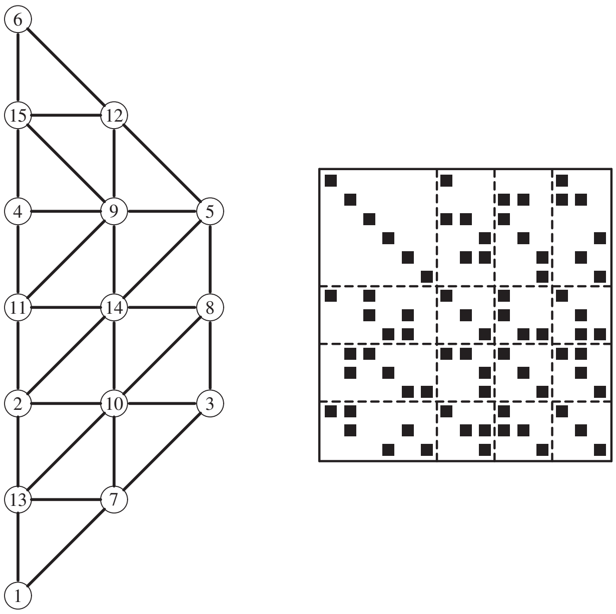
Iterative Methods *for Sparse* Linear Systems

Yousef Saad



SECOND EDITION WITH CORRECTIONS. JANUARY 3RD, 2000.

CONTENTS

PREFACE

Iterative methods for solving general, large sparse linear systems have been gaining popularity in many areas of scientific computing. Until recently, direct solution methods were often preferred to iterative methods in real applications because of their robustness and predictable behavior. However, a number of efficient iterative solvers were discovered and the increased need for solving very large linear systems triggered a noticeable and rapid shift toward iterative techniques in many applications.

This trend can be traced back to the 1960s and 1970s when two important developments revolutionized solution methods for large linear systems. First was the realization that one can take advantage of "sparsity" to design special direct methods that can be quite economical. Initiated by electrical engineers, these "direct sparse solution methods" led to the development of reliable and efficient general-purpose direct solution software codes over the next three decades. Second was the emergence of preconditioned conjugate gradient-like methods for solving linear systems. It was found that the combination of preconditioning and Krylov subspace iterations could provide efficient and simple "general-purpose" procedures that could compete with direct solvers. Preconditioning involves exploiting ideas from sparse direct solvers. Gradually, iterative methods started to approach the quality of direct solvers. In earlier times, iterative methods were often special-purpose in nature. They were developed with certain applications in mind, and their efficiency relied on many problem-dependent parameters.

Now, three-dimensional models are commonplace and iterative methods are almost mandatory. The memory and the computational requirements for solving three-dimensional Partial Differential Equations, or two-dimensional ones involving many degrees of freedom per point, may seriously challenge the most efficient direct solvers available today. Also, iterative methods are gaining ground because they are easier to implement efficiently on high-performance computers than direct methods.

My intention in writing this volume is to provide up-to-date coverage of iterative methods for solving large sparse linear systems. I focused the book on practical methods that work for general sparse matrices rather than for any specific class of problems. It is indeed becoming important to embrace applications not necessarily governed by Partial Differential Equations, as these applications are on the rise. Apart from two recent volumes by Axelsson [15] and Hackbusch [116], few books on iterative methods have appeared since the excellent ones by Varga [213]. and later Young [232]. Since then, researchers and practitioners have achieved remarkable progress in the development and use of effective iterative methods. Unfortunately, fewer elegant results have been discovered since the 1950s and 1960s. The field has moved in other directions. Methods have gained not only in efficiency but also in robustness and in generality. The traditional techniques which required rather complicated procedures to determine optimal acceleration parameters have yielded to the parameter-free conjugate gradient class of methods.

The primary aim of this book is to describe some of the best techniques available today, from both preconditioners and accelerators. One of the aims of the book is to provide a good mix of theory and practice. It also addresses some of the current research issues such as parallel implementations and robust preconditioners. The emphasis is on Krylov subspace methods, currently the most practical and common group of techniques used in applications. Although there is a tutorial chapter that covers the discretization of Partial Differential Equations, the book is not biased toward any specific application area. Instead, the matrices are assumed to be general sparse, possibly irregularly structured.

The book has been structured in four distinct parts. The first part, Chapters 1 to 4, presents the basic tools. The second part, Chapters 5 to 8, presents projection methods and Krylov subspace techniques. The third part, Chapters 9 and 10, discusses preconditioning. The fourth part, Chapters 11 to 13, discusses parallel implementations and parallel algorithms.

ACKNOWLEDGMENTS

I am grateful to a number of colleagues who proofread or reviewed different versions of the manuscript. Among them are Randy Bramley (University of Indiana at Bloomingtin), Xiao-Chuan Cai (University of Colorado at Boulder), Tony Chan (University of California at Los Angeles), Jane Cullum (IBM, Yorktown Heights), Alan Edelman (Massachussett Institute of Technology), Paul Fischer (Brown University), David Keyes (Old Dominion University), Beresford Parlett (University of California at Berkeley) and Shang-Hua Teng (University of Minnesota). Their numerous comments, corrections, and encouragements were a highly appreciated contribution. In particular, they helped improve the presentation considerably and prompted the addition of a number of topics missing from earlier versions.

This book evolved from several successive improvements of a set of lecture notes for the course “Iterative Methods for Linear Systems” which I taught at the University of Minnesota in the last few years. I apologize to those students who used the earlier error-laden and incomplete manuscripts. Their input and criticism contributed significantly to improving the manuscript. I also wish to thank those students at MIT (with Alan Edelman) and UCLA (with Tony Chan) who used this book in manuscript form and provided helpful feedback. My colleagues at the university of Minnesota, staff and faculty members, have helped in different ways. I wish to thank in particular Ahmed Sameh for his encouragements and for fostering a productive environment in the department. Finally, I am grateful to the National Science Foundation for their continued financial support of my research, part of which is represented in this work.

Yousef Saad

SUGGESTIONS FOR TEACHING

This book can be used as a text to teach a graduate-level course on iterative methods for linear systems. Selecting topics to teach depends on whether the course is taught in a mathematics department or a computer science (or engineering) department, and whether the course is over a semester or a quarter. Here are a few comments on the relevance of the topics in each chapter.

For a graduate course in a mathematics department, much of the material in Chapter 1 should be known already. For non-mathematics majors most of the chapter must be covered or reviewed to acquire a good background for later chapters. The important topics for the rest of the book are in Sections: 1.8.1, 1.8.3, 1.8.4, 1.9, 1.11. Section 1.12 is best treated at the beginning of Chapter 5. Chapter 2 is essentially independent from the rest and could be skipped altogether in a quarter course. One lecture on finite differences and the resulting matrices would be enough for a non-math course. Chapter 3 should make the student familiar with some implementation issues associated with iterative solution procedures for general sparse matrices. In a computer science or engineering department, this can be very relevant. For mathematicians, a mention of the graph theory aspects of sparse matrices and a few storage schemes may be sufficient. Most students at this level should be familiar with a few of the elementary relaxation techniques covered in Chapter 4. The convergence theory can be skipped for non-math majors. These methods are now often used as preconditioners and this may be the only motive for covering them.

Chapter 5 introduces key concepts and presents projection techniques in general terms. Non-mathematicians may wish to skip Section 5.2.3. Otherwise, it is recommended to start the theory section by going back to Section 1.12 on general definitions on projectors. Chapters 6 and 7 represent the heart of the matter. It is recommended to describe the first algorithms carefully and put emphasis on the fact that they generalize the one-dimensional methods covered in Chapter 5. It is also important to stress the optimality properties of those methods in Chapter 6 and the fact that these follow immediately from the properties of projectors seen in Section 1.12. When covering the algorithms in Chapter 7, it is crucial to point out the main differences between them and those seen in Chapter 6. The variants such as CGS, BICGSTAB, and TFQMR can be covered in a short time, omitting details of the algebraic derivations or covering only one of the three. The class of methods based on the normal equation approach, i.e., Chapter 8, can be skipped in a math-oriented course, especially in the case of a quarter system. For a semester course, selected topics may be Sections 8.1, 8.2, and 8.4.

Currently, preconditioning is known to be the critical ingredient in the success of iterative methods in solving real-life problems. Therefore, at least some parts of Chapter 9 and Chapter 10 should be covered. Section 9.2 and (very briefly) 9.3 are recommended. From Chapter 10, discuss the basic ideas in Sections 10.1 through 10.3. The rest could be skipped in a quarter course.

Chapter 11 may be useful to present to computer science majors, but may be skimmed or skipped in a mathematics or an engineering course. Parts of Chapter 12 could be taught primarily to make the students aware of the importance of "alternative" preconditioners. Suggested selections are: 12.2, 12.4, and 12.7.2 (for engineers). Chapter 13 presents an important research area and is primilarily geared to mathematics majors. Computer scientists or engineers may prefer to cover this material in less detail.

To make these suggestions more specific, the following two tables are offered as sample course outlines. Numbers refer to sections in the text. A semester course represents approximately 30 lectures of 75 minutes each whereas a quarter course is approximately 20 lectures of 75 minutes each. Different topics are selected for a mathematics course and a non-mathematics course.

Semester course		
Weeks	Mathematics	Computer Science / Eng.
1 – 3	1.9 –1.13 2.1 – 2.5 3.1 – 3.3, 3.7	1.1 – 1.6 (Read) 1.7 – 1.13, 2.1 – 2.2 3.1 – 3.7
4 – 6	4.1 – 4.3 5. 1 – 5.4 6.1 – 6.3	4.1 – 4.2 5.1 – 5.2.1 6.1 – 6.3
7 – 9	6.4 – 6.7 (Except 6.5.2) 6.9 – 6.11 7.1 – 7.3	6.4 – 6.5 (Except 6.5.5) 6.7.1, 6.8–6.9, 6.11.3. 7.1 – 7.3
10 – 12	7.4.1; 7.4.2 – 7.4.3 (Read) 8.1, 8.2, 8.4; 9.1 – 9.3 10.1 – 10.3	7.4.1; 7.4.2 – 7.4.3 (Read) 8.1 – 8.3; 9.1 – 9.3 10.1 – 10.4
13 – 15	10.5.1 – 10.5.6 10.6 ; 12.2 – 12.4 13.1 – 13.6	10.5.1 – 10.5.4 11.1 – 11.4 (Read); 11.5 – 11.6 12.1 – 12.2; 12.4 – 12.7

Quarter course		
Weeks	Mathematics	Computer Science / Eng.
1 – 2	1.9 – 1.13, 3.1 – 3.2 4.1 – 4.3	1.1 – 1.6 (Read); 3.1 – 3.7 4.1
3 – 4	5.1 – 5.4 6.1 – 6.4	5.1 – 5.2.1 6.1 – 6.3
5 – 6	6.4 – 6.7 (Except 6.5.2) 6.11, 7.1 – 7.3	6.4 – 6.5 (Except 6.5.5) 6.7.1, 6.11.3, 7.1 – 7.3
7 – 8	7.4.1; 7.4.2 – 7.4.3 (Read) 9.1 – 9.3; 10.1 – 10.3	7.4.1; 7.4.2 – 7.4.3 (Read) 9.1 – 9.3; 10.1 – 10.3
9 – 10	10.6 ; 12.2 – 12.4 13.1 – 13.4	11.1 – 11.4 (Read); 11.5 – 11.6 12.1 – 12.2; 12.4 – 12.7

CHAPTER

1

BACKGROUND IN LINEAR ALGEBRA

This chapter gives an overview of the relevant concepts in linear algebra which are useful in later chapters. It begins with a review of basic matrix theory and introduces the elementary notation used throughout the book. The convergence analysis of iterative methods requires a good level of knowledge in mathematical analysis and in linear algebra. Traditionally, many of the concepts presented specifically for these analyses have been geared toward matrices arising from the discretization of Partial Differential Equations and basic relaxation-type methods. These concepts are now becoming less important because of the trend toward projection-type methods which have more robust convergence properties and require different analysis tools. The material covered in this chapter will be helpful in establishing some theory for the algorithms and defining the notation used throughout the book.

1.1 MATRICES

For the sake of generality, all vector spaces considered in this chapter are complex, unless otherwise stated. A complex $n \times m$ matrix A is an $n \times m$ array of complex numbers

$$a_{ij}, \;\; i = 1, \ldots, n, \; j = 1, \ldots, m.$$

The set of all $n \times m$ matrices is a complex vector space denoted by $\mathbb{C}^{n \times m}$. The main operations with matrices are the following:

- Addition: $C = A + B$, where A, B, and C are matrices of size $n \times m$ and

$$c_{ij} = a_{ij} + b_{ij}, \quad i = 1, 2, \ldots n, \quad j = 1, 2, \ldots m.$$

- Multiplication by a scalar: $C = \alpha A$, where

$$c_{ij} = \alpha\, a_{ij}, \quad i = 1, 2, \ldots n, \quad j = 1, 2, \ldots m.$$

- Multiplication by another matrix:

$$C = AB,$$

where $A \in \mathbb{C}^{n\times m}, B \in \mathbb{C}^{m\times p}, C \in \mathbb{C}^{n\times p}$, and

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}.$$

Sometimes, a notation with column vectors and row vectors is used. The column vector a_{*j} is the vector consisting of the j-th column of A,

$$a_{*j} = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{pmatrix}.$$

Similarly, the notation a_{i*} will denote the i-th row of the matrix A

$$a_{i*} = (a_{i1}, a_{i2}, \ldots, a_{im}).$$

For example, the following could be written

$$A = (a_{*1}, a_{*2}, \ldots, a_{*m}),$$

or

$$A = \begin{pmatrix} a_{1*} \\ a_{2*} \\ . \\ . \\ a_{n*} \end{pmatrix}.$$

The *transpose* of a matrix A in $\mathbb{C}^{n\times m}$ is a matrix C in $\mathbb{C}^{m\times n}$ whose elements are defined by $c_{ij} = a_{ji}, i = 1, \ldots, m,\ j = 1, \ldots, n$. It is denoted by A^T. It is often more relevant to use the *transpose conjugate* matrix denoted by A^H and defined by

$$A^H = \bar{A}^T = \overline{A^T},$$

in which the bar denotes the (element-wise) complex conjugation.

Matrices are strongly related to linear mappings between vector spaces of finite dimension. This is because they represent these mappings with respect to two given bases: one for the initial vector space and the other for the image vector space, or *range* of A.

1.2 SQUARE MATRICES AND EIGENVALUES

A matrix is *square* if it has the same number of columns and rows, i.e., if $m = n$. An important square matrix is the identity matrix

$$I = \{\delta_{ij}\}_{i,j=1,\ldots,n},$$

where δ_{ij} is the Kronecker symbol. The identity matrix satisfies the equality $AI = IA = A$ for every matrix A of size n. The inverse of a matrix, when it exists, is a matrix C such that

$$CA = AC = I.$$

The inverse of A is denoted by A^{-1}.

The *determinant* of a matrix may be defined in several ways. For simplicity, the following recursive definition is used here. The determinant of a 1×1 matrix (a) is defined as the scalar a. Then the determinant of an $n \times n$ matrix is given by

$$\det(A) = \sum_{j=1}^{n} (-1)^{j+1} a_{1j} \det(A_{1j}),$$

where A_{1j} is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and the j-th column of A. A matrix is said to be *singular* when $\det(A) = 0$ and *nonsingular* otherwise. We have the following simple properties:

- $\det(AB) = \det(BA)$.
- $\det(A^T) = \det(A)$.
- $\det(\alpha A) = \alpha^n \det(A)$.
- $\det(\bar{A}) = \overline{\det(A)}$.
- $\det(I) = 1$.

From the above definition of determinants it can be shown by induction that the function that maps a given complex value λ to the value $p_A(\lambda) = \det(A - \lambda I)$ is a polynomial of degree n; see Exercise 8. This is known as the *characteristic polynomial* of the matrix A.

DEFINITION 1.1 *A complex scalar λ is called an eigenvalue of the square matrix A if a nonzero vector u of $\mathbb{C}^n$ exists such that $Au = \lambda u$. The vector u is called an eigenvector of A associated with λ. The set of all the eigenvalues of A is called the spectrum of A and is denoted by $\sigma(A)$.*

A scalar λ is an eigenvalue of A if and only if $\det(A - \lambda I) \equiv p_A(\lambda) = 0$. That is true *if and only if* (*iff* thereafter) λ is a root of the characteristic polynomial. In particular, there are at most n distinct eigenvalues.

It is clear that a matrix is singular if and only if it admits zero as an eigenvalue. A well known result in linear algebra is stated in the following proposition.

PROPOSITION 1.1 *A matrix A is nonsingular if and only if it admits an inverse.*

Thus, the determinant of a matrix determines whether or not the matrix admits an inverse.

The maximum modulus of the eigenvalues is called *spectral radius* and is denoted by $\rho(A)$

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

The *trace* of a matrix is equal to the sum of all its diagonal elements

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}.$$

It can be easily shown that the trace of A is also equal to the sum of the eigenvalues of A counted with their multiplicities as roots of the characteristic polynomial.

PROPOSITION 1.2 *If λ is an eigenvalue of A, then $\bar{\lambda}$ is an eigenvalue of A^H. An eigenvector v of A^H associated with the eigenvalue $\bar{\lambda}$ is called a left eigenvector of A.*

When a distinction is necessary, an eigenvector of A is often called a right eigenvector. Therefore, the eigenvalue λ as well as the right and left eigenvectors, u and v, satisfy the relations

$$Au = \lambda u, \quad v^H A = \lambda v^H,$$

or, equivalently,

$$u^H A^H = \bar{\lambda} u^H, \quad A^H v = \bar{\lambda} v.$$

1.3 TYPES OF MATRICES

The choice of a method for solving linear systems will often depend on the structure of the matrix A. One of the most important properties of matrices is symmetry, because of its impact on the eigenstructure of A. A number of other classes of matrices also have particular eigenstructures. The most important ones are listed below:

- *Symmetric matrices:* $A^T = A$.
- *Hermitian matrices:* $A^H = A$.
- *Skew-symmetric matrices:* $A^T = -A$.
- *Skew-Hermitian matrices:* $A^H = -A$.
- *Normal matrices:* $A^H A = AA^H$.
- *Nonnegative matrices:* $a_{ij} \geq 0,\ i, j = 1, \ldots, n$ (similar definition for nonpositive, positive, and negative matrices).
- *Unitary matrices:* $Q^H Q = I$.

It is worth noting that a unitary matrix Q is a matrix whose inverse is its transpose conjugate Q^H, since

$$Q^H Q = I \quad \rightarrow \quad Q^{-1} = Q^H. \tag{1.1}$$

A matrix Q such that $Q^H Q$ is diagonal is often called orthogonal.

Some matrices have particular structures that are often convenient for computational purposes. The following list, though incomplete, gives an idea of these special matrices which play an important role in numerical analysis and scientific computing applications.

- *Diagonal matrices:* $a_{ij} = 0$ for $j \neq i$. Notation:

$$A = \text{diag}\ (a_{11}, a_{22}, \ldots, a_{nn})\,.$$

- *Upper triangular matrices:* $a_{ij} = 0$ for $i > j$.
- *Lower triangular matrices:* $a_{ij} = 0$ for $i < j$.
- *Upper bidiagonal matrices:* $a_{ij} = 0$ for $j \neq i$ or $j \neq i + 1$.
- *Lower bidiagonal matrices:* $a_{ij} = 0$ for $j \neq i$ or $j \neq i - 1$.
- *Tridiagonal matrices:* $a_{ij} = 0$ for any pair i, j such that $|j - i| > 1$. Notation:

$$A = \text{tridiag}\ (a_{i,i-1}, a_{ii}, a_{i,i+1})\,.$$

- *Banded matrices:* $a_{ij} \neq 0$ only if $i - m_l \leq j \leq i + m_u$, where m_l and m_u are two nonnegative integers. The number $m_l + m_u + 1$ is called the bandwidth of A.
- *Upper Hessenberg matrices:* $a_{ij} = 0$ for any pair i, j such that $i > j + 1$. Lower Hessenberg matrices can be defined similarly.
- *Outer product matrices:* $A = uv^H$, where both u and v are vectors.
- *Permutation matrices:* the columns of A are a permutation of the columns of the identity matrix.
- *Block diagonal matrices:* generalizes the diagonal matrix by replacing each diagonal entry by a matrix. Notation:

$$A = \text{diag}\ (A_{11}, A_{22}, \ldots, A_{nn})\,.$$

- *Block tridiagonal matrices:* generalizes the tridiagonal matrix by replacing each nonzero entry by a square matrix. Notation:

$$A = \text{tridiag}\ (A_{i,i-1}, A_{ii}, A_{i,i+1})\,.$$

The above properties emphasize structure, i.e., positions of the nonzero elements with respect to the zeros. Also, they assume that there are many zero elements or that the matrix is of low rank. This is in contrast with the classifications listed earlier, such as symmetry or normality.

1.4 VECTOR INNER PRODUCTS AND NORMS

An inner product on a (complex) vector space $\mathbb{X}$ is any mapping s from $\mathbb{X} \times \mathbb{X}$ into $\mathbb{C}$,

$$x \in \mathbb{X}, y \in \mathbb{X} \quad \rightarrow \quad s(x,y) \in \mathbb{C},$$

which satisfies the following conditions:

1. $s(x,y)$ is linear with respect to x, i.e.,

$$s(\lambda_1 x_1 + \lambda_2 x_2, y) = \lambda_1 s(x_1,y) + \lambda_2 s(x_2,y), \quad \forall\, x_1, x_2 \in \mathbb{X}, \forall\, \lambda_1, \lambda_2 \in \mathbb{C}.$$

2. $s(x,y)$ is *Hermitian*, i.e.,

$$s(y,x) = \overline{s(x,y)}, \quad \forall\, x, y \in \mathbb{X}.$$

3. $s(x,y)$ is *positive definite*, i.e.,

$$s(x,x) > 0, \quad \forall\, x \neq 0.$$

Note that (2) implies that $s(x,x)$ is real and therefore, (3) adds the constraint that $s(x,x)$ must also be positive for any nonzero x. For any x and y,

$$s(x,0) = s(x, 0.y) = 0.s(x,y) = 0.$$

Similarly, $s(0,y) = 0$ for any y. Hence, $s(0,y) = s(x,0) = 0$ for any x and y. In particular the condition (3) can be rewritten as

$$s(x,x) \geq 0 \quad \text{and} \quad s(x,x) = 0 \quad \text{iff} \quad x = 0,$$

as can be readily shown. A useful relation satisfied by any inner product is the so-called Cauchy-Schwartz inequality:

$$|s(x,y)|^2 \leq s(x,x)\, s(y,y). \tag{1.2}$$

The proof of this inequality begins by expanding $s(x-\lambda y, x-\lambda y)$ using the properties of s,

$$s(x-\lambda y, x-\lambda y) = s(x,x) - \bar{\lambda} s(x,y) - \lambda s(y,x) + |\lambda|^2 s(y,y).$$

If $y = 0$ then the inequality is trivially satisfied. Assume that $y \neq 0$ and take $\lambda = s(x,y)/s(y,y)$. Then $s(x-\lambda y, x-\lambda y) \geq 0$ shows the above equality

$$\begin{aligned} 0 \leq s(x-\lambda y, x-\lambda y) &= s(x,x) - 2\frac{|s(x,y)|^2}{s(y,y)} + \frac{|s(x,y)|^2}{s(y,y)} \\ &= s(x,x) - \frac{|s(x,y)|^2}{s(y,y)}, \end{aligned}$$

which yields the result (1.2).

