the capabilities and behavior of the system.

It is also worth mentioning that, in addition to the classical systems that are modeled with algebraic and differential equations, there are other application domains that use matrix methods extensively. These include data processing, computational geometry, and network analysis. In data processing, matrix methods help in regression analysis and statistical data analysis. These applications also include data mining in search engines, bioinformatics, and computer security. Computational geometry also uses matrix methods to handle and analyze large sets of data. Applications include computer graphics and visualization, which are also used for pattern

recognition purposes. In network analysis, matrix methods are used together with graph theory to analyze the connectivity and effects of large, complex structures. Applications include the analysis of communication and control systems, as well as large power grids.

We now begin with the definition of a matrix and continue with some of the notations and conventions that are used throughout this book.

Definition 1.1. *A* **matrix** *is a collection of objects, called the* **elements** *of the matrix, arranged in rows and columns.*

These elements of the matrix could be numbers, such as

$$A = \begin{pmatrix} 1 & 0 & -0.3 \\ -2 & 3+i & \frac{1}{2} \end{pmatrix} \qquad \text{with } i = \sqrt{-1}$$

or functions, such as

$$B = \begin{pmatrix} 1 & 2x(t)+a \\ \int \sin(\omega t)dt & dy/dt \end{pmatrix}$$

The elements of matrices are restricted to a set of mathematical objects that allow algebraic binary operations such as addition, subtraction, multiplication, and division. The valid elements of the matrix are referred to as **scalars**. Note that a scalar is not the same as a matrix having only one row and one column.

We often use capital letters to denote matrices, whereas the corresponding small letters stand for the elements. Thus the elements of matrix A positioned at the ith row and jth column are denoted as a_{ij}, for example, for A having N rows and M columns,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{pmatrix} \tag{1.1}$$

The size of the matrix is given by the symbol "[=]", for example, for matrix A having N rows and M columns,

$$A\,[=]\,N \times M \qquad \text{or} \qquad A_{[N\times M]} \tag{1.2}$$

A **row vector** is a matrix having one row, whereas a **column vector** is a matrix having one column. The **length** of a vector means the number of elements of the row or column vector. If the type of vector has not been specified, we take it to mean a column vector. We often use bold small letters to denote vectors. A basic vector is the i^{th}**unit vector** of length N denoted by $\mathbf{e}_i$,

$$\mathbf{e}_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i^{\text{th}} \text{ element} \tag{1.3}$$

The length N of the unit vector is determined by context.

A **square matrix** is a matrix with the same number of columns and rows. Special cases include **lower triangular**, **upper triangular**, and **diagonal** matrices. Lower triangular matrices have zero elements above the main diagonal, whereas upper triangular matrices have zero elements below the main diagonal. Diagonal matrices have zero off-diagonal elements. The diagonal matrix is also represented by

$$D = \text{diag}\left(d_{11}, d_{22}, \ldots, d_{NN} \right) \tag{1.4}$$

A special diagonal matrix in which the main diagonal elements are all 1's is known as the **identity matrix**, denoted by I. If the size of the identity matrix needs to be specified, then we use I_N to denote an $N \times N$ identity matrix. An extensive list of different matrices that have special forms such as bidiagonal, tridiagonal, Hessenberg, Toeplitz, and so forth are given in Tables A.1 through A.5 in Section A.1 as an appendix for easy reference.

1.2 Fundamental Matrix Operations

We assume that the reader is already familiar with several matrix operations. The purpose of the following sections is to summarize these operations, introduce our notations, and relate them to some of the available MATLAB commands. We can divide matrix operations into two major categories. The first category involves the restructuring or combination of matrices. The second category includes the operations that contain algebraic computations such as addition, multiplication, and inverses.

1.2.1 Matrix Restructuring Operations

A list of matrix rearrangement operations with their respective notations are summarized in Tables 1.1 and 1.2 (together with some MATLAB commands associated with the operations).

The row and column augmentation operations are designated by horizontal and vertical bars, respectively. These are used extensively throughout the book because we take advantage of block matrix operations. The reshaping operations are given by the **vectorization** operation and **reshape** operation. Both these operations are quite useful when reformulating equations such as $HX + XB + CXD = F$ into the familiar linear equation form given by $A\mathbf{x} = \mathbf{b}$.

There are two operations that involve exchanging the roles of rows and columns: the standard **transpose** operation, which we denote by superscript T, and the **conjugate transpose**, which we denote by superscript asterisk. In general, $A^T \neq A^*$, except when the elements of A are all real. When $A = A^T$, we say that A is **symmetric**, and when $A = A^*$, we say that A is **Hermitian**. The two cases are generally not the same. For instance, let

$$A = \begin{pmatrix} 1+i & 2 \\ 2 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2+i \\ 2-i & 3 \end{pmatrix}$$

then A is symmetric but not Hermitian, whereas B is Hermitian but not symmetric. On the other hand, when $A = -A^T$, we say that A is **skew-symmetric**, and when $A = -A^*$, we say that A is **skew-Hermitian**.