In the particular case of the vector space $\mathbb{X} = \mathbb{C}^n$, a "canonical" inner product is the *Euclidean inner product*. The Euclidean inner product of two vectors $x = (x_i)_{i=1,\ldots,n}$ and

$y = (y_i)_{i=1,\ldots,n}$ of $\mathbb{C}^n$ is defined by

$$(x, y) = \sum_{i=1}^{n} x_i \bar{y}_i, \tag{1.3}$$

which is often rewritten in matrix notation as

$$(x, y) = y^H x. \tag{1.4}$$

It is easy to verify that this mapping does indeed satisfy the three conditions required for inner products, listed above. A fundamental property of the Euclidean inner product in matrix computations is the simple relation

$$(Ax, y) = (x, A^H y), \quad \forall\, x, y \in \mathbb{C}^n. \tag{1.5}$$

The proof of this is straightforward. The *adjoint* of A *with respect to an arbitrary inner product* is a matrix B such that $(Ax, y) = (x, By)$ for all pairs of vectors x and y. A matrix is *self-adjoint*, or Hermitian with respect to this inner product, if it is equal to its adjoint.

The following proposition is a consequence of the equality (1.5).

PROPOSITION 1.3 *Unitary matrices preserve the Euclidean inner product, i.e.,*

$$(Qx, Qy) = (x, y)$$

for any unitary matrix Q and any vectors x and y.

Proof. Indeed, $(Qx, Qy) = (x, Q^H Qy) = (x, y)$. ∎

A vector norm on a vector space $\mathbb{X}$ is a real-valued function $x \to \|x\|$ on $\mathbb{X}$, which satisfies the following three conditions:

1. $\|x\| \geq 0, \quad \forall\, x \in \mathbb{X}, \quad$ and $\quad \|x\| = 0$ iff $x = 0$.
2. $\|\alpha x\| = |\alpha| \|x\|, \quad \forall\, x \in \mathbb{X}, \quad \forall \alpha \in \mathbb{C}$.
3. $\|x + y\| \leq \|x\| + \|y\|, \quad \forall\, x, y \in \mathbb{X}$.

For the particular case when $\mathbb{X} = \mathbb{C}^n$, we can associate with the inner product (1.3) the *Euclidean norm* of a complex vector defined by

$$\|x\|_2 = (x, x)^{1/2}.$$

It follows from Proposition 1.3 that a unitary matrix preserves the Euclidean norm metric, i.e.,

$$\|Qx\|_2 = \|x\|_2, \ \forall\, x.$$

The linear transformation associated with a unitary matrix Q is therefore an *isometry*.

The most commonly used vector norms in numerical linear algebra are special cases of the Hölder norms

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \tag{1.6}$$

Note that the limit of $\|x\|_p$ when p tends to infinity exists and is equal to the maximum modulus of the x_i's. This defines a norm denoted by $\|.\|_\infty$. The cases $p = 1$, $p = 2$, and $p = \infty$ lead to the most important norms in practice,

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|,$$
$$\|x\|_2 = \left[|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right]^{1/2},$$
$$\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

The Cauchy-Schwartz inequality of (1.2) becomes

$$|(x, y)| \le \|x\|_2 \|y\|_2.$$

1.5 MATRIX NORMS

For a general matrix A in $\mathbb{C}^{n\times m}$, we define the following special set of norms

$$\|A\|_{pq} = \max_{x\in\mathbb{C}^m,\ x\neq 0} \frac{\|Ax\|_p}{\|x\|_q}. \tag{1.7}$$

The norm $\|.\|_{pq}$ is *induced* by the two norms $\|.\|_p$ and $\|.\|_q$. These norms satisfy the usual properties of norms, i.e.,

$$\|A\| \ge 0, \quad \forall\, A \in \mathbb{C}^{n\times m}, \quad \text{and} \quad \|A\| = 0 \quad \text{iff} \quad A = 0$$
$$\|\alpha A\| = |\alpha| \|A\|, \forall\, A \in \mathbb{C}^{n\times m}, \quad \forall\, \alpha \in \mathbb{C}$$
$$\|A + B\| \le \|A\| + \|B\|, \quad \forall\, A, B \in \mathbb{C}^{n\times m}.$$

The most important cases are again those associated with $p, q = 1, 2, \infty$. The case $q = p$ is of particular interest and the associated norm $\|.\|_{pq}$ is simply denoted by $\|.\|_p$ and called a "p-norm." A fundamental property of a p-norm is that

$$\|AB\|_p \le \|A\|_p \|B\|_p,$$

an immediate consequence of the definition (1.7). Matrix norms that satisfy the above property are sometimes called *consistent*. A result of consistency is that for any square matrix A,

$$\|A^k\|_p \le \|A\|_p^k.$$

In particular the matrix A^k converges to zero if *any* of its p-norms is less than 1.

The Frobenius norm of a matrix is defined by

$$\|A\|_F = \left(\sum_{j=1}^{m}\sum_{i=1}^{n} |a_{ij}|^2\right)^{1/2}. \tag{1.8}$$

This can be viewed as the 2-norm of the column (or row) vector in $\mathbb{C}^{n^2}$ consisting of all the columns (respectively rows) of A listed from 1 to m (respectively 1 to n.) It can be shown that this norm is also consistent, in spite of the fact that it is not induced by a pair of vector norms, i.e., it is not derived from a formula of the form (1.7); see Exercise 5. However, it does not satisfy some of the other properties of the p-norms. For example, the Frobenius norm of the identity matrix is not equal to one. To avoid these difficulties, *we will only use the term matrix norm for a norm that is induced by two norms as in the definition (1.7).* Thus, we will not consider the Frobenius norm to be a proper matrix norm, according to our conventions, even though it is consistent.

The following equalities satisfied by the matrix norms defined above lead to alternative definitions that are often easier to work with:

$$\|A\|_1 = \max_{j=1,\ldots,m} \sum_{i=1}^{n} |a_{ij}|, \tag{1.9}$$

$$\|A\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|, \tag{1.10}$$

$$\|A\|_2 = \left[\rho(A^H A)\right]^{1/2} = \left[\rho(AA^H)\right]^{1/2}, \tag{1.11}$$

$$\|A\|_F = \left[\mathrm{tr}(A^H A)\right]^{1/2} = \left[\mathrm{tr}(AA^H)\right]^{1/2}. \tag{1.12}$$

As will be shown later, the eigenvalues of $A^H A$ are nonnegative. Their square roots are called *singular values* of A and are denoted by $\sigma_i, i = 1, \ldots, m$. Thus, the relation (1.11) states that $\|A\|_2$ is equal to σ_1, the largest singular value of A.

Example 1.1 From the relation (1.11), it is clear that the spectral radius $\rho(A)$ is equal to the 2-norm of a matrix when the matrix is Hermitian. However, it is not a matrix norm in general. For example, the first property of norms is not satisfied, since for

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

we have $\rho(A) = 0$ while $A \neq 0$. Also, the triangle inequality is not satisfied for the pair A, and $B = A^T$ where A is defined above. Indeed,

$$\rho(A + B) = 1 \quad \text{while} \quad \rho(A) + \rho(B) = 0.$$

1.6 SUBSPACES, RANGE, AND KERNEL

A subspace of $\mathbb{C}^n$ is a subset of $\mathbb{C}^n$ that is also a complex vector space. The set of all linear combinations of a set of vectors $G = \{a_1, a_2, \ldots, a_q\}$ of $\mathbb{C}^n$ is a vector subspace called the linear span of G,

$$\mathrm{span}\{G\} = \mathrm{span}\,\{a_1, a_2, \ldots, a_q\}$$

$$= \left\{ z \in \mathbb{C}^n \;\middle|\; z = \sum_{i=1}^{q} \alpha_i a_i;\ \{\alpha_i\}_{i=1,\ldots,q} \in \mathbb{C}^q \right\}.$$

If the a_i's are linearly independent, then each vector of $\mathrm{span}\{G\}$ admits a unique expression as a linear combination of the a_i's. The set G is then called a *basis* of the subspace $\mathrm{span}\{G\}$.

Given two vector subspaces S_1 and S_2, their *sum* S is a subspace defined as the set of all vectors that are equal to the sum of a vector of S_1 and a vector of S_2. The intersection of two subspaces is also a subspace. If the intersection of S_1 and S_2 is reduced to $\{0\}$, then the sum of S_1 and S_2 is called their direct sum and is denoted by $S = S_1 \bigoplus S_2$. When S is equal to $\mathbb{C}^n$, then every vector x of $\mathbb{C}^n$ can be written in a unique way as the sum of an element x_1 of S_1 and an element x_2 of S_2. The transformation P that maps x into x_1 is a linear transformation that is *idempotent*, i.e., such that $P^2 = P$. It is called a *projector* onto S_1 along S_2.

Two important subspaces that are associated with a matrix A of $\mathbb{C}^{n\times m}$ are its *range*, defined by

$$\mathrm{Ran}(A) = \{Ax \mid x \in \mathbb{C}^m\}, \tag{1.13}$$

and its *kernel* or *null space*

$$\mathrm{Ker}(A) = \{x \in \mathbb{C}^m \mid Ax = 0\}.$$

The range of A is clearly equal to the linear *span* of its columns. The *rank* of a matrix is equal to the dimension of the range of A, i.e., to the number of linearly independent columns. This *column rank* is equal to the *row rank*, the number of linearly independent rows of A. A matrix in $\mathbb{C}^{n\times m}$ is of *full rank* when its rank is equal to the smallest of m and n.

A subspace S is said to be *invariant* under a (square) matrix A whenever $AS \subset S$. In particular for any eigenvalue λ of A the subspace $\mathrm{Ker}(A - \lambda I)$ is invariant under A. The subspace $\mathrm{Ker}(A - \lambda I)$ is called the eigenspace associated with λ and consists of all the eigenvectors of A associated with λ, in addition to the zero-vector.

1.7 ORTHOGONAL VECTORS AND SUBSPACES

A set of vectors $G = \{a_1, a_2, \ldots, a_r\}$ is said to be *orthogonal* if

$$(a_i, a_j) = 0 \quad \text{when} \quad i \neq j.$$

It is *orthonormal* if, in addition, every vector of G has a 2-norm equal to unity. A vector that is orthogonal to all the vectors of a subspace S is said to be orthogonal to this subspace. The set of all the vectors that are orthogonal to S is a vector subspace called the *orthogonal complement* of S and denoted by $S^\perp$. The space $\mathbb{C}^n$ is the direct sum of S and its orthogonal complement. Thus, any vector x can be written in a unique fashion as the sum of a vector in S and a vector in $S^\perp$. The operator which maps x into its component in the subspace S is the *orthogonal projector* onto S.

Every subspace admits an orthonormal basis which is obtained by taking any basis and "orthonormalizing" it. The orthonormalization can be achieved by an algorithm known as the Gram-Schmidt process which we now describe. Given a set of linearly independent vectors $\{x_1, x_2, \ldots, x_r\}$, first normalize the vector x_1, which means divide it by its 2-norm, to obtain the scaled vector q_1 of norm unity. Then x_2 is orthogonalized against the vector q_1 by subtracting from x_2 a multiple of q_1 to make the resulting vector orthogonal to q_1, i.e.,

$$x_2 \leftarrow x_2 - (x_2, q_1)q_1.$$

The resulting vector is again normalized to yield the second vector q_2. The i-th step of the Gram-Schmidt process consists of orthogonalizing the vector x_i against all previous vectors q_j.

ALGORITHM 1.1: Gram-Schmidt

1. *Compute* $r_{11} := \|x_1\|_2$. *If* $r_{11} = 0$ *Stop, else compute* $q_1 := x_1/r_{11}$.
2. *For* $j = 2, \ldots, r$ *Do:*
3. *Compute* $r_{ij} := (x_j, q_i)$, *for* $i = 1, 2, \ldots, j-1$
4. $\hat{q} := x_j - \sum_{i=1}^{j-1} r_{ij} q_i$
5. $r_{jj} := \|\hat{q}\|_2$,
6. *If* $r_{jj} = 0$ *then Stop, else* $q_j := \hat{q}/r_{jj}$
7. *EndDo*

It is easy to prove that the above algorithm will not break down, i.e., all r steps will be completed if and only if the set of vectors $x_1, x_2, \ldots, x_r$ is linearly independent. From lines 4 and 5, it is clear that at every step of the algorithm the following relation holds:

$$x_j = \sum_{i=1}^{j} r_{ij} q_i.$$

If $X = [x_1, x_2, \ldots, x_r]$, $Q = [q_1, q_2, \ldots, q_r]$, and if R denotes the $r \times r$ upper triangular matrix whose nonzero elements are the r_{ij} defined in the algorithm, then the above relation can be written as

$$X = QR. \tag{1.14}$$

This is called the QR decomposition of the $n \times r$ matrix X. From what was said above, the QR decomposition of a matrix exists whenever the column vectors of X form a linearly independent set of vectors.

The above algorithm is the standard Gram-Schmidt process. There are alternative formulations of the algorithm which have better numerical properties. The best known of these is the Modified Gram-Schmidt (MGS) algorithm.

ALGORITHM 1.2: Modified Gram-Schmidt

1. *Define* $r_{11} := \|x_1\|_2$. *If* $r_{11} = 0$ *Stop, else* $q_1 := x_1/r_{11}$.
2. *For* $j = 2, \ldots, r$ *Do:*

3. *Define* $\hat{q} := x_j$
4. *For* $i = 1, \ldots, j-1$, *Do:*
5. $r_{ij} := (\hat{q}, q_i)$
6. $\hat{q} := \hat{q} - r_{ij} q_i$
7. *EndDo*
8. *Compute* $r_{jj} := \|\hat{q}\|_2$,
9. *If* $r_{jj} = 0$ *then Stop, else* $q_j := \hat{q}/r_{jj}$
10. *EndDo*

Yet another alternative for orthogonalizing a sequence of vectors is the Householder algorithm. This technique uses Householder *reflectors*, i.e., matrices of the form

$$P = I - 2ww^T, \tag{1.15}$$

in which w is a vector of 2-norm unity. Geometrically, the vector Px represents a mirror image of x with respect to the hyperplane $\mathrm{span}\{w\}^\perp$.

To describe the Householder orthogonalization process, the problem can be formulated as that of finding a QR factorization of a given $n \times m$ matrix X. For any vector x, the vector w for the Householder transformation (1.15) is selected in such a way that

$$Px = \alpha e_1,$$

where α is a scalar. Writing $(I - 2ww^T)x = \alpha e_1$ yields

$$2w^T x \, w = x - \alpha e_1. \tag{1.16}$$

This shows that the desired w is a multiple of the vector $x - \alpha e_1$,

$$w = \pm \frac{x - \alpha e_1}{\|x - \alpha e_1\|_2}.$$

For (1.16) to be satisfied, we must impose the condition

$$2(x - \alpha e_1)^T x = \|x - \alpha e_1\|_2^2$$

which gives $2(\|x\|_1^2 - \alpha\xi_1) = \|x\|_2^2 - 2\alpha\xi_1 + \alpha^2$, where $\xi_1 \equiv e_1^T x$ is the first component of the vector x. Therefore, it is necessary that

$$\alpha = \pm\|x\|_2.$$

In order to avoid that the resulting vector w be small, it is customary to take

$$\alpha = -\mathrm{sign}(\xi_1)\|x\|_2,$$

which yields

$$w = \frac{x + \mathrm{sign}(\xi_1)\|x\|_2 e_1}{\|x + \mathrm{sign}(\xi_1)\|x\|_2 e_1\|_2}. \tag{1.17}$$

Given an $n \times m$ matrix, its first column can be transformed to a multiple of the column e_1, by premultiplying it by a Householder matrix P_1,

$$X_1 \equiv P_1 X, \qquad X_1 e_1 = \alpha e_1.$$

Assume, inductively, that the matrix X has been transformed in $k-1$ successive steps into the partially upper triangular form

$$X_k \equiv P_{k-1} \dots P_1 X_1 = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & \cdots & \cdots & x_{1m} \\ & x_{22} & x_{23} & \cdots & \cdots & \cdots & x_{2m} \\ & & x_{33} & \cdots & \cdots & \cdots & x_{3m} \\ & & & \ddots & \cdots & \cdots & \vdots \\ & & & & x_{kk} & \cdots & \vdots \\ & & & & x_{k+1,k} & \cdots & x_{k+1,m} \\ & & & & \vdots & \vdots & \vdots \\ & & & & x_{n,k} & \cdots & x_{n,m} \end{pmatrix}.$$

This matrix is upper triangular up to column number $k-1$. To advance by one step, it must be transformed into one which is upper triangular up the k-th column, leaving the previous columns in the same form. To leave the first $k-1$ columns unchanged, select a w vector which has zeros in positions 1 through $k-1$. So the next Householder reflector matrix is defined as

$$P_k = I - 2w_k w_k^T, \tag{1.18}$$

in which the vector w_k is defined as

$$w_k = \frac{z}{\|z\|_2}, \tag{1.19}$$

where the components of the vector z are given by

$$z_i = \begin{cases} 0 & \text{if} \quad i < k \\ \beta + x_{ii} & \text{if} \quad i = k \\ x_{ik} & \text{if} \quad i > k \end{cases} \tag{1.20}$$

with

$$\beta = \operatorname{sign}(x_{kk}) \times \left(\sum_{i=k}^{n} x_{ik}^2 \right)^{1/2}. \tag{1.21}$$

We note in passing that the premultiplication of a matrix X by a Householder transform requires only a rank-one update since,

$$(I - 2ww^T)X = X - wv^T \quad \text{where} \quad v = 2X^T w.$$

Therefore, the Householder matrices need not, and should not, be explicitly formed. In addition, the vectors w need not be explicitly scaled.

Assume now that $m-1$ Householder transforms have been applied to a certain matrix

X of dimension $n \times m$, to reduce it into the upper triangular form,

$$X_m \equiv P_{m-1}P_{m-2}\dots P_1 X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ & x_{22} & x_{23} & \cdots & x_{2m} \\ & & x_{33} & \cdots & x_{3m} \\ & & & \ddots & \vdots \\ & & & & x_{m,m} \\ & & & & 0 \\ & & & & \vdots \\ & & & & \vdots \end{pmatrix}. \tag{1.22}$$

Recall that our initial goal was to obtain a QR factorization of X. We now wish to recover the Q and R matrices from the P_k's and the above matrix. If we denote by P the product of the P_i on the left-side of (1.22), then (1.22) becomes

$$PX = \begin{pmatrix} R \\ O \end{pmatrix}, \tag{1.23}$$

in which R is an $m \times m$ upper triangular matrix, and O is an $(n-m) \times m$ zero block. Since P is unitary, its inverse is equal to its transpose and, as a result,

$$X = P^T \begin{pmatrix} R \\ O \end{pmatrix} = P_1 P_2 \dots P_{m-1} \begin{pmatrix} R \\ O \end{pmatrix}.$$

If E_m is the matrix of size $n \times m$ which consists of the first m columns of the identity matrix, then the above equality translates into

$$X = P^T E_m R.$$

The matrix $Q = P^T E_m$ represents the m first columns of P^T. Since

$$Q^T Q = E_m^T P P^T E_m = I,$$

Q and R are the matrices sought. In summary,

$$X = QR,$$

in which R is the triangular matrix obtained from the Householder reduction of X (see (1.22) and (1.23)) and

$$Qe_j = P_1 P_2 \dots P_{m-1} e_j.$$

ALGORITHM 1.3: Householder Orthogonalization

1. *Define* $X = [x_1, \dots, x_m]$
2. *For* $k = 1, \dots, m$ *Do:*
3. *If* $k > 1$ *compute* $r_k := P_{k-1}P_{k-2}\dots P_1 x_k$
4. *Compute* w_k *using (1.19), (1.20), (1.21)*
5. *Compute* $r_k := P_k r_k$ *with* $P_k = I - 2w_k w_k^T$
6. *Compute* $q_k = P_1 P_2 \dots P_k e_k$
7. *EndDo*

Note that line 6 can be omitted since the q_i are not needed in the execution of the next steps. It must be executed only when the matrix Q is needed at the completion of the algorithm. Also, the operation in line 5 consists only of zeroing the components $k+1,\ldots,n$ and updating the k-th component of r_k. In practice, a work vector can be used for r_k and its nonzero components after this step can be saved into an upper triangular matrix. Since the components 1 through k of the vector w_k are zero, the upper triangular matrix R can be saved in those zero locations which would otherwise be unused.

1.8 CANONICAL FORMS OF MATRICES

This section discusses the reduction of square matrices into matrices that have simpler forms, such as diagonal, bidiagonal, or triangular. Reduction means a transformation that preserves the eigenvalues of a matrix.

DEFINITION 1.2 *Two matrices A and B are said to be similar if there is a nonsingular matrix X such that*

$$A = XBX^{-1}.$$

The mapping $B \to A$ is called a similarity transformation.

It is clear that *similarity* is an equivalence relation. Similarity transformations preserve the eigenvalues of matrices. An eigenvector u_B of B is transformed into the eigenvector $u_A = Xu_B$ of A. In effect, a similarity transformation amounts to representing the matrix B in a different basis.

We now introduce some terminology.

1. An eigenvalue λ of A has *algebraic multiplicity* μ, if it is a root of multiplicity μ of the characteristic polynomial.
2. If an eigenvalue is of algebraic multiplicity one, it is said to be *simple*. A nonsimple eigenvalue is *multiple*.
3. The *geometric multiplicity* γ of an eigenvalue λ of A is the maximum number of independent eigenvectors associated with it. In other words, the geometric multiplicity γ is the dimension of the eigenspace $\mathrm{Ker}\,(A - \lambda I)$.
4. A matrix is *derogatory* if the geometric multiplicity of at least one of its eigenvalues is larger than one.
5. An eigenvalue is *semisimple* if its algebraic multiplicity is equal to its geometric multiplicity. An eigenvalue that is not semisimple is called *defective*.

Often, $\lambda_1, \lambda_2, \ldots, \lambda_p$ $(p \leq n)$ are used to denote the *distinct* eigenvalues of A. It is easy to show that the characteristic polynomials of two similar matrices are identical; see Exercise 9. Therefore, the eigenvalues of two similar matrices are equal and so are their algebraic multiplicities. Moreover, if v is an eigenvector of B, then Xv is an eigenvector of A and, conversely, if y is an eigenvector of A then $X^{-1}y$ is an eigenvector of B. As a result the number of independent eigenvectors associated with a given eigenvalue is the same for two similar matrices, i.e., their geometric multiplicity is also the same.

1.8.1 REDUCTION TO THE DIAGONAL FORM

The simplest form in which a matrix can be reduced is undoubtedly the diagonal form. Unfortunately, this reduction is not always possible. A matrix that can be reduced to the diagonal form is called *diagonalizable*. The following theorem characterizes such matrices.

THEOREM 1.1 *A matrix of dimension n is diagonalizable if and only if it has n linearly independent eigenvectors.*

Proof. A matrix A is diagonalizable if and only if there exists a nonsingular matrix X and a diagonal matrix D such that $A = XDX^{-1}$, or equivalently $AX = XD$, where D is a diagonal matrix. This is equivalent to saying that n linearly independent vectors exist — the n column-vectors of X — such that $Ax_i = d_i x_i$. Each of these column-vectors is an eigenvector of A. ∎

A matrix that is diagonalizable has only semisimple eigenvalues. Conversely, if all the eigenvalues of a matrix A are semisimple, then A has n eigenvectors. It can be easily shown that these eigenvectors are linearly independent; see Exercise 2. As a result, we have the following proposition.

PROPOSITION 1.4 *A matrix is diagonalizable if and only if all its eigenvalues are semisimple.*

Since every simple eigenvalue is semisimple, an immediate corollary of the above result is: When A has n distinct eigenvalues, then it is diagonalizable.

1.8.2 THE JORDAN CANONICAL FORM

From the theoretical viewpoint, one of the most important canonical forms of matrices is the well known Jordan form. A full development of the steps leading to the Jordan form is beyond the scope of this book. Only the main theorem is stated. Details, including the proof, can be found in standard books of linear algebra such as [117]. In the following, m_i refers to the algebraic multiplicity of the individual eigenvalue λ_i and l_i is the *index* of the eigenvalue, i.e., the smallest integer for which $\mathrm{Ker}(A - \lambda_i I)^{l_i+1} = \mathrm{Ker}(A - \lambda_i I)^{l_i}$.

THEOREM 1.2 *Any matrix A can be reduced to a block diagonal matrix consisting of p diagonal blocks, each associated with a distinct eigenvalue λ_i. Each of these diagonal blocks has itself a block diagonal structure consisting of γ_i sub-blocks, where γ_i is the geometric multiplicity of the eigenvalue λ_i. Each of the sub-blocks, referred to as a Jordan* *block, is an upper bidiagonal matrix of size not exceeding $l_i \leq m_i$, with the constant λ_i on the diagonal and the constant one on the super diagonal.*

The i-th diagonal block, $i = 1, \ldots, p$, is known as the i-th Jordan submatrix (sometimes "Jordan Box"). The Jordan submatrix number i starts in column $j_i \equiv m_1 + m_2 + \cdots + m_{i-1} + 1$. Thus,

$$X^{-1}AX = J = \begin{pmatrix} J_1 & & & & & \\ & J_2 & & & & \\ & & \ddots & & & \\ & & & J_i & & \\ & & & & \ddots & \\ & & & & & J_p \end{pmatrix},$$

where each J_i is associated with λ_i and is of size m_i the algebraic multiplicity of λ_i. It has itself the following structure,

$$J_i = \begin{pmatrix} J_{i1} & & & \\ & J_{i2} & & \\ & & \ddots & \\ & & & J_{i\gamma_i} \end{pmatrix} \text{ with } J_{ik} = \begin{pmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_i & 1 \\ & & & \lambda_i \end{pmatrix}.$$

Each of the blocks J_{ik} corresponds to a different eigenvector associated with the eigenvalue λ_i. Its size l_i is the index of λ_i.

1.8.3 THE SCHUR CANONICAL FORM

Here, it will be shown that any matrix is unitarily similar to an upper triangular matrix. The only result needed to prove the following theorem is that any vector of 2-norm one can be completed by $n - 1$ additional vectors to form an orthonormal basis of $\mathbb{C}^n$.

THEOREM 1.3 *For any square matrix A, there exists a unitary matrix Q such that*

$$Q^H AQ = R$$

is upper triangular.

Proof. The proof is by induction over the dimension n. The result is trivial for $n = 1$. Assume that it is true for $n - 1$ and consider any matrix A of size n. The matrix admits at least one eigenvector u that is associated with an eigenvalue λ. Also assume without loss of generality that $\|u\|_2 = 1$. First, complete the vector u into an orthonormal set, i.e., find an $n \times (n - 1)$ matrix V such that the $n \times n$ matrix $U = [u, V]$ is unitary. Then $AU = [\lambda u, AV]$ and hence,

$$U^H AU = \begin{bmatrix} u^H \\ V^H \end{bmatrix} [\lambda u, AV] = \begin{pmatrix} \lambda & u^H AV \\ 0 & V^H AV \end{pmatrix}. \tag{1.24}$$

Now use the induction hypothesis for the $(n - 1) \times (n - 1)$ matrix $B = V^H AV$: There exists an $(n - 1) \times (n - 1)$ unitary matrix Q_1 such that $Q_1^H BQ_1 = R_1$ is upper triangular.

Define the $n \times n$ matrix

$$\hat{Q}_1 = \begin{pmatrix} 1 & 0 \\ 0 & Q_1 \end{pmatrix}$$

and multiply both members of (1.24) by $\hat{Q}_1^H$ from the left and $\hat{Q}_1$ from the right. The resulting matrix is clearly upper triangular and this shows that the result is true for A, with $Q = \hat{Q}_1 U$ which is a unitary $n \times n$ matrix. ∎

A simpler proof that uses the Jordan canonical form and the QR decomposition is the subject of Exercise 7. Since the matrix R is triangular and similar to A, its diagonal elements are equal to the eigenvalues of A ordered in a certain manner. In fact, it is easy to extend the proof of the theorem to show that this factorization can be obtained with *any order* for the eigenvalues. Despite its simplicity, the above theorem has far-reaching consequences, some of which will be examined in the next section.

It is important to note that for any $k \le n$, the subspace spanned by the first k columns of Q is invariant under A. Indeed, the relation $AQ = QR$ implies that for $1 \le j \le k$, we have

$$Aq_j = \sum_{i=1}^{i=j} r_{ij} q_i.$$

If we let $Q_k = [q_1, q_2, \ldots, q_k]$ and if R_k is the principal leading submatrix of dimension k of R, the above relation can be rewritten as

$$AQ_k = Q_k R_k,$$

which is known as the partial Schur decomposition of A. The simplest case of this decomposition is when $k = 1$, in which case q_1 is an eigenvector. The vectors q_i are usually called Schur vectors. Schur vectors are not unique and depend, in particular, on the order chosen for the eigenvalues.

A slight variation on the Schur canonical form is the quasi-Schur form, also called the real Schur form. Here, diagonal blocks of size 2×2 are allowed in the upper triangular matrix R. The reason for this is to avoid complex arithmetic when the original matrix is real. A 2×2 block is associated with each complex conjugate pair of eigenvalues of the matrix.

Example 1.2 Consider the 3×3 matrix

$$A = \begin{pmatrix} 1 & 10 & 0 \\ -1 & 3 & 1 \\ -1 & 0 & 1 \end{pmatrix}.$$

The matrix A has the pair of complex conjugate eigenvalues

$$2.4069\ldots \pm i \times 3.2110\ldots$$

and the real eigenvalue $0.1863\ldots$. The standard (complex) Schur form is given by the pair of matrices

$$V = \begin{pmatrix} 0.3381 - 0.8462i & 0.3572 - 0.1071i & 0.1749 \\ 0.3193 - 0.0105i & -0.2263 - 0.6786i & -0.6214 \\ 0.1824 + 0.1852i & -0.2659 - 0.5277i & 0.7637 \end{pmatrix}$$

and

$$S = \begin{pmatrix} 2.4069 + 3.2110i & 4.6073 - 4.7030i & -2.3418 - 5.2330i \\ 0 & 2.4069 - 3.2110i & -2.0251 - 1.2016i \\ 0 & 0 & 0.1863 \end{pmatrix}.$$

It is possible to avoid complex arithmetic by using the quasi-Schur form which consists of the pair of matrices

$$U = \begin{pmatrix} -0.9768 & 0.1236 & 0.1749 \\ -0.0121 & 0.7834 & -0.6214 \\ 0.2138 & 0.6091 & 0.7637 \end{pmatrix}$$

and

$$R = \begin{pmatrix} 1.3129 & -7.7033 & 6.0407 \\ 1.4938 & 3.5008 & -1.3870 \\ 0 & 0 & 0.1863 \end{pmatrix}.$$

We conclude this section by pointing out that the Schur and the quasi-Schur forms of a given matrix are in no way unique. In addition to the dependence on the ordering of the eigenvalues, any column of Q can be multiplied by a complex sign $e^{i\theta}$ and a new corresponding R can be found. For the quasi-Schur form, there are infinitely many ways to select the 2×2 blocks, corresponding to applying arbitrary rotations to the columns of Q associated with these blocks.

1.8.4 APPLICATION TO POWERS OF MATRICES

The analysis of many numerical techniques is based on understanding the behavior of the successive powers A^k of a given matrix A. In this regard, the following theorem plays a fundamental role in numerical linear algebra, more particularly in the analysis of iterative methods.

THEOREM 1.4 *The sequence A^k, $k = 0, 1, \ldots$, converges to zero if and only if $\rho(A) < 1$.*

Proof. To prove the necessary condition, assume that $A^k \to 0$ and consider u_1 a unit eigenvector associated with an eigenvalue λ_1 of maximum modulus. We have

$$A^k u_1 = \lambda_1^k u_1,$$

which implies, by taking the 2-norms of both sides,

$$|\lambda_1^k| = \|A^k u_1\|_2 \to 0.$$

This shows that $\rho(A) = |\lambda_1| < 1$.

The Jordan canonical form must be used to show the sufficient condition. Assume that $\rho(A) < 1$. Start with the equality

$$A^k = XJ^kX^{-1}.$$

To prove that A^k converges to zero, it is sufficient to show that J^k converges to zero. An important observation is that J^k preserves its block form. Therefore, it is sufficient to prove that each of the Jordan blocks converges to zero. Each block is of the form

$$J_i = \lambda_i I + E_i$$

where E_i is a nilpotent matrix of index l_i, i.e., $E_i^{l_1} = 0$. Therefore, for $k \geq l_i$,

$$J_i^k = \sum_{j=0}^{l_1-1} \frac{k!}{j!(k-j)!} \lambda_i^{k-j} E_i^j.$$

Using the triangle inequality for any norm and taking $k \geq l_i$ yields

$$\|J_i^k\| \leq \sum_{j=0}^{l_1-1} \frac{k!}{j!(k-j)!} |\lambda_i|^{k-j} \|E_i^j\|.$$

Since $|\lambda_i| < 1$, each of the terms in this *finite* sum converges to zero as $k \to \infty$. Therefore, the matrix J_i^k converges to zero. ■

An equally important result is stated in the following theorem.

THEOREM 1.5 *The series*

$$\sum_{k=0}^{\infty} A^k$$

converges if and only if $\rho(A) < 1$. Under this condition, $I - A$ is nonsingular and the limit of the series is equal to $(I - A)^{-1}$.

Proof. The first part of the theorem is an immediate consequence of Theorem 1.4. Indeed, if the series converges, then $\|A^k\| \to 0$. By the previous theorem, this implies that $\rho(A) < 1$. To show that the converse is also true, use the equality

$$I - A^{k+1} = (I - A)(I + A + A^2 + \ldots + A^k)$$

and exploit the fact that since $\rho(A) < 1$, then $I - A$ is nonsingular, and therefore,

$$(I - A)^{-1}(I - A^{k+1}) = I + A + A^2 + \ldots + A^k.$$

This shows that the series converges since the left-hand side will converge to $(I - A)^{-1}$. In addition, it also shows the second part of the theorem. ■

Another important consequence of the Jordan canonical form is a result that relates the spectral radius of a matrix to its matrix norm.

THEOREM 1.6 *For any matrix norm* $\|.\|$, *we have*

$$\lim_{k\to\infty} \|A^k\|^{1/k} = \rho(A).$$

Proof. The proof is a direct application of the Jordan canonical form and is the subject of Exercise 10. ∎

1.9 NORMAL AND HERMITIAN MATRICES

This section examines specific properties of normal matrices and Hermitian matrices, including some optimality properties related to their spectra. The most common normal matrices that arise in practice are Hermitian or skew-Hermitian.

1.9.1 NORMAL MATRICES

By definition, a matrix is said to be normal if it commutes with its transpose conjugate, i.e., if it satisfies the relation

$$A^H A = A A^H. \tag{1.25}$$

An immediate property of normal matrices is stated in the following lemma.

LEMMA 1.1 *If a normal matrix is triangular, then it is a diagonal matrix.*

Proof. Assume, for example, that A is upper triangular and normal. Compare the first diagonal element of the left-hand side matrix of (1.25) with the corresponding element of the matrix on the right-hand side. We obtain that

$$|a_{11}|^2 = \sum_{j=1}^{n} |a_{1j}|^2,$$

which shows that the elements of the first row are zeros except for the diagonal one. The same argument can now be used for the second row, the third row, and so on to the last row, to show that $a_{ij} = 0$ for $i \neq j$. ∎

A consequence of this lemma is the following important result.

THEOREM 1.7 *A matrix is normal if and only if it is unitarily similar to a diagonal matrix.*

Proof. It is straightforward to verify that a matrix which is unitarily similar to a diagonal matrix is normal. We now prove that any normal matrix A is unitarily similar to a diagonal matrix. Let $A = QRQ^H$ be the Schur canonical form of A where Q is unitary and R is upper triangular. By the normality of A,

$$QR^HQ^HQRQ^H = QRQ^HQR^HQ^H$$

or,

$$QR^HRQ^H = QRR^HQ^H.$$

Upon multiplication by Q^H on the left and Q on the right, this leads to the equality $R^HR = RR^H$ which means that R is normal, and according to the previous lemma this is only possible if R is diagonal. ■

Thus, any normal matrix is diagonalizable and admits an orthonormal basis of eigenvectors, namely, the column vectors of Q.

The following result will be used in a later chapter. The question that is asked is: Assuming that any eigenvector of a matrix A is also an eigenvector of A^H, is A normal? If A had a full set of eigenvectors, then the result is true and easy to prove. Indeed, if V is the $n \times n$ matrix of common eigenvectors, then $AV = VD_1$ and $A^HV = VD_2$, with D_1 and D_2 diagonal. Then, $AA^HV = VD_1D_2$ and $A^HAV = VD_2D_1$ and, therefore, $AA^H = A^HA$. It turns out that the result is true in general, i.e., independently of the number of eigenvectors that A admits.

LEMMA 1.2 *A matrix A is normal if and only if each of its eigenvectors is also an eigenvector of A^H.*

Proof. If A is normal, then its left and right eigenvectors are identical, so the sufficient condition is trivial. Assume now that a matrix A is such that each of its eigenvectors $v_i, i = 1, \ldots, k$, with $k \leq n$ is an eigenvector of A^H. For each eigenvector v_i of A, $Av_i = \lambda_i v_i$, and since v_i is also an eigenvector of A^H, then $A^Hv_i = \mu v_i$. Observe that $(A^Hv_i, v_i) = \mu(v_i, v_i)$ and because $(A^Hv_i, v_i) = (v_i, Av_i) = \bar{\lambda}_i(v_i, v_i)$, it follows that $\mu = \bar{\lambda}_i$. Next, it is proved by contradiction that there are no elementary divisors. Assume that the contrary is true for λ_i. Then, the first principal vector u_i associated with λ_i is defined by

$$(A - \lambda_i I)u_i = v_i.$$

Taking the inner product of the above relation with v_i, we obtain

$$(Au_i, v_i) = \lambda_i(u_i, v_i) + (v_i, v_i). \tag{1.26}$$

On the other hand, it is also true that

$$(Au_i, v_i) = (u_i, A^Hv_i) = (u_i, \bar{\lambda}_i v_i) = \lambda_i(u_i, v_i). \tag{1.27}$$

A result of (1.26) and (1.27) is that $(v_i, v_i) = 0$ which is a contradiction. Therefore, A has a full set of eigenvectors. This leads to the situation discussed just before the lemma, from which it is concluded that A must be normal. ■

Clearly, Hermitian matrices are a particular case of normal matrices. Since a normal matrix satisfies the relation $A = QDQ^H$, with D diagonal and Q unitary, the eigenvalues of A are the diagonal entries of D. Therefore, if these entries are real it is clear that $A^H = A$. This is restated in the following corollary.

COROLLARY 1.1 *A normal matrix whose eigenvalues are real is Hermitian.*

As will be seen shortly, the converse is also true, i.e., a Hermitian matrix has real eigenvalues.

An eigenvalue λ of any matrix satisfies the relation

$$\lambda = \frac{(Au, u)}{(u, u)},$$

where u is an associated eigenvector. Generally, one might consider the complex scalars

$$\mu(x) = \frac{(Ax, x)}{(x, x)}, \tag{1.28}$$

defined for any nonzero vector in $\mathbb{C}^n$. These ratios are known as *Rayleigh quotients* and are important both for theoretical and practical purposes. The set of all possible Rayleigh quotients as x runs over $\mathbb{C}^n$ is called the *field of values* of A. This set is clearly bounded since each $|\mu(x)|$ is bounded by the the 2-norm of A, i.e., $|\mu(x)| \leq \|A\|_2$ for all x.

If a matrix is normal, then any vector x in $\mathbb{C}^n$ can be expressed as

$$\sum_{i=1}^{n} \xi_i q_i,$$

where the vectors q_i form an orthogonal basis of eigenvectors, and the expression for $\mu(x)$ becomes

$$\mu(x) \;=\; \frac{(Ax, x)}{(x, x)} \;=\; \frac{\sum_{k=1}^{n} \lambda_k |\xi_k|^2}{\sum_{k=1}^{n} |\xi_k|^2} \;\equiv\; \sum_{k=1}^{n} \beta_k \lambda_k, \tag{1.29}$$

where

$$0 \leq \beta_i = \frac{|\xi_i|^2}{\sum_{k=1}^{n} |\xi_k|^2} \leq 1, \quad \text{and} \quad \sum_{i=1}^{n} \beta_i = 1.$$

From a well known characterization of convex hulls established by Hausdorff (Hausdorff's convex hull theorem), this means that the set of all possible Rayleigh quotients as x runs over all of $\mathbb{C}^n$ is equal to the convex hull of the λ_i's. This leads to the following theorem which is stated without proof.

THEOREM 1.8 *The field of values of a normal matrix is equal to the convex hull of its spectrum.*

The next question is whether or not this is also true for nonnormal matrices and the answer is no: The convex hull of the eigenvalues and the field of values of a nonnormal matrix are different in general. As a generic example, one can take any nonsymmetric real matrix which has real eigenvalues only. In this case, the convex hull of the spectrum is a real interval but its field of values will contain imaginary values. See Exercise 12 for another example. It has been shown (Hausdorff) that the field of values of a matrix is a convex set. Since the eigenvalues are members of the field of values, their convex hull is contained in the field of values. This is summarized in the following proposition.

PROPOSITION 1.5 *The field of values of an arbitrary matrix is a convex set which contains the convex hull of its spectrum. It is equal to the convex hull of the spectrum when the matrix is normal.*

1.9.2 HERMITIAN MATRICES

A first result on Hermitian matrices is the following.

THEOREM 1.9 *The eigenvalues of a Hermitian matrix are real, i.e.,* $\sigma(A) \subset \mathbb{R}$.

Proof. Let λ be an eigenvalue of A and u an associated eigenvector or 2-norm unity. Then

$$\lambda = (Au, u) = (u, Au) = \overline{(Au, u)} = \overline{\lambda},$$

which is the stated result. ■

It is not difficult to see that if, in addition, the matrix is real, then the eigenvectors can be chosen to be real; see Exercise 21. Since a Hermitian matrix is normal, the following is a consequence of Theorem 1.7.

THEOREM 1.10 *Any Hermitian matrix is unitarily similar to a real diagonal matrix.*

In particular a Hermitian matrix admits a set of orthonormal eigenvectors that form a basis of $\mathbb{C}^n$.

In the proof of Theorem 1.8 we used the fact that the inner products (Au, u) are real. Generally, it is clear that any Hermitian matrix is such that (Ax, x) is real for any vector $x \in \mathbb{C}^n$. It turns out that the converse is also true, i.e., it can be shown that if (Az, z) is real for all vectors z in $\mathbb{C}^n$, then the matrix A is Hermitian; see Exercise 15.

Eigenvalues of Hermitian matrices can be characterized by optimality properties of the Rayleigh quotients (1.28). The best known of these is the min-max principle. We now label all the eigenvalues of A in descending order:

$$\lambda_1 \ \geq \ \lambda_2 \ \geq \ \cdots \ \geq \ \lambda_n.$$

Here, the eigenvalues are not necessarily distinct and they are repeated, each according to its multiplicity. In the following theorem, known as the *Min-Max Theorem*, S represents a generic subspace of $\mathbb{C}^n$.