Table 1.1. *Matrix restructuring operations*

	Operation	Notation	Rule
1	Column Augment	$C = \left(\begin{array}{c\|c} A & B \end{array} \right)$ MATLAB: `C=[A,B]`	$\begin{pmatrix} c_{11} & \cdots & c_{1,M+P} \\ \vdots & \ddots & \vdots \\ c_{N1} & \cdots & c_{N,M+P} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1M} & b_{11} & \cdots & b_{1P} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} & b_{N1} & \cdots & b_{NP} \end{pmatrix}$
2	Row Augment	$C = \begin{pmatrix} A \\ \hline B \end{pmatrix}$ MATLAB: `C=[A;B]`	$\begin{pmatrix} c_{11} & \cdots & c_{1,M} \\ \vdots & \ddots & \vdots \\ c_{N+P,1} & \cdots & c_{N+P,M} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \\ b_{11} & \cdots & b_{1M} \\ \vdots & \ddots & \vdots \\ b_{P1} & \cdots & b_{PM} \end{pmatrix}$
3	Vectorize	$C = \text{vec}(A)$ MATLAB: `C=A(:)`	$\begin{pmatrix} c_1 \\ \vdots \\ c_{N\cdot M} \end{pmatrix} = \begin{pmatrix} A_{\bullet,1} \\ \hline \vdots \\ \hline A_{\bullet,M} \end{pmatrix}$ where $A_{\bullet,i}$ is the i^{th} column of A

The **submatrix** operation is denoted by using a list of k subscript indices and ℓ superscript indices to refer to the rows and columns, respectively, extracted from a matrix. For instance,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \rightarrow \quad A^{2,3}_{1,2} = \begin{pmatrix} 2 & 3 \\ 5 & 6 \end{pmatrix}$$

For a square matrix, if the diagonals of the submatrix are a subset of the diagonals of the original matrix, then we call it a **principal submatrix**. This happens if the superscript indices and the subscript indices of the submatrix are the same. For instance,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \rightarrow \quad A^{1,3}_{1,3} = \begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix}$$

then $A^{1,3}_{1,3}$ is a principal submatrix.

Table 1.2. *Matrix restructuring operations*

	Operation	Notation	Rule
4	Reshape	$C =$ reshape $(\mathbf{v}, N, M)$ MATLAB: `reshape(v,N,M)`	$C = \begin{pmatrix} v_1 & v_{N+1} & & v_{(M-1)N+1} \\ \vdots & \vdots & \cdots & \vdots \\ v_N & v_{2N} & & v_{MN} \end{pmatrix}$ where $\mathbf{v}$ is a vector of length NM
5	Transpose	$C = A^T$ MATLAB: `C=A.'`	$C = \begin{pmatrix} a_{11} & \cdots & a_{M1} \\ \vdots & \ddots & \vdots \\ a_{1N} & \cdots & a_{MN} \end{pmatrix}$
6	Conjugate Transpose	$C = A^*$ MATLAB: `C=A'`	$C = \begin{pmatrix} \overline{a}_{11} & \cdots & \overline{a}_{M1} \\ \vdots & \ddots & \vdots \\ \overline{a}_{1N} & \cdots & \overline{a}_{MN} \end{pmatrix}$ where $\overline{a}_{ij} =$ complex conjugate of a_{ij}

7	Submatrix	$C = A^{j_1,j_2,\ldots,j_\ell}_{i_1,i_2,\ldots,i_k}$
		MATLAB:
		`rows=[i1,i2,...]`
		`cols=[j1,j2,...]`
		`C=A(rows,cols)`
			$C_{[\,k\times\ell\,]} = \begin{pmatrix} a_{i_1 j_1} & \cdots & a_{i_1 j_\ell} \\ \vdots & \ddots & \vdots \\ a_{i_k j_1} & \cdots & a_{i_k j_\ell} \end{pmatrix}$

8	$(ij)^{\text{th}}$ Redact	$C = A_{ij\downarrow}$
		MATLAB:
		`C=A`
		`C(i,:)=[ ]`
		`C(:,j)=[ ]`
			$C_{[\,(N-1)\times(M-1)\,]} =$ $\left(\begin{array}{c\|c} A^{1,\ldots,j-1,j+1,\ldots,M}_{1,\ldots,i-1,i+1,\ldots,N} & A^{1,\ldots,j-1,j+1,\ldots,M}_{i+1,\ldots,N} \\ \hline A^{1,\ldots,j-1,j+1,\ldots,M}_{1,\ldots,i-1} & A^{j+1,\ldots,M}_{i+1,\ldots,N} \end{array}\right)$

Next, the operation to remove some specified rows and columns is referred to here as the $(ij)^{\text{th}}$ **redact** operation. We use $A_{ij\downarrow}$ to denote the removal of the i^{th} row and j^{th} column. For instance,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \rightarrow \quad A_{23\downarrow} = \begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix} \tag{1.5}$$

This operation is useful in finding determinants, cofactors, and adjugates.