THEOREM 1.11 *The eigenvalues of a Hermitian matrix A are characterized by the relation*

$$\lambda_k = \min_{S,\ \dim\ (S)=n-k+1} \quad \max_{x \in S, x \neq 0} \ \frac{(Ax, x)}{(x, x)}. \tag{1.30}$$

Proof. Let $\{q_i\}_{i=1,\ldots,n}$ be an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of A associated with $\lambda_1, \ldots, \lambda_n$ respectively. Let S_k be the subspace spanned by the first k of these vectors and denote by $\mu(S)$ the maximum of $(Ax, x)/(x, x)$ over all nonzero vectors of a subspace S. Since the dimension of S_k is k, a well known theorem of linear algebra shows that its intersection with any subspace S of dimension $n - k + 1$ is not reduced to $\{0\}$, i.e., there is vector x in $S \bigcap S_k$. For this $x = \sum_{i=1}^{k} \xi_i q_i$, we have

$$\frac{(Ax, x)}{(x, x)} \;=\; \frac{\sum_{i=1}^{k} \lambda_i |\xi_i|^2}{\sum_{i=1}^{k} |\xi_i|^2} \;\geq\; \lambda_k$$

so that $\mu(S) \;\geq\; \lambda_k$.

Consider, on the other hand, the particular subspace S_0 of dimension $n - k + 1$ which is spanned by $q_k, \ldots, q_n$. For each vector x in this subspace, we have

$$\frac{(Ax, x)}{(x, x)} = \frac{\sum_{i=k}^{n} \lambda_i |\xi_i|^2}{\sum_{i=k}^{n} |\xi_i|^2} \;\leq\; \lambda_k$$

so that $\mu(S_0) \;\leq\; \lambda_k$. In other words, as S runs over all the $(n - k + 1)$-dimensional subspaces, $\mu(S)$ is always $\geq \lambda_k$ and there is at least one subspace S_0 for which $\mu(S_0) \leq \lambda_k$. This shows the desired result. ■

The above result is often called the Courant-Fisher min-max principle or theorem. As a particular case, the largest eigenvalue of A satisfies

$$\lambda_1 = \max_{x \neq 0} \frac{(Ax, x)}{(x, x)}. \tag{1.31}$$

Actually, there are four different ways of rewriting the above characterization. The second formulation is

$$\lambda_k = \max_{S,\ \dim\ (S)=k}\ \min_{x \in S, x \neq 0} \frac{(Ax, x)}{(x, x)} \tag{1.32}$$

and the two other ones can be obtained from (1.30) and (1.32) by simply relabeling the eigenvalues increasingly instead of decreasingly. Thus, with our labeling of the eigenvalues in descending order, (1.32) tells us that the smallest eigenvalue satisfies

$$\lambda_n = \min_{x \neq 0} \frac{(Ax, x)}{(x, x)}, \tag{1.33}$$

with λ_n replaced by λ_1 if the eigenvalues are relabeled increasingly.

In order for all the eigenvalues of a Hermitian matrix to be positive, it is necessary and sufficient that

$$(Ax, x) > 0, \quad \forall\, x \in \mathbb{C}^n, \quad x \neq 0.$$

Such a matrix is called *positive definite*. A matrix which satisfies $(Ax, x) \geq 0$ for any x is said to be *positive semidefinite*. In particular, the matrix $A^H A$ is semipositive definite for any rectangular matrix, since

$$(A^H Ax, x) = (Ax, Ax) \;\geq\; 0, \quad \forall\, x.$$

Similarly, AA^H is also a Hermitian semipositive definite matrix. The square roots of the eigenvalues of $A^H A$ for a general rectangular matrix A are called the *singular values* of

A and are denoted by σ_i. In Section 1.5, we have stated without proof that the 2-norm of any matrix A is equal to the largest singular value σ_1 of A. This is now an obvious fact, because

$$\|A\|_2^2 = \max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{x \neq 0} \frac{(Ax, Ax)}{(x, x)} = \max_{x \neq 0} \frac{(A^H Ax, x)}{(x, x)} = \sigma_1^2$$

which results from (1.31).

Another characterization of eigenvalues, known as the Courant characterization, is stated in the next theorem. In contrast with the min-max theorem, this property is recursive in nature.

THEOREM 1.12 *The eigenvalue λ_i and the corresponding eigenvector q_i of a Hermitian matrix are such that*

$$\lambda_1 = \frac{(Aq_1, q_1)}{(q_1, q_1)} = \max_{x \in \mathbb{C}^n, x \neq 0} \frac{(Ax, x)}{(x, x)}$$

and for $k > 1$,

$$\lambda_k = \frac{(Aq_k, q_k)}{(q_k, q_k)} = \max_{x \neq 0, q_1^H x = \ldots = q_{k-1}^H x = 0} \frac{(Ax, x)}{(x, x)}. \tag{1.34}$$

In other words, the maximum of the Rayleigh quotient over a subspace that is orthogonal to the first $k-1$ eigenvectors is equal to λ_k and is achieved for the eigenvector q_k associated with λ_k. The proof follows easily from the expansion (1.29) of the Rayleigh quotient.

1.10 NONNEGATIVE MATRICES, M-MATRICES

Nonnegative matrices play a crucial role in the theory of matrices. They are important in the study of convergence of iterative methods and arise in many applications including economics, queuing theory, and chemical engineering.

A *nonnegative matrix* is simply a matrix whose entries are nonnegative. More generally, a partial order relation can be defined on the set of matrices.

DEFINITION 1.3 *Let A and B be two $n \times m$ matrices. Then*

$$A \leq B$$

if by definition, $a_{ij} \leq b_{ij}$ for $1 \leq i \leq n$, $1 \leq j \leq m$. If O denotes the $n \times m$ zero matrix, then A is nonnegative if $A \geq O$, and positive if $A > O$. Similar definitions hold in which "positive" is replaced by "negative".

The binary relation "$\leq$" imposes only a *partial* order on $\mathbb{R}^{n \times m}$ since two arbitrary matrices in $\mathbb{R}^{n \times m}$ are not necessarily comparable by this relation. For the remainder of this section, we now assume that only square matrices are involved. The next proposition lists a number of rather trivial properties regarding the partial order relation just defined.

PROPOSITION 1.6 *The following properties hold.*

1. *The relation $\leq$ for matrices is reflexive ($A \leq A$), antisymmetric (if $A \leq B$ and $B \leq A$, then $A = B$), and transitive (if $A \leq B$ and $B \leq C$, then $A \leq C$).*
2. *If A and B are nonnegative, then so is their product AB and their sum $A + B$.*
3. *If A is nonnegative, then so is A^k.*
4. *If $A \leq B$, then $A^T \leq B^T$.*
5. *If $O \leq A \leq B$, then $\|A\|_1 \leq \|B\|_1$ and similarly $\|A\|_\infty \leq \|B\|_\infty$.*

The proof of these properties is left as Exercise 23.

A matrix is said to be *reducible* if there is a permutation matrix P such that PAP^T is block upper triangular. Otherwise, it is *irreducible*. An important result concerning nonnegative matrices is the following theorem known as the Perron-Frobenius theorem.

THEOREM 1.13 *Let A be a real $n \times n$ nonnegative irreducible matrix. Then $\lambda \equiv \rho(A)$, the spectral radius of A, is a simple eigenvalue of A. Moreover, there exists an eigenvector u with positive elements associated with this eigenvalue.*

A relaxed version of this theorem allows the matrix to be reducible but the conclusion is somewhat weakened in the sense that the elements of the eigenvectors are only guaranteed to be *nonnegative*.

Next, a useful property is established.

PROPOSITION 1.7 *Let A, B, C be nonnegative matrices, with $A \leq B$. Then*

$$AC \leq BC \quad \text{and} \quad CA \leq CB.$$

Proof. Consider the first inequality only, since the proof for the second is identical. The result that is claimed translates into

$$\sum_{k=1}^{n} a_{ik} c_{kj} \leq \sum_{k=1}^{n} b_{ik} c_{kj}, \quad 1 \leq i, j \leq n,$$

which is clearly true by the assumptions. ∎

A consequence of the proposition is the following corollary.

COROLLARY 1.2 *Let A and B be two nonnegative matrices, with $A \leq B$. Then*

$$A^k \leq B^k, \quad \forall\, k \geq 0. \tag{1.35}$$

Proof. The proof is by induction. The inequality is clearly true for $k = 0$. Assume that (1.35) is true for k. According to the previous proposition, multiplying (1.35) from the left by A results in

$$A^{k+1} \leq AB^k. \tag{1.36}$$

Now, it is clear that if $B \geq 0$, then also $B^k \geq 0$, by Proposition 1.6. We now multiply both sides of the inequality $A \leq B$ by B^k to the right, and obtain

$$AB^k \leq B^{k+1}. \tag{1.37}$$

The inequalities (1.36) and (1.37) show that $A^{k+1} \leq B^{k+1}$, which completes the induction proof. ∎

A theorem which has important consequences on the analysis of iterative methods will now be stated.

THEOREM 1.14 *Let A and B be two square matrices that satisfy the inequalities*

$$O \leq A \leq B. \tag{1.38}$$

Then

$$\rho(A) \leq \rho(B). \tag{1.39}$$

Proof. The proof is based on the following equality stated in Theorem 1.6

$$\rho(X) = \lim_{k \to \infty} \|X^k\|^{1/k}$$

for any matrix norm. Choosing the $1-$norm, for example, we have from the last property in Proposition 1.6

$$\rho(A) = \lim_{k \to \infty} \|A^k\|_1^{1/k} \leq \lim_{k \to \infty} \|B^k\|_1^{1/k} = \rho(B)$$

which completes the proof. ∎

THEOREM 1.15 *Let B be a nonnegative matrix. Then $\rho(B) < 1$ if and only if $I - B$ is nonsingular and $(I - B)^{-1}$ is nonnegative.*

Proof. Define $C = I - B$. If it is assumed that $\rho(B) < 1$, then by Theorem 1.5, $C = I - B$ is nonsingular and

$$C^{-1} = (I - B)^{-1} = \sum_{i=0}^{\infty} B^i. \tag{1.40}$$

In addition, since $B \geq 0$, all the powers of B as well as their sum in (1.40) are also nonnegative.

To prove the sufficient condition, assume that C is nonsingular and that its inverse is nonnegative. By the Perron-Frobenius theorem, there is a nonnegative eigenvector u associated with $\rho(B)$, which is an eigenvalue, i.e.,

$$Bu = \rho(B)u$$

or, equivalently,

$$C^{-1}u = \frac{1}{1 - \rho(B)}u.$$

Since u and C^{-1} are nonnegative, and $I - B$ is nonsingular, this shows that $1 - \rho(B) > 0$, which is the desired result. ∎

DEFINITION 1.4 *A matrix is said to be an M-matrix if it satisfies the following four properties:*

1. $a_{i,i} > 0$ *for* $i = 1, \ldots, n$.
2. $a_{i,j} \leq 0$ *for* $i \neq j,\ i, j = 1, \ldots, n$.
3. A *is nonsingular.*
4. $A^{-1} \geq 0$.

In reality, the four conditions in the above definition are somewhat redundant and equivalent conditions that are more rigorous will be given later. Let A be any matrix which satisfies properties (1) and (2) in the above definition and let D be the diagonal of A. Since $D > 0$,

$$A = D - (D - A) = D\left(I - (I - D^{-1}A)\right).$$

Now define

$$B \equiv I - D^{-1}A.$$

Using the previous theorem, $I - B = D^{-1}A$ is nonsingular and $(I - B)^{-1} = A^{-1}D \geq 0$ if and only if $\rho(B) < 1$. It is now easy to see that conditions (3) and (4) of Definition 1.4 can be replaced by the condition $\rho(B) < 1$.

THEOREM 1.16 *Let a matrix A be given such that*

1. $a_{i,i} > 0$ *for* $i = 1, \ldots, n$.
2. $a_{i,j} \leq 0$ *for* $i \neq j,\ i, j = 1, \ldots, n$.

Then A is an M-matrix if and only if

3. $\rho(B) < 1$, *where* $B = I - D^{-1}A$.

Proof. From the above argument, an immediate application of Theorem 1.15 shows that properties (3) and (4) of the above definition are equivalent to $\rho(B) < 1$, where $B = I - C$ and $C = D^{-1}A$. In addition, C is nonsingular iff A is and C^{-1} is nonnegative iff A is.

■

The next theorem shows that the condition (1) in Definition 1.4 is implied by the other three.

THEOREM 1.17 *Let a matrix A be given such that*

1. $a_{i,j} \leq 0$ *for* $i \neq j,\ i, j = 1, \ldots, n$.
2. A *is nonsingular.*
3. $A^{-1} \geq 0$.

Then

4. $a_{i,i} > 0$ *for* $i = 1, \ldots, n$, *i.e.,* A *is an M-matrix.*
5. $\rho(B) < 1$ *where* $B = I - D^{-1}A$.

Proof. Define $C \equiv A^{-1}$. Writing that $(AC)_{ii} = 1$ yields

$$\sum_{k=1}^{n} a_{ik} c_{ki} = 1$$

which gives

$$a_{ii} c_{ii} = 1 - \sum_{\substack{k=1 \\ k \neq i}}^{n} a_{ik} c_{ki}.$$

Since $a_{ik} c_{ki} \le 0$ for all k, the right-hand side is ≥ 1 and since $c_{ii} \ge 0$, then $a_{ii} > 0$. The second part of the result now follows immediately from an application of the previous theorem. ∎

Finally, this useful result follows.

THEOREM 1.18 *Let A, B be two matrices which satisfy*

1. $A \le B$.
2. $b_{ij} \le 0$ *for all* $i \neq j$.

Then if A is an M-matrix, so is the matrix B.

Proof. Assume that A is an M-matrix and let D_X denote the diagonal of a matrix X. The matrix D_B is positive because

$$D_B \ge D_A > 0.$$

Consider now the matrix $I - D_B^{-1} B$. Since $A \le B$, then

$$D_A - A \ge D_B - B \ge O$$

which, upon multiplying through by D_A^{-1}, yields

$$I - D_A^{-1} A \ge D_A^{-1}(D_B - B) \ge D_B^{-1}(D_B - B) = I - D_B^{-1} B \ge O.$$

Since the matrices $I - D_B^{-1} B$ and $I - D_A^{-1} A$ are nonnegative, Theorems 1.14 and 1.16 imply that

$$\rho(I - D_B^{-1} B) \le \rho(I - D_A^{-1} A) < 1.$$

This establishes the result by using Theorem 1.16 once again. ∎

1.11 POSITIVE-DEFINITE MATRICES

A *real* matrix is said to be *positive definite* or *positive real* if

$$(Au, u) > 0, \quad \forall\, u \in \mathbb{R}^n,\ u \neq 0. \tag{1.41}$$

It must be emphasized that this definition is only useful when formulated entirely for real variables. Indeed, if u were not restricted to be real, then assuming that (Au, u) is real for all u complex would imply that A is Hermitian; see Exercise 15. If, in addition to Definition 1.41, A is symmetric (real), then A is said to be *Symmetric Positive Definite* (SPD). Similarly, if A is Hermitian, then A is said to be *Hermitian Positive Definite* (HPD). Some properties of HPD matrices were seen in Section 1.9, in particular with regards to their eigenvalues. Now the more general case where A is non-Hermitian and positive definite is considered.

We begin with the observation that any square matrix (real or complex) can be decomposed as

$$A = H + iS, \tag{1.42}$$

in which

$$H = \frac{1}{2}(A + A^H) \tag{1.43}$$

$$S = \frac{1}{2i}(A - A^H). \tag{1.44}$$

Note that both H and S are Hermitian while the matrix iS in the decomposition (1.42) is skew-Hermitian. The matrix H in the decomposition is called the *Hermitian part* of A, while the matrix iS is the *skew-Hermitian part* of A. The above decomposition is the analogue of the decomposition of a complex number z into $z = x + iy$,

$$x = \Re e(z) = \frac{1}{2}(z + \bar{z}), \; y = \Im m(z) = \frac{1}{2i}(z - \bar{z}).$$

When A is real and u is a real vector then (Au, u) is real and, as a result, the decomposition (1.42) immediately gives the equality

$$(Au, u) = (Hu, u). \tag{1.45}$$

This results in the following theorem.

THEOREM 1.19 *Let A be a real positive definite matrix. Then A is nonsingular. In addition, there exists a scalar $\alpha > 0$ such that*

$$(Au, u) \geq \alpha \|u\|_2^2, \tag{1.46}$$

for any real vector u.

Proof. The first statement is an immediate consequence of the definition of positive definiteness. Indeed, if A were singular, then there would be a nonzero vector such that $Au = 0$ and as a result $(Au, u) = 0$ for this vector, which would contradict (1.41). We now prove the second part of the theorem. From (1.45) and the fact that A is positive definite, we conclude that H is HPD. Hence, from (1.33) based on the min-max theorem, we get

$$\min_{u \neq 0} \frac{(Au, u)}{(u, u)} = \min_{u \neq 0} \frac{(Hu, u)}{(u, u)} \geq \lambda_{min}(H) > 0.$$

Taking $\alpha \equiv \lambda_{min}(H)$ yields the desired inequality (1.46). ∎

A simple yet important result which locates the eigenvalues of A in terms of the spectra of H and S can now be proved.

THEOREM 1.20 *Let A be any square (possibly complex) matrix and let $H = \frac{1}{2}(A + A^H)$ and $S = \frac{1}{2i}(A - A^H)$. Then any eigenvalue λ_j of A is such that*

$$\lambda_{min}(H) \leq \Re e(\lambda_j) \leq \lambda_{max}(H) \tag{1.47}$$

$$\lambda_{min}(S) \leq \Im m(\lambda_j) \leq \lambda_{max}(S). \tag{1.48}$$

Proof. When the decomposition (1.42) is applied to the Rayleigh quotient of the eigenvector u_j associated with λ_j, we obtain

$$\lambda_j = (Au_j, u_j) = (Hu_j, u_j) + i(Su_j, u_j), \tag{1.49}$$

assuming that $\|u_j\|_2 = 1$. This leads to

$$\Re e(\lambda_j) = (Hu_j, u_j)$$
$$\Im m(\lambda_j) = (Su_j, u_j).$$

The result follows using properties established in Section 1.9. ∎

Thus, the eigenvalues of a matrix are contained in a rectangle defined by the eigenvalues of its Hermitian part and its non-Hermitian part. In the particular case where A is real, then iS is skew-Hermitian and its eigenvalues form a set that is symmetric with respect to the real axis in the complex plane. Indeed, in this case, iS is real and its eigenvalues come in conjugate pairs.

Note that all the arguments herein are based on the field of values and, therefore, they provide ways to localize the eigenvalues of A from knowledge of the field of values. However, this approximation can be inaccurate in some cases.

Example 1.3 Consider the matrix

$$A = \begin{pmatrix} 1 & 1 \\ 10^4 & 1 \end{pmatrix}.$$

The eigenvalues of A are -99 and 101. Those of H are $1 \pm (10^4 + 1)/2$ and those of iS are $\pm i(10^4 - 1)/2$.

When a matrix B is Symmetric Positive Definite, the mapping

$$x, y \quad \rightarrow \quad (x, y)_B \equiv (Bx, y) \tag{1.50}$$

from $\mathbb{C}^n \times \mathbb{C}^n$ to $\mathbb{C}$ is a proper inner product on $\mathbb{C}^n$, in the sense defined in Section 1.4. The associated norm is often referred to as the *energy norm*. Sometimes, it is possible to find an appropriate HPD matrix B which makes a given matrix A Hermitian, i.e., such that

$$(Ax, y)_B = (x, Ay)_B, \quad \forall\, x, y$$

although A is a non-Hermitian matrix with respect to the Euclidean inner product. The simplest examples are $A = B^{-1}C$ and $A = CB$, where C is Hermitian and B is Hermitian Positive Definite.

1.12 PROJECTION OPERATORS

Projection operators or *projectors* play an important role in numerical linear algebra, particularly in iterative methods for solving various matrix problems. This section introduces these operators from a purely algebraic point of view and gives a few of their important properties.

1.12.1 RANGE AND NULL SPACE OF A PROJECTOR

A projector P is any linear mapping from $\mathbb{C}^n$ to itself which is idempotent, i.e., such that

$$P^2 = P.$$

A few simple properties follow from this definition. First, if P is a projector, then so is $(I - P)$, and the following relation holds,

$$\mathrm{Ker}(P) = \mathrm{Ran}(I - P).$$

In addition, the two subspaces $\mathrm{Ker}(P)$ and $\mathrm{Ran}(P)$ intersect only at the element zero. Indeed, if a vector x belongs to $\mathrm{Ran}(P)$, then $Px = x$, by the idempotence property. If it is also in $\mathrm{Ker}(P)$, then $Px = 0$. Hence, $x = Px = 0$ which proves the result. Moreover, every element of $\mathbb{C}^n$ can be written as $x = Px + (I - P)x$. Therefore, the space $\mathbb{C}^n$ can be decomposed as the direct sum

$$\mathbb{C}^n = \mathrm{Ker}(P) \oplus \mathrm{Ran}(P).$$

Conversely, every pair of subspaces M and S which forms a direct sum of $\mathbb{C}^n$ defines a unique projector such that $\mathrm{Ran}(P) = M$ and $\mathrm{Ker}(P) = S$. This associated projector P maps an element x of $\mathbb{C}^n$ into the component x_1, where x_1 is the M-component in the unique decomposition $x = x_1 + x_2$ associated with the direct sum.

In fact, this association is unique, that is, an arbitrary projector P can be entirely determined by the given of two subspaces: (1) The range M of P, and (2) its null space S which is also the range of $I - P$. For any x, the vector Px satisfies the conditions,

$$\begin{aligned} Px &\in M \\ x - Px &\in S. \end{aligned}$$

The linear mapping P is said to project x *onto* M and *along* or *parallel to* the subspace S. If P is of rank m, then the range of $I - P$ is of dimension $n - m$. Therefore, it is natural to define S through its orthogonal complement $L = S^\perp$ which has dimension m. The above conditions that define $u = Px$ for any x become

$$u \in M \tag{1.51}$$

$$x - u \perp L. \tag{1.52}$$

These equations define a projector P *onto* M and *orthogonal* to the subspace L. The first statement, (1.51), establishes the m degrees of freedom, while the second, (1.52), gives the m constraints that define Px from these degrees of freedom. The general definition of projectors is illustrated in Figure 1.1.

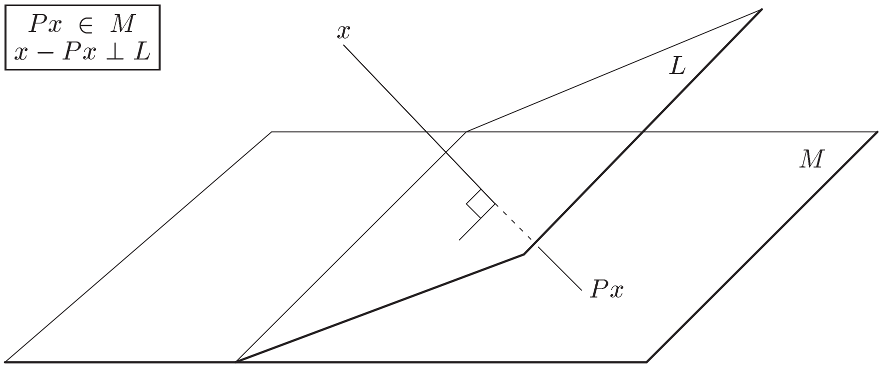


Figure 1.1 *Projection of x onto M and orthogonal to L.*

The question now is: Given two arbitrary subspaces, M and L both of dimension m, is it always possible to define a projector onto M orthogonal to L through the conditions (1.51) and (1.52)? The following lemma answers this question.

LEMMA 1.3 *Given two subspaces M and L of the same dimension m, the following two conditions are mathematically equivalent.*

i. *No nonzero vector of M is orthogonal to L;*

ii. *For any x in $\mathbb{C}^n$ there is a unique vector u which satisfies the conditions (1.51) and (1.52).*

Proof. The first condition states that any vector which is in M and also orthogonal to L must be the zero vector. It is equivalent to the condition

$$M \cap L^\perp = \{0\}.$$

Since L is of dimension m, $L^\perp$ is of dimension $n-m$ and the above condition is equivalent to the condition that

$$\mathbb{C}^n = M \oplus L^\perp. \tag{1.53}$$

This in turn is equivalent to the statement that for any x, there exists a unique pair of vectors u, w such that

$$x = u + w,$$

where u belongs to M, and $w = x - u$ belongs to $L^\perp$, a statement which is identical with *ii.* ■

In summary, given two subspaces M and L, satisfying the condition $M \cap L^\perp = \{0\}$, there is a projector P onto M orthogonal to L, which defines the projected vector u of any vector

x from equations (1.51) and (1.52). This projector is such that

$$\mathrm{Ran}(P) = M, \qquad \mathrm{Ker}(P) = L^{\perp}.$$

In particular, the condition $Px = 0$ translates into $x \in \mathrm{Ker}(P)$ which means that $x \in L^{\perp}$. The converse is also true. Hence, the following useful property,

$$Px = 0 \qquad \text{iff} \qquad x \perp L. \tag{1.54}$$

1.12.2 MATRIX REPRESENTATIONS

Two bases are required to obtain a matrix representation of a general projector: a basis $V = [v_1, \ldots, v_m]$ for the subspace $M = \mathrm{Ran}(P)$ and a second one $W = [w_1, \ldots, w_m]$ for the subspace L. These two bases are *biorthogonal* when

$$(v_i, w_j) = \delta_{ij}. \tag{1.55}$$

In matrix form this means $W^H V = I$. Since Px belongs to M, let Vy be its representation in the V basis. The constraint $x - Px \perp L$ is equivalent to the condition,

$$((x - Vy), w_j) = 0 \quad \text{for } j = 1, \ldots, m.$$

In matrix form, this can be rewritten as

$$W^H (x - Vy) = 0. \tag{1.56}$$

If the two bases are biorthogonal, then it follows that $y = W^H x$. Therefore, in this case, $Px = VW^H x$, which yields the matrix representation of P,

$$P = VW^H. \tag{1.57}$$

In case the bases V and W are not biorthogonal, then it is easily seen from the condition (1.56) that

$$P = V(W^H V)^{-1} W^H. \tag{1.58}$$

If we assume that no vector of M is orthogonal to L, then it can be shown that the $m \times m$ matrix $W^H V$ is nonsingular.

1.12.3 ORTHOGONAL AND OBLIQUE PROJECTORS

An important class of projectors is obtained in the case when the subspace L is equal to M, i.e., when

$$\mathrm{Ker}(P) = \mathrm{Ran}(P)^{\perp}.$$

Then, the projector P is said to be the *orthogonal projector* onto M. A projector that is not orthogonal is *oblique*. Thus, an orthogonal projector is defined through the following requirements satisfied for any vector x,

$$Px \in M \quad \text{and} \quad (I - P)\, x \perp M \tag{1.59}$$

or equivalently,

$$Px \in M \quad \text{and} \quad ((I-P)x, y) = 0 \quad \forall\, y \in M.$$

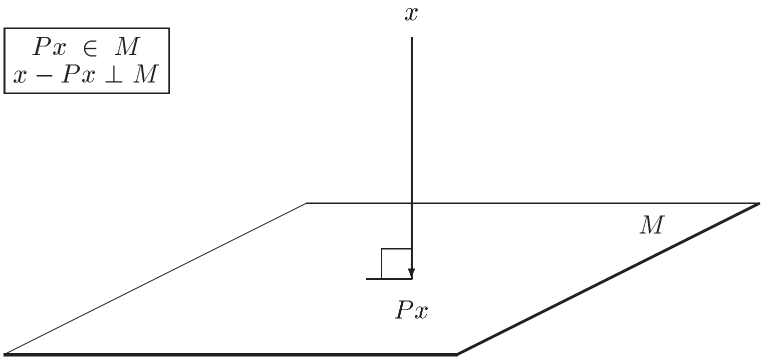


Figure 1.2 *Orthogonal projection of x onto a subspace M.*

It is interesting to consider the mapping P^H defined as the adjoint of P

$$(P^H x, y) = (x, Py), \quad \forall\, x,\ \forall\, y. \tag{1.60}$$

First note that P^H is also a projector because for all x and y,

$$((P^H)^2 x, y) = (P^H x, Py) = (x, P^2 y) = (x, Py) = (P^H x, y).$$

A consequence of the relation (1.60) is

$$\mathrm{Ker}(P^H) = \mathrm{Ran}(P)^\perp \tag{1.61}$$
$$\mathrm{Ker}(P) = \mathrm{Ran}(P^H)^\perp. \tag{1.62}$$

The above relations lead to the following proposition.

PROPOSITION 1.8 *A projector is orthogonal if and only if it is Hermitian.*

Proof. By definition, an orthogonal projector is one for which $\mathrm{Ker}(P) = \mathrm{Ran}(P)^\perp$. Therefore, by (1.61), if P is Hermitian, then it is orthogonal. Conversely, if P is orthogonal, then (1.61) implies $\mathrm{Ker}(P) = \mathrm{Ker}(P^H)$ while (1.62) implies $\mathrm{Ran}(P) = \mathrm{Ran}(P^H)$. Since P^H is a projector and since projectors are uniquely determined by their range and null spaces, this implies that $P = P^H$. ■

Given *any* unitary $n \times m$ matrix V whose columns form an orthonormal basis of $M = \mathrm{Ran}(P)$, we can represent P by the matrix $P = VV^H$. This is a particular case of the matrix representation of projectors (1.57). In addition to being idempotent, the linear mapping associated with this matrix satisfies the characterization given above, i.e.,

$$VV^H x \ \in M \quad \text{and} \quad (I - VV^H)x \ \in\ M^\perp.$$

It is important to note that this representation of the orthogonal projector P is not unique. In fact, any orthonormal basis V will give a different representation of P in the above form. As a consequence for any two orthogonal bases V_1, V_2 of M, we must have $V_1 V_1^H = V_2 V_2^H$, an equality which can also be verified independently; see Exercise 26.

1.12.4 PROPERTIES OF ORTHOGONAL PROJECTORS

When P is an orthogonal projector, then the two vectors Px and $(I-P)x$ in the decomposition $x = Px + (I-P)x$ are orthogonal. The following relation results:

$$\|x\|_2^2 = \|Px\|_2^2 + \|(I-P)x\|_2^2.$$

A consequence of this is that for any x,

$$\|Px\|_2 \le \|x\|_2.$$

Thus, the maximum of $\|Px\|_2/\|x\|_2$, for all x in $\mathbb{C}^n$ does not exceed one. In addition the value one is reached for any element in $\mathrm{Ran}(P)$. Therefore,

$$\|P\|_2 = 1$$

for any orthogonal projector P.

An orthogonal projector has only two eigenvalues: zero or one. Any vector of the range of P is an eigenvector associated with the eigenvalue one. Any vector of the null-space is obviously an eigenvector associated with the eigenvalue zero.

Next, an important optimality property of orthogonal projectors is established.

THEOREM 1.21 *Let P be the orthogonal projector onto a subspace M. Then for any given vector x in $\mathbb{C}^n$, the following is true:*

$$\min_{y \in M} \|x-y\|_2 = \|x - Px\|_2. \tag{1.63}$$

Proof. Let y be any vector of M and consider the square of its distance from x. Since $x - Px$ is orthogonal to M to which $Px - y$ belongs, then

$$\|x-y\|_2^2 = \|x - Px + (Px - y)\|_2^2 = \|x - Px\|_2^2 + \|(Px-y)\|_2^2.$$

Therefore, $\|x-y\|_2 \ge \|x - Px\|_2$ for all y in M. This establishes the result by noticing that the minimum is reached for $y = Px$. ∎

By expressing the conditions that define $y^* \equiv Px$ for an orthogonal projector P onto a subspace M, it is possible to reformulate the above result in the form of necessary and sufficient conditions which enable us to determine the best approximation to a given vector x in the least-squares sense.

COROLLARY 1.3 *Let a subspace M, and a vector x in $\mathbb{C}^n$ be given. Then*

$$\min_{y \in M} \|x-y\|_2 = \|x - y^*\|_2, \tag{1.64}$$

if and only if the following two conditions are satisfied,

$$\begin{cases} y^* & \in & M \\ x - y^* & \perp & M. \end{cases}$$

1.13 BASIC CONCEPTS IN LINEAR SYSTEMS

Linear systems are among the most important and common problems encountered in scientific computing. From the theoretical point of view, the problem is rather easy and explicit solutions using determinants exist. In addition, it is well understood when a solution exists, when it does not, and when there are infinitely many solutions. However, the numerical viewpoint is far more complex. Approximations may be available but it may be difficult to estimate how accurate they are. This clearly will depend on the data at hand, i.e., primarily on the coefficient matrix. This section gives a very brief overview of the existence theory as well as the sensitivity of the solutions.

1.13.1 EXISTENCE OF A SOLUTION

Consider the *linear system*

$$Ax = b. \tag{1.65}$$

Here, x is termed the *unknown* and b the *right-hand side.* When solving the linear system (1.65), we distinguish three situations.

Case 1 The matrix A is nonsingular. There is a unique solution given by $x = A^{-1}b$.

Case 2 The matrix A is singular and $b \in \mathrm{Ran}(A)$. Since $b \in \mathrm{Ran}(A)$, there is an x_0 such that $Ax_0 = b$. Then $x_0 + v$ is also a solution for any v in $\mathrm{Ker}(A)$. Since $\mathrm{Ker}(A)$ is at least one-dimensional, there are infinitely many solutions.

Case 3 The matrix A is singular and $b \notin \mathrm{Ran}(A)$. There are no solutions.

Example 1.4 The simplest illustration of the above three cases is with small diagonal matrices. Let

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 8 \end{pmatrix}.$$

Then A is nonsingular and there is a unique x given by

$$x = \begin{pmatrix} 0.5 \\ 2 \end{pmatrix}.$$

Now let

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Then A is singular and, as is easily seen, $b \in \mathrm{Ran}(A)$. For example, a particular element x_0 such that $Ax_0 = b$ is $x_0 = \left(\begin{smallmatrix} 0.5 \\ 0 \end{smallmatrix}\right)$. The null space of A consists of all vectors whose first component is zero, i.e., all vectors of the form $\left(\begin{smallmatrix} 0 \\ \alpha \end{smallmatrix}\right)$. Therefore, there are infinitely many solution which are given by

$$x(\alpha) = \begin{pmatrix} 0.5 \\ \alpha \end{pmatrix} \quad \forall\, \alpha.$$

Finally, let A be the same as in the previous case, but define the right-hand side as

$$b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

In this case there are no solutions because the second equation cannot be satisfied.

1.13.2 PERTURBATION ANALYSIS

Consider the linear system (1.65) where A is an $n \times n$ nonsingular matrix. Given any matrix E, the matrix $A(\epsilon) = A + \epsilon E$ is nonsingular for ϵ small enough, i.e., for $\epsilon \le \alpha$ where α is some small number; see Exercise 32. Assume that we perturb the data in the above system, i.e., that we perturb the matrix A by ϵE and the right-hand side b by ϵe. The solution $x(\epsilon)$ of the perturbed system satisfies the equation,

$$(A + \epsilon E)x(\epsilon) = b + \epsilon e. \tag{1.66}$$

Let $\delta(\epsilon) = x(\epsilon) - x$. Then,

$$\begin{aligned}(A + \epsilon E)\delta(\epsilon) &= (b + \epsilon e) - (A + \epsilon E)x \\ &= \epsilon\ (e - Ex) \\ \delta(\epsilon) &= \epsilon\ (A + \epsilon E)^{-1}(e - Ex).\end{aligned}$$

As an immediate result, the function $x(\epsilon)$ is differentiable at $\epsilon = 0$ and its derivative is given by

$$x'(0) = \lim_{\epsilon \to 0} \frac{\delta(\epsilon)}{\epsilon} = A^{-1}\,(e - Ex)\,. \tag{1.67}$$

The size of the derivative of $x(\epsilon)$ is an indication of the size of the variation that the solution $x(\epsilon)$ undergoes when the data, i.e., the pair $[A, b]$ is perturbed in the direction $[E, e]$. In absolute terms, a small variation $[\epsilon E, \epsilon e]$ will cause the solution to vary by roughly $\epsilon x'(0) = \epsilon A^{-1}(e - Ex)$. The relative variation is such that

$$\frac{\|x(\epsilon) - x\|}{\|x\|} \le \epsilon \|A^{-1}\| \left(\frac{\|e\|}{\|x\|} + \|E\| \right) + o(\epsilon).$$

Using the fact that $\|b\| \le \|A\|\|x\|$ in the above equation yields

$$\frac{\|x(\epsilon) - x\|}{\|x\|} \le \epsilon \|A\|\|A^{-1}\| \left(\frac{\|e\|}{\|b\|} + \frac{\|E\|}{\|A\|} \right) + o(\epsilon) \tag{1.68}$$

which relates the relative variation in the solution to the relative sizes of the perturbations. The quantity

$$\kappa(A) = \|A\|\,\|A^{-1}\|$$

is called the *condition number* of the linear system (1.65) with respect to the norm $\|.\|$. The condition number is relative to a norm. When using the standard norms $\|.\|_p, p = 1, \ldots, \infty$, it is customary to label $\kappa(A)$ with the same label as the associated norm. Thus,

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p.$$

For large matrices, the determinant of a matrix is almost never a good indication of "near" singularity or degree of sensitivity of the linear system. The reason is that $\det(A)$ is the product of the eigenvalues which depends very much on a scaling of a matrix, whereas the condition number of a matrix is scaling-invariant. For example, for $A = \alpha I$ the determinant is $\det(A) = \alpha^n$, which can be very small if $|\alpha| < 1$, whereas $\kappa(A) = 1$ for any of the standard norms.

In addition, small eigenvalues do not always give a good indication of poor conditioning. Indeed, a matrix can have all its eigenvalues equal to one yet be poorly conditioned.

Example 1.5 The simplest example is provided by matrices of the form

$$A_n = I + \alpha e_1 e_n^T$$

for large α. The inverse of A_n is

$$A_n^{-1} = I - \alpha e_1 e_n^T$$

and for the ∞-norm we have

$$\|A_n\|_\infty = \|A_n^{-1}\|_\infty = 1 + |\alpha|$$

so that

$$\kappa_\infty(A_n) = (1 + |\alpha|)^2.$$

For a large α, this can give a very large condition number, whereas all the eigenvalues of A_n are equal to unity.

When an iterative procedure is used for solving a linear system, we typically face the problem of choosing a good stopping procedure for the algorithm. Often a residual norm,

$$\|r\| = \|b - A\tilde{x}\|$$

is available for some current approximation $\tilde{x}$ and an estimate of the absolute error $\|x - \tilde{x}\|$ or the relative error $\|x - \tilde{x}\|/\|x\|$ is desired. The following simple relation is helpful in this regard,

$$\frac{\|x - \tilde{x}\|}{\|x\|} \le \kappa(A) \, \frac{\|r\|}{\|b\|}.$$

It is necessary to have an estimate of the condition number $\kappa(A)$ in order to exploit the above relation.

EXERCISES

1. Verify that the Euclidean inner product defined by (1.4) does indeed satisfy the general definition of inner products on vector spaces.

2. Show that two eigenvectors associated with two distinct eigenvalues are linearly independent. In a more general sense, show that a family of eigenvectors associated with distinct eigenvalues forms a linearly independent family.

3. Show that if λ is any nonzero eigenvalue of the matrix AB, then it is also an eigenvalue of the matrix BA. Start with the particular case where A and B are square and B is nonsingular, then consider the more general case where A, B may be singular or even rectangular (but such that AB and BA are square).

4. Let A be an $n \times n$ orthogonal matrix, i.e., such that $A^H A = D$, where D is a diagonal matrix. Assuming that D is nonsingular, what is the inverse of A? Assuming that $D > 0$, how can A be transformed into a unitary matrix (by operations on its rows or columns)?

5. Show that the Frobenius norm is consistent. Can this norm be associated to two vector norms via (1.7)? What is the Frobenius norm of a diagonal matrix? What is the p-norm of a diagonal matrix (for any p)?

6. Find the Jordan canonical form of the matrix:

$$A = \begin{pmatrix} 1 & 2 & -4 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix}.$$

   Same question for the matrix obtained by replacing the element a_{33} by 1.

7. Give an alternative proof of Theorem 1.3 on the Schur form by starting from the Jordan canonical form. [Hint: Write $A = XJX^{-1}$ and use the QR decomposition of X.]

8. Show from the definition of determinants used in Section 1.2 that the characteristic polynomial is a polynomial of degree n for an $n \times n$ matrix.

9. Show that the characteristic polynomials of two similar matrices are equal.

10. Show that

$$\lim_{k \to \infty} \|A^k\|^{1/k} = \rho(A),$$

   for any matrix norm. [Hint: Use the Jordan canonical form.]

11. Let X be a nonsingular matrix and, for any matrix norm $\|.\|$, define $\|A\|_X = \|AX\|$. Show that this is indeed a matrix norm. Is this matrix norm consistent? Show the same for $\|XA\|$ and $\|YAX\|$ where Y is also a nonsingular matrix. These norms are not, in general, associated with any vector norms, i.e., they can't be defined by a formula of the form (1.7). Why? What about the particular case $\|A\|' = \|XAX^{-1}\|$?

12. Find the field of values of the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

   and verify that it is not equal to the convex hull of its eigenvalues.

13. Show that for a skew-Hermitian matrix S,

$$\Re e(Sx, x) = 0 \quad \text{for any } x \in \mathbb{C}^n .$$

14. Given an arbitrary matrix S, show that if $(Sx, x) = 0$ for all x in $\mathbb{C}^n$, then it is true that

$$(Sy, z) + (Sz, y) = 0 \quad \forall\, y,\ z\ \in\ \mathbb{C}^n .$$

[Hint: Expand $(S(y + z), y + z)$.]

15. Using the results of the previous two problems, show that if (Ax, x) is real for all x in $\mathbb{C}^n$, then A must be Hermitian. Would this result be true if the assumption were to be replaced by: (Ax, x) *is real for all real* x? Explain.

16. The definition of a positive definite matrix is that (Ax, x) be real and positive for all real vectors x. Show that this is equivalent to requiring that the Hermitian part of A, namely, $\frac{1}{2}(A + A^H)$, be (Hermitian) positive definite.

17. Let $A_1 = B^{-1}C$ and $A_2 = CB$ where C is a Hermitian matrix and B is a Hermitian Positive Definite matrix. Are A_1 and A_2 Hermitian *in general*? Show that A_1 and A_2 are Hermitian (self-adjoint) with respect to the B-inner product.

18. Let a matrix A be such that $A^H = p(A)$ where p is a polynomial. Show that A is normal. [Hint: Use Lemma 1.2.]

19. Show that A is normal iff its Hermitian and skew-Hermitian parts, as defined in Section 1.11, commute.

20. Let A be a Hermitian matrix and B a Hermitian Positive Definite matrix defining a B-inner product. Show that A is Hermitian (self-adjoint) with respect to the B-inner product if and only if A and B commute. What condition must satisfy B for the same condition to hold in the more general case where A is not Hermitian?

21. Let A be a real symmetric matrix and λ an eigenvalue of A. Show that if u is an eigenvector associated with λ, then so is $\bar{u}$. As a result, prove that for any eigenvalue of a real symmetric matrix, there is an associated eigenvector which is real.

22. Show that a Hessenberg matrix H such that $h_{j+1,j} \neq 0, j = 1, 2, \ldots, n-1$, cannot be derogatory.

23. Prove all the properties listed in Proposition 1.6.

24. Let A be an M-matrix and u, v two nonnegative vectors such that $v^T A^{-1} u < 1$. Show that $A - uv^T$ is an M-matrix.

25. Show that if $O \le A \le B$ then $O \le A^T A \le B^T B$. Conclude that under the same assumption, we have $\|A\|_2 \le \|B\|_2$.

26. Show that for two orthogonal bases V_1, V_2 of the same subspace M of $\mathbb{C}^n$ we have $V_1 V_1^H x = V_2 V_2^H x,\ \forall\, x$.

27. What are the eigenvalues of a projector? What about its eigenvectors?

28. Show that if two projectors P_1 and P_2 commute, then their product $P = P_1 P_2$ is a projector. What are the range and kernel of P?