Table 1.3. *Matrix algebraic operations*

	Operation	Notation	Rule	MATLAB commands
1	Sum	$C = A + B$	$c_{ij} = a_{ij} + b_{ij}$	`C=A+B`
2	Scalar Product	$C = qA$	$c_{ij} = q\, a_{ij}$	`C=q*A`
3	Matrix Product	$C = AB$	$c_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj}$	`C=A*B`
4	Haddamard Product	$C = A \circ B$	$c_{ij} = a_{ij} b_{ij}$	`C=A.*B`
5	Kronecker Product (tensor product)	$C = A \otimes B$	$C = \left(\begin{array}{c\|c\|c} a_{11}B & \cdots & a_{1M}B \\ \hline \vdots & & \vdots \\ \hline a_{N1}B & \cdots & a_{NM}B \end{array}\right)$	`C=kron(A,B)`
6	Determinant	$q = \det(A)$ or $q = \|A\|$	$q = \sum_{K} \epsilon(K) \left(\prod_{i=1}^{N} a_{i,k_i}\right)$ see (1.10)	`q=det(A)`
7	Cofactor	$q = \mathbf{cof}\,(a_{ij})$	$q = (-1)^{i+j} \left\| A_{ij\downarrow} \right\|$	
8	Adjugate	$C = \mathbf{adj}\,(A)$	$c_{ij} = \mathbf{cof}\,(a_{ji})$	
9	Inverse	$C = A^{-1}$	$C = \frac{1}{\|A\|}\, \mathbf{adj}\,(A)$	`C=inv(A)`
10	Trace	$q = \mathbf{tr}\,(A)$	$q = \sum_{i=1}^{N} a_{ii}$	`q=trace(A)`
11	Real Part	$C = \mathbf{Real}(A)$	$c_{ij} = \mathrm{real}\,(a_{ij})$	`C=Real(A)`
12	Imag Part	$C = \mathbf{Imag}(A)$	$c_{ij} = \mathrm{imag}\,(a_{ij})$	`C=Imag(A)`
13	Complex Congugate	$C = \overline{A}$	$c_{ij} = \overline{a_{ij}}$	`C=Conj(A)`

1.2.2 Matrix Algebraic Operations

The matrix algebraic operations can be classified further as either binary or unary. For binary operations, the algebraic operations require two inputs, either a scalar and a matrix or two matrices of appropriate sizes. For unary operations, the input is a matrix, and the algebraic operations are applied on the elements of the matrix. The matrix algebraic operations are given in Table 1.3, together with their corresponding MATLAB commands.

1.2.2.1 Binary Algebraic Operations

The most basic matrix binary computational operations are matrix sums, scalar products, and matrix products, which are quite familiar to most readers. To see how these operations seem to be the natural consequences of solving simultaneous equations, we refer the reader to Section A.2 included in the appendices.

Matrix products of A and B are denoted simply by $C = AB$, which requires $A[=]N \times K$, $B[=]K \times M$ and $C[=]N \times M$ (i.e., the columns of A must be equal to the rows of B). If this is the case, we say that A and B are **conformable** for the operation AB. Furthermore, based on the sizes of the matrices, $A_{[N\times K]}B_{[K\times M]} = C_{[N\times M]}$, we see that dropping the common value K leaves the size of C to be $N \times M$. For the matrix product AB, we say that A **premultiplies** B, or B **postmultiplies** A. For instance, let

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ -1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -2 & 1 \\ -1 & 3 \end{pmatrix} \quad \text{then} \quad C = AB = \begin{pmatrix} -3 & 4 \\ -5 & 5 \\ 2 & -1 \end{pmatrix}$$

However, B and A is not conformable for the product BA.

In several cases, $AB \neq BA$, even if the reversed order is conformable, and thus one needs to be clear whether a matrix premultiplies or postmultiplies another matrix. For the special case in which switching the order yields the same product (i.e., $AB = BA$), then we say that A and B **commutes**. It is necessary that commuting matrices are square and have the same sizes.

We list a few key results regarding matrix products:

1. For matrix products between a matrix $A[=]N \times M$ and the appropriately sized identity matrix, we have

$$AI_M = I_N A = A$$

where I_M and I_N are identity matrices of size M and size N, respectively.

2. Based on the definition of matrix products, when B premultiplies A, the row elements of B are pairwise multiplied with the column elements of A, and the results are then summed together. This fact implies that to scale the i^{th} row of A by a factor d_i, we can simply premultiply A by a diagonal matrix $D = \text{diag}\,(d_1, \ldots, d_N)$. For instance,

$$DA = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 4 & 5 & 6 \\ -7 & -8 & -9 \end{pmatrix}$$

Likewise, to scale the j^{th} column of A by a factor d_j, we can simply postmultiply A by a diagonal matrix $D = \text{diag}\,(d_1, \ldots, d_N)$. For instance,

$$AD = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & -3 \\ 8 & 5 & -6 \\ 14 & 8 & -9 \end{pmatrix}$$