29. Consider the matrix A of size $n \times n$ and the vector $x \in \mathbb{R}^n$,

$$A = \begin{pmatrix} 1 & -1 & -1 & -1 & \dots & -1 \\ 0 & 1 & -1 & -1 & \dots & -1 \\ 0 & 0 & 1 & -1 & \dots & -1 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ 1/2 \\ 1/4 \\ 1/8 \\ \vdots \\ 1/2^{n-1} \end{pmatrix}.$$

- ***a.*** Compute Ax, $\|Ax\|_2$, and $\|x\|_2$.
- ***b.*** Show that $\|A\|_2 \geq \sqrt{n}$.
- ***c.*** Give a lower bound for $\kappa_2(A)$.

30. What is the inverse of the matrix A of the previous exercise? Give an expression of $\kappa_1(A)$ and $\kappa_\infty(A)$ based on this.

31. Find a small rank-one perturbation which makes the matrix A in Exercise 29 singular. Derive a lower bound for the singular values of A.

32. Consider a nonsingular matrix A. Given any matrix E, show that there exists α such that the matrix $A(\epsilon) = A + \epsilon E$ is nonsingular for all $\epsilon < \alpha$. What is the largest possible value for α satisfying the condition? [Hint: Consider the eigenvalues of the generalized eigenvalue problem $Au = \lambda Eu$.]

NOTES AND REFERENCES. For additional reading on the material presented in this chapter, see Golub and Van Loan [108], Datta [64], Stewart [202], and Varga [213]. Details on matrix eigenvalue problems can be found in Gantmacher's book [100] and Wilkinson [227]. An excellent treatise of nonnegative matrices is in the book by Varga [213] which remains a good reference on iterative methods more three decades after its first publication. Another book with state-of-the-art coverage on iterative methods up to the very beginning of the 1970s is the book by Young [232] which covers M-matrices and related topics in great detail. For a good overview of the linear algebra aspects of matrix theory and a complete proof of Jordan's canonical form, Halmos [117] is recommended. ■

CHAPTER

2

DISCRETIZATION OF PDES

Partial Differential Equations (PDEs) constitute by far the biggest source of sparse matrix problems. The typical way to solve such equations is to *discretize* them, i.e., to approximate them by equations that involve a finite number of unknowns. The matrix problems that arise from these discretizations are generally large and sparse, i.e., they have very few nonzero entries. There are several different ways to discretize a Partial Differential Equation. The simplest method uses *finite difference* approximations for the partial differential operators. The *Finite Element Method* replaces the original function by a function which has some degree of smoothness over the global domain, but which is piecewise polynomial on simple cells, such as small triangles or rectangles. This method is probably the most general and well understood discretization technique available. In between these two methods, there are a few conservative schemes called *Finite Volume Methods*, which attempt to emulate continuous *conservation laws* of physics. This chapter introduces these three different discretization methods.

2.1 PARTIAL DIFFERENTIAL EQUATIONS

Physical phenomena are often modeled by equations that relate several partial derivatives of physical quantities, such as forces, momentums, velocities, energy, temperature, etc. These equations rarely have a *closed-form* (explicit) solution. In this chapter, a few types of Partial Differential Equations are introduced, which will serve as models throughout the book. Only one- or two-dimensional problems are considered, and the space variables are denoted by x in the case of one-dimensional problems or x_1 and x_2 for two-dimensional problems. In two dimensions, x denotes the "vector" of components (x_1, x_2).

2.1.1 ELLIPTIC OPERATORS

One of the most common Partial Differential Equations encountered in various areas of engineering is Poisson's equation:

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = f, \text{ for } x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ in } \Omega \tag{2.1}$$

where Ω is a bounded, open domain in $\mathbb{R}^2$. Here, x_1, x_2 are the two space variables.

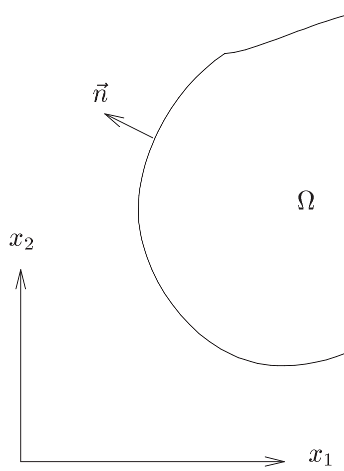


Figure 2.1 *Domain Ω for Poisson's equation.*

The above equation is to be satisfied only for points that are located at the interior of the domain Ω. Equally important are the conditions that must be satisfied on the *boundary* Γ of Ω. These are termed *boundary conditions*, and they come in three common types:

Dirichlet condition	$u(x)$	$=$	$\phi(x)$
Neumann condition	$\frac{\partial u}{\partial \vec{n}}(x)$	$=$	0
Cauchy condition	$\frac{\partial u}{\partial \vec{n}}(x) + \alpha(x)u(x)$	$=$	$\gamma(x)$

The vector $\vec{n}$ usually refers to a unit vector that is normal to Γ and directed outwards. Note that the Neumann boundary conditions are a particular case of the Cauchy conditions with $\gamma = \alpha = 0$. For a given unit vector, $\vec{v}$ with components v_1 and v_2, the directional derivative $\partial u / \partial \vec{v}$ is defined by

$$\begin{aligned} \frac{\partial u}{\partial \vec{v}}(x) &= \lim_{h \to 0} \frac{u(x + h\vec{v}) - u(x)}{h} \\ &= \frac{\partial u}{\partial x_1}(x) v_1 + \frac{\partial u}{\partial x_2}(x) v_2 \qquad (2.2) \\ &= \nabla u . \vec{v} \qquad (2.3) \end{aligned}$$

where ∇u is the gradient of u,

$$\nabla u = \begin{pmatrix} \frac{\partial u}{\partial x_1} \\ \frac{\partial u}{\partial x_2} \end{pmatrix}, \tag{2.4}$$

and the dot in (2.3) indicates a dot product of two vectors in $\mathbb{R}^2$.

In reality, Poisson's equation is often a limit case of a time-dependent problem. It can, for example, represent the steady-state temperature distribution in a region Ω when there is a heat source f that is constant with respect to time. The boundary conditions should then model heat loss across the boundary Γ.

The particular case where $f(x) = 0$, i.e., the equation

$$\Delta u = 0,$$

to which boundary conditions must be added, is called the *Laplace equation* and its solutions are called *harmonic functions*.

Many problems in physics have boundary conditions of *mixed type*, e.g., of Dirichlet type in one part of the boundary and of Cauchy type in another. Another observation is that the Neumann conditions do not define the solution uniquely. Indeed, if u is a solution, then so is $u + c$ for any constant c.

The operator

$$\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$$

is called the *Laplacean operator* and appears in many models of physical and mechanical phenomena. These models often lead to more general elliptic operators of the form

$$\begin{aligned} L &= \frac{\partial}{\partial x_1}\left(a\frac{\partial}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(a\frac{\partial}{\partial x_2}\right) \\ &= \nabla.\ (a\nabla) \end{aligned} \tag{2.5}$$

where the scalar function a depends on the coordinate and may represent some specific parameter of the medium, such as density, porosity, etc. At this point it may be useful to recall some notation which is widely used in physics and mechanics. The ∇ operator can be considered as a vector consisting of the components $\frac{\partial}{\partial x_1}$ and $\frac{\partial}{\partial x_2}$. When applied to a scalar function u, this operator is nothing but the *gradient* operator, since it yields a vector with the components $\frac{\partial u}{\partial x_1}$ and $\frac{\partial u}{\partial x_2}$ as is shown in (2.4). The dot notation allows dot products of vectors in $\mathbb{R}^2$ to be defined. These vectors can include partial differential operators. For example, the dot product $\nabla.u$ of ∇ with $u = \binom{u_1}{u_2}$ yields the scalar quantity,

$$\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2},$$

which is called the *divergence* of the vector function $\vec{u} = \binom{u_1}{u_2}$. Applying this *divergence operator* to $u = a\nabla$, where a is a scalar function, yields the L operator in (2.5). The divergence of the vector function $\vec{v}$ is often denoted by div $\vec{v}$ or $\nabla.\vec{v}$. Thus,

$$\text{div } \vec{v} = \nabla.\vec{v} = \frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2}.$$

The closely related operator

$$\begin{aligned} L &= \frac{\partial}{\partial x_1}\left(a_1 \frac{\partial}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(a_2 \frac{\partial}{\partial x_2}\right) \\ &= \nabla \ \ (\vec{a}.\ \nabla) \end{aligned} \tag{2.6}$$

is a further generalization of the Laplacean operator Δ in the case where the medium is *anisotropic* and *inhomogeneous*. The coefficients a_1, a_2 depend on the space variable x and reflect the position as well as the directional dependence of the material properties, such as porosity in the case of fluid flow or dielectric constants in electrostatics. In fact, the above operator can be viewed as a particular case of $L = \nabla.(A\nabla)$, where A is a 2×2 matrix which acts on the two components of ∇.

2.1.2 THE CONVECTION DIFFUSION EQUATION

Many physical problems involve a combination of "diffusion" and "convection" phenomena. Such phenomena are modeled by the convection-diffusion equation

$$\frac{\partial u}{\partial t} + b_1 \frac{\partial u}{\partial x_1} + b_2 \frac{\partial u}{\partial x_2} = \nabla.(a\nabla)u + f$$

or

$$\frac{\partial u}{\partial t} + \vec{b}.\nabla u = \nabla.(a\nabla)u + f$$

the steady-state version of which can be written as

$$-\nabla.(a\nabla)u + \vec{b}.\nabla u = f. \tag{2.7}$$

Problems of this type are often used as model problems because they represent the simplest form of conservation of mass in fluid mechanics. Note that the vector $\vec{b}$ is sometimes quite large, which may cause some difficulties either to the discretization schemes or to the iterative solution techniques.

2.2 FINITE DIFFERENCE METHODS

The *finite difference* method is based on local approximations of the partial derivatives in a Partial Differential Equation, which are derived by low order Taylor series expansions. The method is quite simple to define and rather easy to implement. Also, it is particularly appealing for simple regions, such as rectangles, and when uniform meshes are used. The matrices that result from these discretizations are often well structured, which means that they typically consist of a few nonzero diagonals. Another advantage is that there are a number of "fast solvers" for constant coefficient problems, which can deliver the solution in logarithmic time per grid point. This means the total number of operations is of the order of $n \log(n)$ where n is the total number of discretization points. This section gives an overview of finite difference discretization techniques.

2.2.1 BASIC APPROXIMATIONS

The simplest way to approximate the first derivative of a function u at the point x is via the formula

$$\left(\frac{du}{dx}\right)(x) \approx \frac{u(x+h) - u(x)}{h}. \tag{2.8}$$

When u is differentiable at x, then the limit of the above ratio when h tends to zero is the derivative of u at x. For a function that is C^4 in the neighborhood of x, we have by Taylor's formula

$$u(x+h) = u(x) + h\frac{du}{dx} + \frac{h^2}{2}\frac{d^2u}{dx^2} + \frac{h^3}{6}\frac{d^3u}{dx^3} + \frac{h^4}{24}\frac{d^4u}{dx^4}(\xi_+), \tag{2.9}$$

for some ξ_+ in the interval $(x, x+h)$. Therefore, the above approximation (2.8) satisfies

$$\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} - \frac{h}{2}\frac{d^2u(x)}{dx^2} + O(h^2). \tag{2.10}$$

The formula (2.9) can be rewritten with h replaced by $-h$ to obtain

$$u(x-h) = u(x) - h\frac{du}{dx} + \frac{h^2}{2}\frac{d^2u}{dx^2} - \frac{h^3}{6}\frac{d^3u}{dx^3} + \frac{h^4}{24}\frac{d^4u(\xi_-)}{dx^4}, \tag{2.11}$$

in which ξ_- belongs to the interval $(x-h, x)$. Adding (2.9) and (2.11), dividing through by h^2, and using the mean value theorem for the fourth order derivatives results in the following approximation of the second derivative

$$\frac{d^2u(x)}{dx^2} = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - \frac{h^2}{12}\frac{d^4u(\xi)}{dx^4}, \tag{2.12}$$

where $\xi_- \le \xi \le \xi_+$. The above formula is called a *centered difference approximation* of the second derivative since the point at which the derivative is being approximated is the center of the points used for the approximation. The dependence of this derivative on the values of u at the points involved in the approximation is often represented by a "stencil" or "molecule," shown in Figure 2.2.

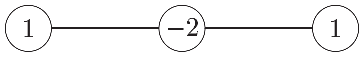


Figure 2.2 *The three-point stencil for the centered difference approximation to the second order derivative.*

The approximation (2.8) for the first derivative is *forward* rather than centered. Also, a *backward* formula can be used which consists of replacing h with $-h$ in (2.8). The two formulas can also be averaged to obtain the *centered difference* formula:

$$\frac{du(x)}{dx} \approx \frac{u(x+h) - u(x-h)}{2\,h}. \tag{2.13}$$

It is easy to show that the above centered difference formula is of the second order, while (2.8) is only first order accurate. Denoted by δ^+ and δ^-, the forward and backward difference operators are defined by

$$\delta^+ u(x) = u(x+h) - u(x) \tag{2.14}$$
$$\delta^- u(x) = u(x) - u(x-h). \tag{2.15}$$

All previous approximations can be rewritten using these operators.

In addition to standard first order and second order derivatives, it is sometimes necessary to approximate the second order operator

$$\frac{d}{dx}\left[a(x)\,\frac{d}{dx}\right].$$

A centered difference formula for this, which has second order accuracy, is given by

$$\begin{aligned} \frac{d}{dx}\left[a(x)\,\frac{du}{dx}\right] &= \frac{1}{h^2}\delta^+\left(a_{i-1/2}\,\delta^- u\right) + O(h^2) \\ &\approx \frac{a_{i+1/2}(u_{i+1}-u_i) - a_{i-1/2}(u_i - u_{i-1})}{h^2}. \end{aligned} \tag{2.16}$$

2.2.2 DIFFERENCE SCHEMES FOR THE LAPLACEAN OPERATOR

If the approximation (2.12) is used for both the $\frac{\partial^2}{\partial x_1^2}$ and $\frac{\partial^2}{\partial x_2^2}$ terms in the Laplacean operator, using a mesh size of h_1 for the x_1 variable and h_2 for the x_2 variable, the following second order accurate approximation results:

$$\begin{aligned} \Delta u(x) \approx\ & \frac{u(x_1+h_1,x_2) - 2u(x_1,x_2) + u(x-h_1,x_2)}{h_1^2} + \\ & \frac{u(x_1,x_2+h_2) - 2u(x_1,x_2) + u(x_1,x_2-h_2)}{h_2^2}. \end{aligned}$$

In the particular case where the mesh sizes h_1 and h_2 are the same and equal to a mesh size h, the approximation becomes

$$\begin{aligned} \Delta u(x) \approx \frac{1}{h^2}\,[&u(x_1+h,x_2) + u(x_1-h,x_2) + u(x_1,x_2+h) \\ &+ u(x_1,x_2-h) - 4u(x_1,x_2)]\,, \end{aligned} \tag{2.17}$$

which is called the five-point centered approximation to the Laplacean. The stencil of this finite difference approximation is illustrated in (a) of Figure 2.3.

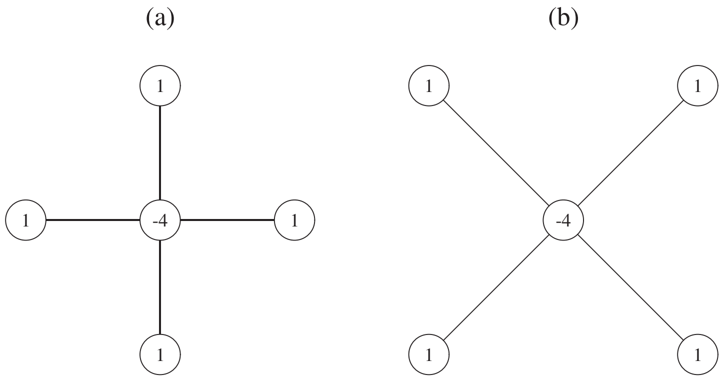


Figure 2.3 *Five-point stencils for the centered difference approximation to the Laplacean operator: (a) the standard stencil, (b) the skewed stencil.*

Another approximation may be obtained by exploiting the four points $u(x_1 \pm h, x_2 \pm h)$ located on the two diagonal lines from $u(x_1, x_2)$. These points can be used in the same manner as in the previous approximation except that the mesh size has changed. The corresponding stencil is illustrated in (b) of Figure 2.3.

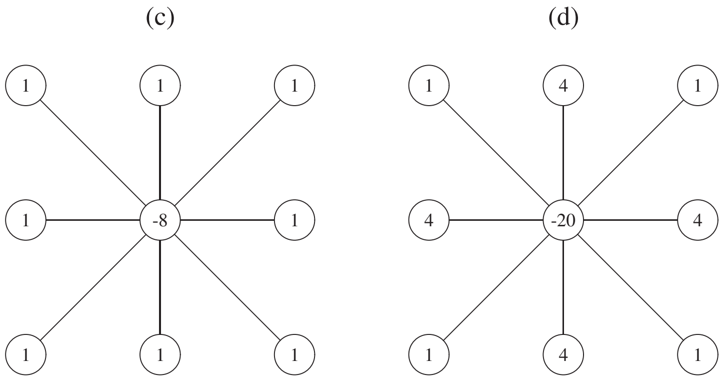


Figure 2.4 *Two nine-point centered difference stencils for the Laplacean operator.*

The approximation (2.17) is second order accurate and the error takes the form

$$\frac{h^2}{12}\left(\frac{\partial^4 u}{\partial^4 x_1} + \frac{\partial^4 u}{\partial^4 x_2}\right) + O(h^3).$$

There are other schemes that utilize nine-point formulas as opposed to five-point formulas. Two such schemes obtained by combining the standard and skewed stencils described above are shown in Figure 2.4. Both approximations (c) and (d) are second order accurate. However, (d) is sixth order for harmonic functions, i.e., functions whose Laplacean is zero.

2.2.3 FINITE DIFFERENCES FOR 1-D PROBLEMS

Consider the one-dimensional equation,

$$-u''(x) = f(x) \text{ for } x \in (0,1) \tag{2.18}$$

$$u(0) = u(1) = 0. \tag{2.19}$$

The interval [0,1] can be discretized uniformly by taking the $n+2$ points

$$x_i = i \times h,\ i = 0, \ldots, n+1$$

where $h = 1/(n+1)$. Because of the Dirichlet boundary conditions, the values $u(x_0)$ and $u(x_{n+1})$ are known. At every other point, an approximation u_i is sought for the exact solution $u(x_i)$.

If the centered difference approximation (2.12) is used, then by the equation (2.18) expressed at the point x_i, the unknowns u_i, u_{i-1}, u_{i+1} satisfy the relation

$$-u_{i-1} + 2u_i - u_{i+1} = h^2\, f_i,$$

in which $f_i \equiv f(x_i)$. Notice that for $i = 1$ and $i = n$, the equation will involve u_0 and u_{n+1} which are known quantities, both equal to zero in this case. Thus, for $n = 6$, the linear system obtained is of the form

$$Ax = f$$

where

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}.$$

2.2.4 UPWIND SCHEMES

Consider now the one-dimensional version of the convection-diffusion equation (2.7) in which the coefficients a and b are constant, and $f = 0$, using Dirichlet boundary conditions,

$$\begin{cases} -a\,u'' + b\,u' &= 0, \quad 0 < x < L = 1 \\ u(0) = 0,\ u(L) &= 1. \end{cases} \tag{2.20}$$

In this particular case, it is easy to verify that the exact solution to the above equation is given by

$$u(x) = \frac{1 - e^{Rx}}{1 - e^{R}}$$

where R is the so-called Péclet number defined by $R = bL/a$. Now consider the approximate solution provided by using the centered difference schemes seen above, for both the first- and second order derivatives. The equation for unknown number i becomes

$$b\frac{u_{i+1}-u_{i-1}}{2h} - a\frac{u_{i+1}-2u_i+u_{i-1}}{h^2} = 0,$$

or, defining $c = Rh/2$,

$$-(1-c)u_{i+1} + 2u_i - (1+c)u_{i-1} = 0. \tag{2.21}$$

This is a second order homogeneous linear difference equation and the usual way to solve it is to seek a general solution in the form $u_j = r^j$. Substituting in (2.21), r must satisfy

$$(1-c)r^2 - 2r + (c+1) = 0.$$

Therefore, $r_1 = 1$ is a root and the second root is $r_2 = (1+c)/(1-c)$. The general solution of the above difference equation is now sought as a linear combination of the two solutions corresponding to these two roots,

$$u_i = \alpha r_1^i + \beta r_2^i = \alpha + \beta\left(\frac{1+c}{1-c}\right)^i.$$

Because of the boundary condition $u_0 = 0$, it is necessary that $\beta = -\alpha$. Likewise, the boundary condition $u_{n+1} = 1$ yields

$$\alpha = \frac{1}{1-\sigma^{n+1}} \quad \text{with} \quad \sigma \equiv \frac{1+c}{1-c}.$$

Thus, the solution is

$$u_i = \frac{1-\sigma^i}{1-\sigma^{n+1}}.$$

When $h > 2/R$ the factor σ becomes negative and the above approximations will oscillate around zero. In contrast, the exact solution is positive and monotone in the range $[0,1]$. In this situation the solution is very inaccurate regardless of the arithmetic. In other words, the scheme itself creates the oscillations. To avoid this, a small enough mesh h can be taken to ensure that $c < 1$. The resulting approximation is in much better agreement with the exact solution. Unfortunately, this condition can limit the mesh size too drastically for large values of b.

Note that when $b < 0$, the oscillations disappear since $\sigma < 1$. In fact, a linear algebra interpretation of the oscillations comes from comparing the tridiagonal matrices obtained from the discretization. Again, for the case $n = 6$, the tridiagonal matrix resulting from discretizing the equation (2.7) takes the form

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1+c & & & & \\ -1-c & 2 & -1+c & & & \\ & -1-c & 2 & -1+c & & \\ & & -1-c & 2 & -1+c & \\ & & & -1-c & 2 & -1+c \\ & & & & -1-c & 2 \end{pmatrix}.$$

The above matrix is no longer a diagonally dominant M-matrix. Observe that if the backward difference formula for the first order derivative is used, we obtain

$$b\,\frac{u_i-u_{i-1}}{h} - a\frac{u_{i-1}-2u_i+u_{i+1}}{h^2} = 0.$$

Then (weak) diagonal dominance is preserved if $b > 0$. This is because the new matrix obtained for the above backward scheme is

$$A = \frac{1}{h^2}\begin{pmatrix} 2+c & -1 & & & & \\ -1-c & 2+c & -1 & & & \\ & -1-c & 2+c & -1 & & \\ & & -1-c & 2+c & -1 & \\ & & & -1-c & 2+c & -1 \\ & & & & -1-c & 2+c \end{pmatrix}$$

where c is now defined by $c = Rh$. Each diagonal term a_{ii} gets reinforced by the positive term c while each subdiagonal term $a_{i,i-1}$ increases by the same amount in absolute value. In the case where $b < 0$, the forward difference formula

$$b\frac{u_{i+1} - u_i}{h} - a\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = 0$$

can be used to achieve the same effect. Generally speaking, if b depends on the space variable x, the effect of weak-diagonal dominance can be achieved by simply adopting the following discretization known as an "upwind scheme":

$$b\,\frac{\delta_i^* u_i}{h} - a\,\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = 0$$

where

$$\delta_i^* = \begin{cases} \delta_i^- & \text{if} \quad b > 0 \\ \delta_i^+ & \text{if} \quad b < 0. \end{cases}$$

The above difference scheme can be rewritten by introducing the sign function $\mathrm{sign}\,(b) = |b|/b$. The approximation to u' at x_i is then defined by

$$u'(x_i) \approx \frac{1}{2}(1 - \mathrm{sign}(b))\frac{\delta^+ u_i}{h} + \frac{1}{2}(1 + \mathrm{sign}(b))\frac{\delta^- u_i}{h}.$$

Making use of the notation

$$(x)^+ = \frac{1}{2}(x + |x|), \quad (x)^- = \frac{1}{2}(x - |x|), \tag{2.22}$$

a slightly more elegant formula can be obtained by expressing the approximation of the product $b(x_i)u'(x_i)$,

$$\begin{aligned} b(x_i)u'(x_i) &\approx \frac{1}{2}(b_i - |b_i|)\frac{\delta^+ u_i}{h} + \frac{1}{2}(b_i + |b_i|)\frac{\delta^- u_i}{h} \\ &\approx \frac{1}{h}\left[-b_i^+ u_{i-1} + |b_i| u_i + b_i^- u_{i+1}\right], \end{aligned} \tag{2.23}$$

where b_i stands for $b(x_i)$. The diagonal term in the resulting tridiagonal matrix is nonnegative, the offdiagonal terms are nonpositive, and the diagonal term is the negative sum of the offdiagonal terms. This property characterizes upwind schemes.

A notable disadvantage of upwind schemes is the low order of approximation which they yield. An advantage is that upwind schemes yield linear systems that are easier to solve by iterative methods.

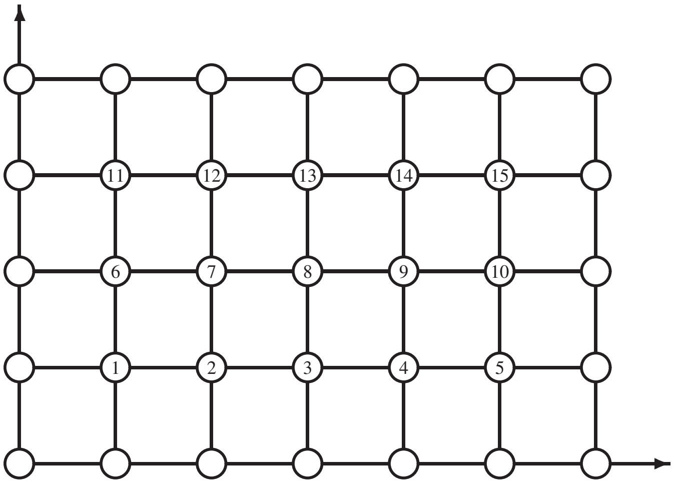


Figure 2.5 *Natural ordering of the unknowns for a* 7×5 *two-dimensional grid.*

2.2.5 FINITE DIFFERENCES FOR 2-D PROBLEMS

Similar to the previous case, consider this simple problem,

$$-\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}\right) = f \quad \text{in } \Omega \tag{2.24}$$

$$u = 0 \quad \text{on } \Gamma \tag{2.25}$$

where Ω is now the rectangle $(0, l_1) \times (0, l_2)$ and Γ its boundary. Both intervals can be discretized uniformly by taking $n_1 + 2$ points in the x_1 direction and $n_2 + 2$ points in the x_2 directions:

$$x_{1,i} = i \times h_1, i = 0, \ldots, n_1 + 1 \quad x_{2,j} = j \times h_2, j = 0, \ldots, n_2 + 1$$

where

$$h_1 = \frac{l_1}{n_1 + 1} \quad h_2 = \frac{l_2}{n_2 + 1}.$$

Since the values at the boundaries are known, we number only the interior points, i.e., the points $(x_{1,i}, x_{2,j})$ with $0 < i < n_1$ and $0 < j < n_2$. The points are labeled from the bottom up, one horizontal line at a time. This labeling is called *natural ordering* and is shown in Figure 2.5 for the very simple case when $n_1 = 7$ and $n_2 = 5$. The pattern of the matrix corresponding to the above equations appears in Figure 2.6.

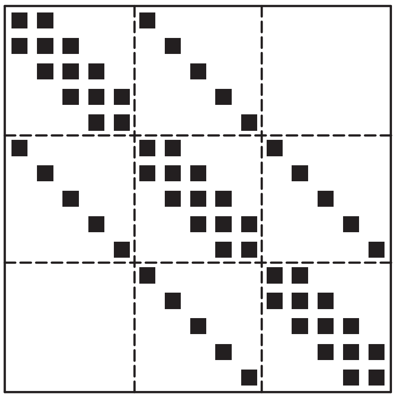


Figure 2.6 *Pattern of matrix associated with the 7×5 finite difference mesh of Figure 2.5.*

To be more accurate, the matrix has the following block structure:

$$A = \frac{1}{h^2}\begin{pmatrix} B & -I & \\ -I & B & -I \\ & -I & B \end{pmatrix} \quad \text{with} \quad B = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}.$$

2.3 THE FINITE ELEMENT METHOD

The finite element method is best illustrated with the solution of a simple elliptic Partial Differential Equation in a two-dimensional space. Consider again Poisson's equation (2.24) with the Dirichlet boundary condition (2.25), where Ω is a bounded open domain in $\mathbb{R}^2$ and Γ its boundary. The Laplacean operator

$$\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$$

appears in many models of physical and mechanical phenomena. Equations involving the more general elliptic operators (2.5) and (2.6) can be treated in the same way as Poisson's equation (2.24) and (2.25), at least from the viewpoint of the numerical solutions techniques.

An essential ingredient for understanding the finite element method is *Green's formula*. The setting for this formula is an open set Ω whose boundary consists of a closed and smooth curve Γ as illustrated in Figure 2.1. A vector-valued function $\vec{v} = \binom{v_1}{v_2}$, which is continuously differentiable in Ω, is given. The *divergence theorem* in two-dimensional spaces states that

$$\int_\Omega \text{div}\vec{v}\ dx = \int_\Gamma \vec{v}.\vec{n}\ \ ds. \tag{2.26}$$

The dot in the right-hand side represents a dot product of two vectors in $\mathbb{R}^2$. In this case it is between the vector $\vec{v}$ and the unit vector $\vec{n}$ which is normal to Γ at the point of consideration and oriented outward. To derive Green's formula, consider a scalar function v and a vector function $\vec{w} = \binom{w_1}{w_2}$. By standard differentiation,

$$\nabla.(v\vec{w}) = (\nabla v).\vec{w} + v\nabla.\vec{w},$$

which expresses $\nabla v.\vec{w}$ as

$$\nabla v.\vec{w} = -v\nabla.\vec{w} + \nabla.(v\vec{w}). \tag{2.27}$$

Integrating the above equality over Ω and using the divergence theorem, we obtain

$$\begin{aligned}\int_\Omega \nabla v.\vec{w}\ \ dx &= -\int_\Omega v\nabla.\vec{w}\ \ dx + \int_\Omega \nabla.(v\vec{w})\ dx \\ &= -\int_\Omega v\nabla.\vec{w}\ \ dx + \int_\Gamma v\vec{w}.\vec{n}\ ds.\end{aligned} \tag{2.28}$$

The above equality can be viewed as a generalization of the standard integration by part formula in calculus. Green's formula results from (2.28) by simply taking a vector $\vec{w}$ which is itself a gradient of a scalar function u, namely, $\vec{w} = \nabla u$,

$$\int_\Omega \nabla v.\nabla u\ \ dx = -\int_\Omega v\nabla.\nabla u\ \ dx + \int_\Gamma v\nabla u.\vec{n}\ ds.$$

Observe that $\nabla.\nabla u = \Delta u$. Also the function $\nabla u.\vec{n}$ is called the *normal derivative* and is denoted by

$$\nabla u.\vec{n} = \frac{\partial u}{\partial \vec{n}}.$$

With this, we obtain Green's formula

$$\int_\Omega \nabla v.\nabla u\ \ dx = -\int_\Omega v\Delta u\ \ dx + \int_\Gamma v\frac{\partial u}{\partial \vec{n}}\ ds. \tag{2.29}$$

We now return to the initial problem (2.24-2.25). To solve this problem approximately, it is necessary to (1) take approximations to the unknown function u, and (2) translate the equations into a system which can be solved numerically. The options for approximating u are numerous. However, the primary requirement is that these approximations should be in a (small) finite dimensional space. There are also some additional desirable numerical properties. For example, it is difficult to approximate high degree polynomials numerically. To extract systems of equations which yield the solution, it is common to use the *weak formulation* of the problem. Let us define

$$\begin{aligned}a(u,v) &\equiv \int_\Omega \nabla u.\nabla v\ \ dx = \int_\Omega \left(\frac{\partial u}{\partial x_1}\frac{\partial v}{\partial x_1} + \frac{\partial u}{\partial x_2}\frac{\partial v}{\partial x_2}\right)\ dx, \\ (f,v) &\equiv \int_\Omega fv\ dx.\end{aligned}$$

An immediate property of the functional a is that it is *bilinear.* That means that it is linear with respect to u and v, namely,

$$\begin{aligned}a(\mu_1 u_1 + \mu_2 u_2, v) &= \mu_1 a(u_1,v) + \mu_2 a(u_2,v), \quad \forall \mu_1,\mu_2\ \in \mathbb{R}, \\ a(u, \lambda_1 v_1 + \lambda_2 v_2) &= \lambda_1 a(u,v_1) + \lambda_2 a(u,v_2), \quad \forall \lambda_1,\lambda_2\ \in\ \mathbb{R}.\end{aligned}$$

Notice that (u, v) denotes the L_2-inner product of u and v in Ω, i.e.,

$$(u, v) = \int_\Omega u(x)v(x)dx,$$

then, for functions satisfying the Dirichlet boundary conditions, which are at least twice differentiable, Green's formula (2.29) shows that

$$a(u, v) = -(\Delta u, v).$$

The weak formulation of the initial problem (2.24-2.25) consists of selecting a subspace of reference V of L^2 and then defining the following problem:

$$\text{Find} \quad u \in V \quad \text{such that} \quad a(u, v) = (f, v), \quad \forall\, v \in V. \tag{2.30}$$

In order to understand the usual choices for the space V, note that the definition of the weak problem only requires the dot products of the gradients of u and v and the functions f and v to be L_2–integrable. The most general V under these conditions is the space of all functions whose derivatives up to the first order are in L_2. This is known as $H^1(\Omega)$. However, this space does not take into account the boundary conditions. The functions in V must be restricted to have zero values on Γ. The resulting space is called $H_0^1(\Omega)$.

The finite element method consists of approximating the weak problem by a finite-dimensional problem obtained by replacing V with a subspace of functions that are defined as low-degree polynomials on small pieces (elements) of the original domain.

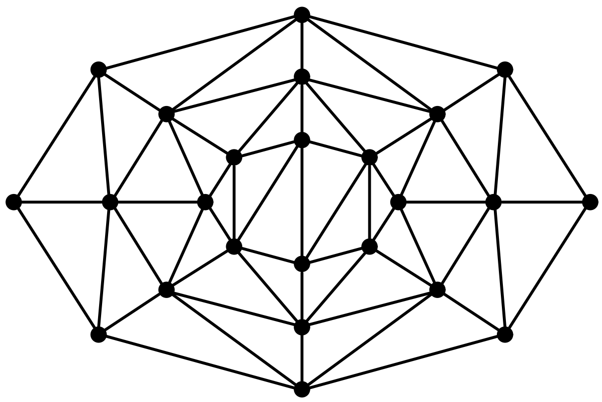


Figure 2.7 *Finite element triangulation of a domain.*

Consider a region Ω in the plane which is triangulated as shown in Figure 2.7. In this example, the domain is simply an ellipse but the external enclosing curve is not shown. The original domain is thus approximated by the union Ω_h of m triangles K_i,

$$\Omega_h = \bigcup_{i=1}^{m} K_i.$$

For the triangulation to be valid, these triangles must have no vertex that lies on the edge of any other triangle. The *mesh size* h is defined by

$$h = \max_{i=1,\ldots,m} \text{diam}(K_i)$$

where diam(K), the diameter of a triangle K, is the length of its longest side.

Then the finite dimensional space V_h is defined as the space of all functions which are piecewise linear and continuous on the polygonal region Ω_h, and which vanish on the boundary Γ. More specifically,

$$V_h = \{\phi \mid \phi_{|\Omega_h} \text{continuous}, \phi_{|\Gamma_h} = 0,\ \phi_{|K_j} \text{ linear} \,\forall\, j\}.$$

Here, $\phi_{|X}$ represents the restriction of the function ϕ to the subset X. If $x_j, j = 1, \ldots, n$ are the nodes of the triangulation, then a function ϕ_j in V_h can be associated with each node x_j, so that the family of functions ϕ_j's satisfies the following conditions:

$$\phi_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{if} \quad x_i = x_j \\ 0 & \text{if} \quad x_i \neq x_j \end{cases}. \tag{2.31}$$

These conditions define $\phi_i, i = 1, \ldots, n$ uniquely. In addition, the ϕ_i's form a basis of the space V_h.

Each function of V_h can be expressed as

$$\phi(x) = \sum_{i=1}^{n} \xi_i \phi_i(x).$$

The finite element approximation consists of writing the Galerkin condition (2.30) for functions in V_h. This defines the approximate problem:

$$\text{Find} \quad u \in V_h \quad \text{such that} \quad a(u, v) = (f, v), \ \forall \ v \in V_h. \tag{2.32}$$

Since u is in V_h, there are n degrees of freedom. By the linearity of a with respect to v, it is only necessary to impose the condition $a(u, \phi_i) = (f, \phi_i)$ for $i = 1, \ldots, n$. This results in n constraints.

Writing the desired solution u in the basis $\{\phi_i\}$ as

$$u = \sum_{i=1}^{n} \xi_i \phi_i(x)$$

and substituting in (2.32) gives the linear problem

$$\sum_{j=1}^{n} \alpha_{ij} \xi_i = \beta_i \tag{2.33}$$

where

$$\alpha_{ij} = a(\phi_i, \phi_j), \quad \beta_i = (f, \phi_i).$$

The above equations form a linear system of equations

$$Ax = b,$$

in which the coefficients of A are the α_{ij}'s; those of b are the β_j's. In addition, A is a *Symmetric Positive Definite* matrix. Indeed, it is clear that

$$\int_\Omega \nabla\phi_i \nabla\phi_j \, dx = \int_\Omega \nabla\phi_j \nabla\phi_i \, dx,$$

which means that $\alpha_{ij} = \alpha_{ji}$. To see that A is positive definite, first note that $a(u, u) \geq 0$ for any function u. If $a(\phi, \phi) = 0$ for a function in V_h, then it must be true that $\nabla \phi = 0$ *almost everywhere* in Ω_h. Since ϕ is linear in each triangle and continuous, then it is clear that it must be constant on all Ω. Since, in addition, it vanishes on the boundary, then it must be equal to zero on all of Ω. The result follows by exploiting the relation

$$(A\xi, \xi) = a(\phi, \phi) \quad \text{with} \quad \phi = \sum_{i=1}^{n} \xi_i \phi_i,$$

which is valid for any vector $\{\xi_i\}_{i=1,\dots,n}$.

Another important observation is that the matrix A is also sparse. Indeed, α_{ij} is nonzero only when the two basis functions ϕ_i and ϕ_j have common support triangles, or equivalently when the nodes i and j are the vertices of a common triangle. Specifically, for a given node i, the coefficient α_{ij} will be nonzero only when the node j is one of the nodes of a triangle that is adjacent to node i.

In practice, the matrix is built by summing up the contributions of all triangles by applying the formula

$$a(\phi_i, \phi_j) = \sum_K a_K(\phi_i, \phi_j)$$

in which the sum is over all the triangles K and

$$a_K(\phi_i, \phi_j) = \int_K \nabla \phi_i \, \nabla \phi_j \, dx.$$

Note that $a_K(\phi_i, \phi_j)$ is zero unless the nodes i and j are both vertices of K. Thus, a triangle contributes nonzero values to its three vertices from the above formula. The 3×3 matrix

$$A_K = \begin{pmatrix} a_K(\phi_i, \phi_i) & a_K(\phi_i, \phi_j) & a_K(\phi_i, \phi_k) \\ a_K(\phi_j, \phi_i) & a_K(\phi_j, \phi_j) & a_K(\phi_j, \phi_k) \\ a_K(\phi_k, \phi_i) & a_K(\phi_k, \phi_j) & a_K(\phi_k, \phi_k) \end{pmatrix}$$

associated with the triangle $K(i, j, k)$ with vertices i, j, k is called an *element stiffness matrix*. In order to form the matrix A, it is necessary to sum up all the contributions $a_K(\phi_k, \phi_m)$ to the position k, m of the matrix. This process is called an *assembly* process. In the assembly, the matrix is computed as

$$A = \sum_{e=1}^{nel} A^{[e]}, \tag{2.34}$$

in which nel is the number of elements. Each of the matrices $A^{[e]}$ is of the form

$$A^{[e]} = P_e A_{K_e} P_e^T$$

where A_{K_e} is the element matrix for the element K_e as defined above. Also P_e is an $n \times 3$ Boolean connectivity matrix which maps the coordinates of the 3×3 matrix A_{K_e} into the coordinates of the full matrix A.

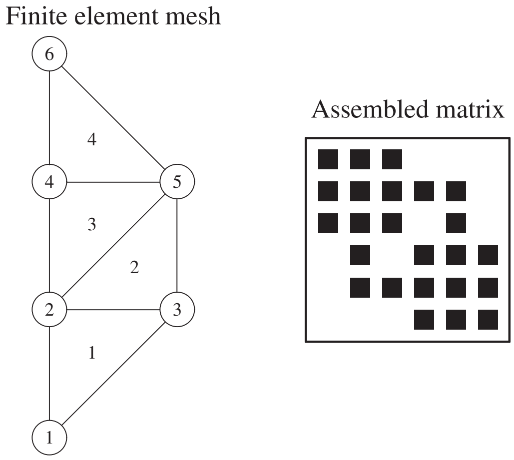


Figure 2.8 *A simple finite element mesh and the pattern of the corresponding assembled matrix.*

Example 2.1 The assembly process can be illustrated with a very simple example. Consider the finite element mesh shown in Figure 2.8. The four elements are numbered from bottom to top as indicated by the labels located at their centers. There are six nodes in this mesh and their labeling is indicated in the circled numbers. The four matrices $A^{[e]}$ associated with these elements are shown in Figure 2.9. Thus, the first element will contribute to the nodes $1, 2, 3$, the second to nodes $2, 3, 5$, the third to nodes $2, 4, 5$, and the fourth to nodes $4, 5, 6$.

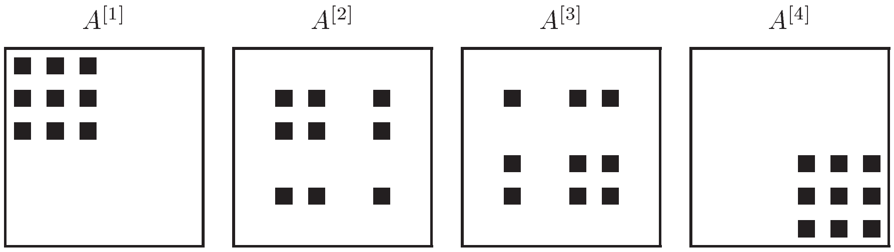


Figure 2.9 *The element matrices $A^{[e]}$, $e = 1, \ldots, 4$ for the finite element mesh shown in Figure 2.8.*

In fact there are two different ways to represent and use the matrix A. We can form all the element matrices one by one and then we can store them, e.g., in an $nel \times 3 \times 3$ rectangular array. This representation is often called the *unassembled form* of A. Then the matrix A may be assembled if it is needed. However, element stiffness matrices can also be used in different ways without having to assemble the matrix. For example, *frontal techniques* are direct solution methods that take the linear system in unassembled form and compute the solution by a form of Gaussian elimination. There are also iterative solution techniques which work directly with unassembled matrices. One of the main operations required in many iterative methods is to compute $y = Ax$, the product of the matrix A by an arbitrary vector x. In unassembled form, this can be achieved as follows:

$$y = Ax = \sum_{e=1}^{nel} A^{[e]}x \; = \; \sum_{e=1}^{nel} P_e A_{K_e}(P_e^T x). \tag{2.35}$$

Thus, the product $P_e^T x$ gathers the x data associated with the e-element into a 3-vector consistent with the ordering of the matrix A_{K_e}. After this is done, this vector must be multiplied by A_{K_e}. Finally, the result is added to the current y vector in appropriate locations determined by the P_e array. This sequence of operations must be done for each of the nel elements.

A more common and somewhat more appealing technique is to perform the assembly of the matrix. All the elements are scanned one by one and the nine associated contributions $a_K(\phi_k, \phi_m)$, $k, m \; \in \; \{i, j, k\}$ added to the corresponding positions in the global "stiffness" matrix. The assembled matrix must now be stored but the element matrices may be discarded. The structure of the assembled matrix depends on the ordering of the nodes. To facilitate the computations, a widely used strategy transforms all triangles into a reference triangle with vertices $(0, 0), (0, 1), (1, 0)$. The area of the triangle is then simply the determinant of the Jacobian of the transformation that allows passage from one set of axes to the other.

Simple boundary conditions such as Neumann or Dirichlet do not cause any difficulty. The simplest way to handle Dirichlet conditions is to include boundary values as unknowns and modify the assembled system to incorporate the boundary values. Thus, each equation associated with the boundary point in the assembled system is replaced by the equation $u_i = f_i$. This yields a small identity block hidden within the linear system. For Neumann conditions, Green's formula will give rise to the equations

$$\int_\Omega \nabla u.\nabla \phi_j \;\; dx = \int_\Omega f\phi_j dx + \int_\Gamma \phi_j \frac{\partial u}{\partial \vec{n}} \, ds, \tag{2.36}$$

which will involve the Neumann data $\frac{\partial u}{\partial \vec{n}}$ over the boundary. Since the Neumann data is typically given at some points only (the boundary nodes), linear interpolation (trapezoidal rule) or the mid-line value (midpoint rule) can be used to approximate the integral. Note that (2.36) can be viewed as the j-th equation of the linear system. Another important point is that if the boundary conditions are only of Neumann type, then the resulting system is singular. An equation must be removed, or the linear system must be solved by taking this singularity into account.

2.4 MESH GENERATION AND REFINEMENT

Generating a finite element triangulation can be done quite easily by exploiting some initial grid and then refining the mesh a few times either uniformly or in specific areas. The simplest refinement technique consists of taking the three midpoints of a triangle, thus creating four smaller triangles from a larger triangle and losing one triangle, namely, the original one. A systematic use of one level of this strategy is illustrated for the mesh in Figure 2.8, and is shown in Figure 2.10.

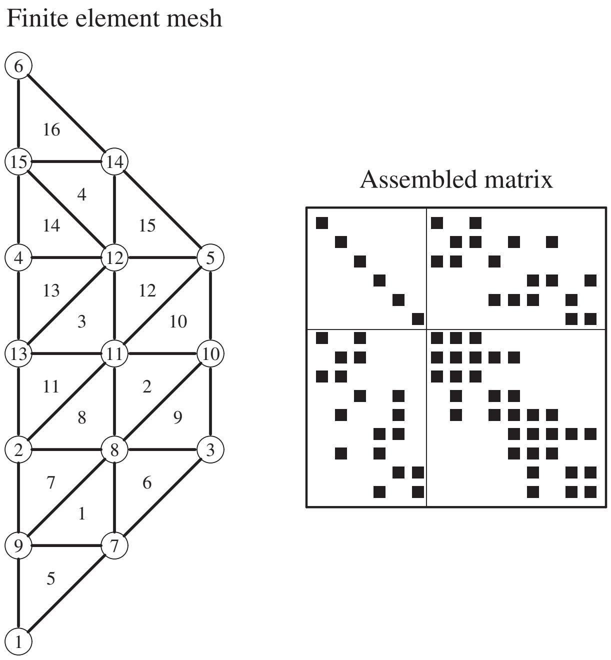


Figure 2.10 *The simple finite element mesh of Figure 2.8 after one level of refinement and the corresponding matrix.*

One advantage of this approach is that it preserves the angles of the original triangulation. This is an important property since the angles on a good quality triangulation must satisfy certain bounds. On the other hand, the indiscriminate use of the uniform refinement strategy may lead to some inefficiencies. Indeed, it is desirable to introduce more triangles in areas where the solution is likely to have large variations. In terms of vertices, midpoints should be introduced only where needed. To obtain standard finite element triangles, the points that have been created on the edges of a triangle must be linked to existing vertices in the triangle. This is because no vertex of a triangle is allowed to lie on the edge of another triangle.

Figure 2.11 shows three possible cases that can arise. The original triangle is (a). In (b), only one new vertex (numbered 4) has appeared on one edge of the triangle and it is joined to the vertex opposite to it. In (c), two new vertices appear inside the original triangle. There is no alternative but to join vertices (4) and (5). However, after this is done, either vertices (4) and (3) or vertices (1) and (5) must be joined. If angles are desired that will not become too small with further refinements, the second choice is clearly better in this case. In fact, various strategies for improving the quality of the triangles have been devised. The final case (d) corresponds to the “uniform refinement” case where all edges have been split in two. There are three new vertices and four new elements, and the larger initial element is removed.

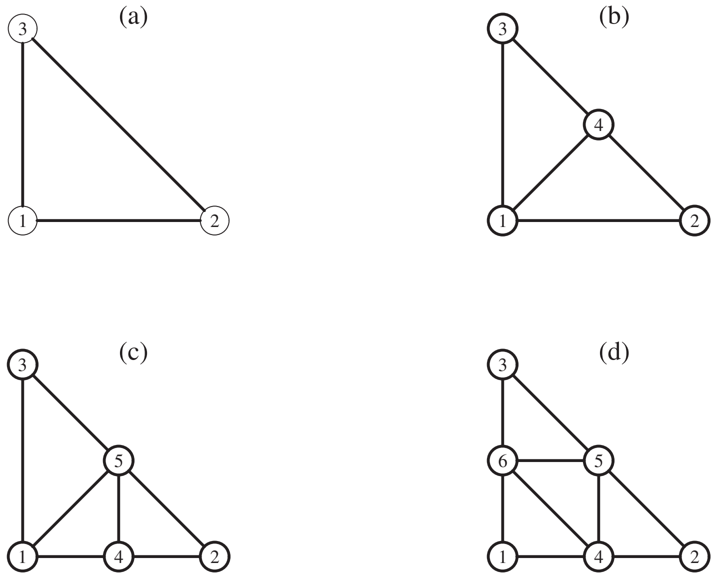


Figure 2.11 *Original triangle (a) and three possible refinement scenarios.*

2.5 FINITE VOLUME METHOD

The finite volume method is geared toward the solution of conservation laws of the form:

$$\frac{\partial u}{\partial t} + \nabla.\vec{F} = Q. \tag{2.37}$$

In the above equation, $\vec{F}(u,t)$ is a certain vector function of u and time, possibly nonlinear. This is called the "flux vector." The *source term* Q is a function of space and time. We now apply the principle used in the weak formulation, described before. Multiply both sides by a test function w, and take the integral

$$\int_\Omega w\,\frac{\partial u}{\partial t}\,dx + \int_\Omega w\,\nabla.\vec{F}\,dx = \int_\Omega w\,Q\,dx.$$

Then integrate by part using formula (2.28) for the second term on the left-hand side to obtain

$$\int_\Omega w\frac{\partial u}{\partial t}\,dx - \int_\Omega \nabla w.\vec{F}\,dx + \int_\Gamma w\,\vec{F}.\vec{n}\,ds = \int_\Omega w\,Q\;\,dx.$$

Consider now a *control volume* consisting, for example, of an elementary triangle K_i in the two-dimensional case, such as those used in the finite element method. Take for w a function w_i whose value is one on the triangle and zero elsewhere. The second term in the above equation vanishes and the following relation results:

$$\int_{K_i} \frac{\partial u}{\partial t} \, dx + \int_{\Gamma_i} \vec{F}.\vec{n} \, ds = \int_{K_i} Q \;\; dx. \tag{2.38}$$

The above relation is at the basis of the finite volume approximation. To go a little further, the assumptions will be simplified slightly by taking a vector function $\vec{F}$ that is linear with respect to u. Specifically, assume

$$\vec{F} = \begin{pmatrix} \lambda_1 u \\ \lambda_2 u \end{pmatrix} \equiv \vec{\lambda} u.$$

Note that, in this case, the term $\nabla.\vec{F}$ in (2.37) becomes $\vec{F}(u) = \vec{\lambda}.\nabla u$. In addition, the right-hand side and the first term in the left-hand side of (2.38) can be approximated as follows:

$$\int_{K_i} \frac{\partial u}{\partial t} \, dx \approx \frac{\partial u_i}{\partial t} |K_i|, \qquad \int_{K_i} Q \, dx \approx q_i |K_i|.$$

Here, $|K_i|$ represents the volume [1] of K_i, and q_i is some average value of Q in the cell K_i. These are crude approximations but they serve the purpose of illustrating the scheme.

The finite volume equation (2.38) yields

$$\frac{\partial u_i}{\partial t} |K_i| + \vec{\lambda}. \int_{\Gamma_i} u \, \vec{n} \, ds = q_i |K_i|. \tag{2.39}$$

The contour integral

$$\int_{\Gamma_i} u \, \vec{n} \, ds$$

is the sum of the integrals over all edges of the control volume. Let the value of u on each edge j be approximated by some "average" $\bar{u}_j$. In addition, s_j denotes the length of each edge and a common notation is

$$\vec{s}_j = s_j \vec{n}_j.$$

Then the contour integral is approximated by

$$\vec{\lambda}. \int_{\Gamma_i} u \, \vec{n} \, ds \approx \sum_{edges} \bar{u}_j \vec{\lambda}.\vec{n}_j s_j = \sum_{edges} \bar{u}_j \vec{\lambda}.\vec{s}_j. \tag{2.40}$$

The situation in the case where the control volume is a simple triangle is depicted in Figure 2.12. The unknowns are the approximations u_i of the function u associated with each cell. These can be viewed as approximations of u at the centers of gravity of each cell i. This type of model is called *cell-centered* finite volume approximations. Other techniques based on using approximations on the vertices of the cells are known as *cell-vertex* finite volume techniques.

[1] In two dimensions, "volume" is considered to mean area.

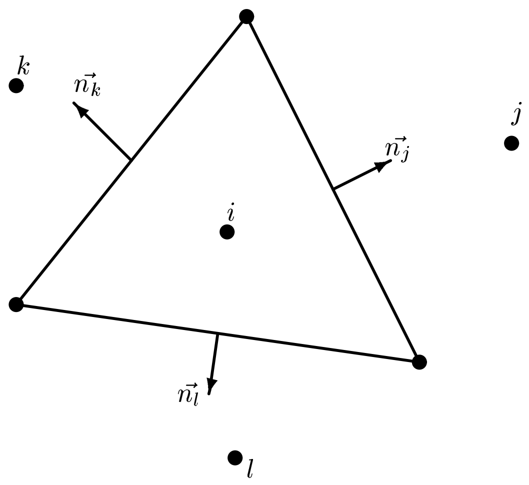


Figure 2.12 *Finite volume cell associated with node i and three neighboring cells.*

The value $\bar{u}_j$ required in (2.40) can be taken simply as the average between the approximation u_i of u in cell i and the approximation u_j in the cell j on the other side of the edge

$$\bar{u}_j = \frac{1}{2}(u_j + u_i). \tag{2.41}$$

This gives

$$\frac{\partial u_i}{\partial t}|K_i| \; + \frac{1}{2}\sum_j (u_i + u_j)\vec{\lambda}.\vec{s}_j = q_i|K_i|.$$

One further simplification takes place by observing that

$$\sum_j \vec{s}_j = 0$$

and therefore

$$\sum_j \; u_i\vec{\lambda}.\vec{s}_j = u_i\vec{\lambda}. \; \sum_j \vec{s}_j = 0.$$

This yields

$$\frac{\partial u_i}{\partial t}\;|K_i| + \frac{1}{2}\sum_j u_j\vec{\lambda}.\vec{s}_j = q_i|K_i|.$$

In the above equation, the summation is over all the neighboring cells j. One problem with such simple approximations is that they do not account for large gradients of u in the components. In finite volume approximations, it is typical to exploit upwind schemes which are more suitable in such cases. By comparing with one-dimensional upwind schemes, it can be easily seen that the suitable modification to (2.41) is as follows:

$$\bar{u}_j = \frac{1}{2}(u_j + u_i) - \frac{1}{2}\ \text{sign}\left(\vec{\lambda}.\vec{s}_j\right)\ (u_j - u_i). \tag{2.42}$$

This gives

$$\frac{\partial u_i}{\partial t}\ |K_i| + \sum_j \vec{\lambda}.\vec{s}_j\ \left(\frac{1}{2}(u_j + u_i) - \frac{1}{2}\ \text{sign}(\vec{\lambda}.\vec{s}_j)(u_j - u_i)\right) = q_i|K_i|.$$

Now write

$$\frac{\partial u_i}{\partial t}\ |K_i| + \sum_j \left(\frac{1}{2}(u_j + u_i)\vec{\lambda}.\vec{s}_j - \frac{1}{2}|\vec{\lambda}.\vec{s}_j|(u_j - u_i)\right) = q_i|K_i|$$

$$\frac{\partial u_i}{\partial t}\ |K_i| + \sum_j \left(u_i(\vec{\lambda}.\vec{s}_j)^+ + u_j(\vec{\lambda}.\vec{s}_j)^-\right) = q_i|K_i|$$

where

$$(z)^{\pm} \equiv \frac{z\ \pm\ |z|}{2}.$$

The equation for cell i takes the form

$$\frac{\partial u_i}{\partial t}\ |K_i| + \beta_i u_i + \sum_j \alpha_{ij} u_j = q_i|K_i|,$$

where

$$\beta_i = \sum_j (\vec{\lambda}.\vec{s}_j)^+\ \geq 0, \tag{2.43}$$

$$\alpha_{ij} = (\vec{\lambda}.\vec{s}_j)^-\ \leq 0. \tag{2.44}$$

Thus, the diagonal elements of the matrix are nonnegative, while its offdiagonal elements are nonpositive. In addition, the row-sum of the elements, i.e., the sum of all elements in the same row, is equal to zero. This is because

$$\beta_i + \sum_j \alpha_{ij} = \sum_j (\vec{\lambda}.\vec{s}_j)^+ + \sum_j (\vec{\lambda}.\vec{s}_j)^- = \sum_j \vec{\lambda}.\vec{s}_j = \vec{\lambda}.\sum_j \vec{s}_j = 0.$$

The matrices obtained have the same desirable property of weak diagonal dominance seen in the one-dimensional case. A disadvantage of upwind schemes, whether in the context of irregular grids or in one-dimensional equations, is the loss of accuracy due to the low order of the schemes.

EXERCISES

1. Derive Forward Difference formulas similar to (2.8), i.e., involving $u(x), u(x+h), u(x+2h), \ldots$, which are of second and third order. Write down the discretization errors explicitly.

2. Derive a Centered Difference formula for the first derivative, similar to (2.13), which is at least of third order.

3. Show that the Upwind Difference scheme described in 2.2.4, when a and $\vec{b}$ are constant, is stable for the model problem (2.7).

4. Develop the two nine-point formulas illustrated in Figure 2.4. Find the corresponding discretization errors. [Hint: Combine $\frac{1}{3}$ of the five-point formula (2.17) plus $\frac{2}{3}$ of the same formula based on the diagonal stencil $\{(x,y),(x+h,y+h)+(x+h,y-h),(x-h,y+h),(x-h,y-h)\}$ to get one formula. Use the reverse combination $\frac{2}{3}$, $\frac{1}{3}$ to get the other formula.]

5. Consider a (two-dimensional) rectangular mesh which is discretized as in the finite difference approximation. Show that the finite volume approximation to $\vec{\lambda}.\nabla u$ yields the same matrix as an upwind scheme applied to the same problem. What would be the mesh of the equivalent upwind finite difference approximation?

6. Show that the right-hand side of equation (2.16) can also be written as

$$\frac{1}{h^2}\delta^-\left(a_{i+\frac{1}{2}}\ \delta^+ u\right).$$

7. Show that the formula (2.16) is indeed second order accurate for functions that are in C^4.

8. Show that the functions ϕ_i's defined by (2.31) form a basis of V_h.

9. Develop the equivalent of Green's formula for the elliptic operator L defined in (2.6).

10. Write a short FORTRAN or C program to perform a matrix-by-vector product when the matrix is stored in unassembled form.

11. Consider the finite element mesh of Example 2.1. Compare the number of operations required to perform a matrix-by-vector product when the matrix is in assembled and in unassembled form. Compare also the storage required in each case. For a general finite element matrix, what can the ratio be between the two in the worst case (consider only linear approximations on triangular elements) for arithmetic? Express the number of operations in terms of the number of nodes and edges of the mesh. You may make the assumption that the maximum number of elements that are adjacent to a given node is p (e.g., $p = 8$).

12. Let K be a polygon in $\mathbb{R}^2$ with m edges, and let $\vec{s}_j = s_j\vec{n}_j$, for $j = 1,\ldots,m$, where s_j is the length of the j-th edge and $\vec{n}_j$ is the unit outward normal at the j-th edge. Use the divergence theorem to prove that $\sum_{j=1}^{m}\vec{s}_j = 0$.

NOTES AND REFERENCES. The material in this chapter is based on several sources. For a basic description of the finite element method, the book by C. Johnson is a good reference [128]. Axelsson and Barker [16] gives a treatment which includes various solution techniques emphasizing iterative techniques. For finite difference and finite volume methods, we recommend C. Hirsch [121], which also gives a good description of the equations and solution methods for fluid flow problems. ■

CHAPTER

3

SPARSE MATRICES

As described in the previous chapter, standard discretizations of Partial Differential Equations typically lead to large and *sparse* matrices. A sparse matrix is defined, somewhat vaguely, as a matrix which has very few nonzero elements. But, in fact, a matrix can be termed sparse whenever special techniques can be utilized to take advantage of the large number of zero elements and their locations. These sparse matrix techniques begin with the idea that the zero elements need not be stored. One of the key issues is to define data structures for these matrices that are well suited for efficient implementation of standard solution methods, whether direct or iterative. This chapter gives an overview of sparse matrices, their properties, their representations, and the data structures used to store them.

3.1 INTRODUCTION

The natural idea to take advantage of the zeros of a matrix and their location was initiated by engineers in various disciplines. In the simplest case involving banded matrices, special techniques are straightforward to develop. Electrical engineers dealing with electrical networks in the 1960s were the first to exploit sparsity to solve general sparse linear systems for matrices with irregular structure. The main issue, and the first addressed by sparse matrix technology, was to devise direct solution methods for linear systems. These had to be economical, both in terms of storage and computational effort. Sparse direct solvers can handle very large problems that cannot be tackled by the usual "dense" solvers.

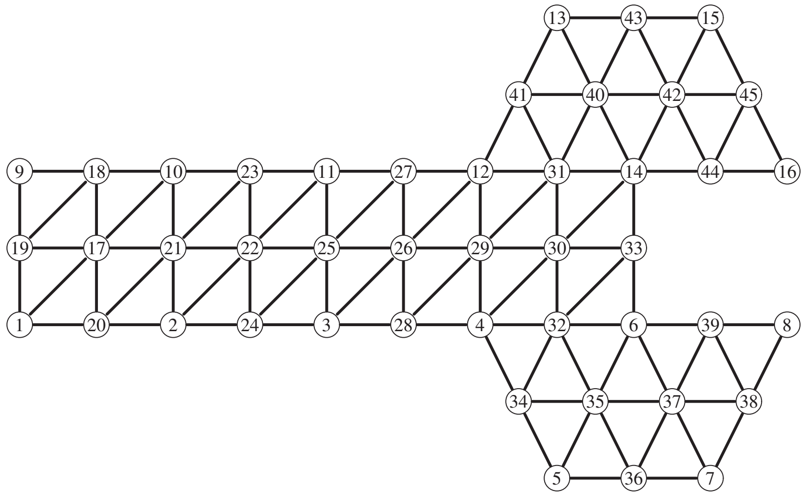


Figure 3.1 *A finite element grid model.*

Essentially, there are two broad types of sparse matrices: *structured* and *unstructured*. A structured matrix is one whose nonzero entries form a regular pattern, often along a small number of diagonals. Alternatively, the nonzero elements may lie in blocks (dense submatrices) of the same size, which form a regular pattern, typically along a small number of (block) diagonals. A matrix with irregularly located entries is said to be irregularly structured. The best example of a regularly structured matrix is a matrix that consists of only a few diagonals. Finite difference matrices on rectangular grids, such as the ones seen in the previous chapter, are typical examples of matrices with regular structure. Most finite element or finite volume techniques applied to complex geometries lead to irregularly structured matrices. Figure 3.2 shows a small irregularly structured sparse matrix associated with the finite element grid problem shown in Figure 3.1.

The distinction between the two types of matrices may not noticeably affect direct solution techniques, and it has not received much attention in the past. However, this distinction can be important for iterative solution methods. In these methods, one of the essential operations is matrix-by-vector products. The performance of these operations can differ significantly on high performance computers, depending on whether they are regularly structured or not. For example, on vector computers, storing the matrix by diagonals is ideal, but the more general schemes may suffer because they require indirect addressing.

The next section discusses graph representations of sparse matrices. This is followed by an overview of some of the storage schemes used for sparse matrices and an explanation of how some of the simplest operations with sparse matrices can be performed. Then sparse linear system solution methods will be covered. Finally, Section 3.7 discusses test matrices.

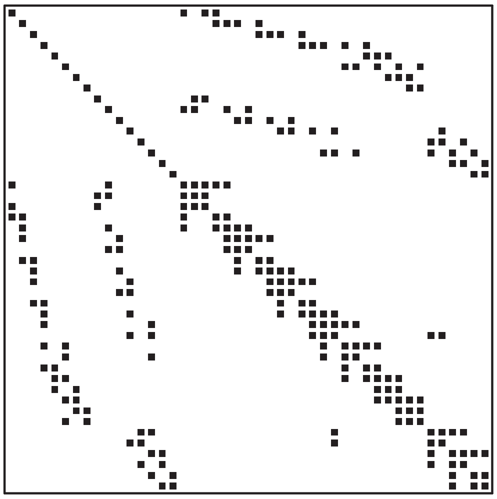


Figure 3.2 *Sparse matrix associated with the finite element grid of Figure 3.1.*

3.2 GRAPH REPRESENTATIONS

Graph theory is an ideal tool for representing the structure of sparse matrices and for this reason it plays a major role in sparse matrix techniques. For example, graph theory is the key ingredient used in unraveling parallelism in sparse Gaussian elimination or in preconditioning techniques. In the following section, graphs are discussed in general terms and then their applications to finite element or finite difference matrices are discussed.

3.2.1 GRAPHS AND ADJACENCY GRAPHS

Remember that a graph is defined by two sets, a set of vertices

$$V = \{v_1, v_2, \ldots, v_n\},$$

and a set of edges E which consists of pairs (v_i, v_j), where v_i, v_j are elements of V, i.e.,

$$E \subseteq V \times V.$$

This graph $G = (V, E)$ is often represented by a set of points in the plane linked by a directed line between the points that are connected by an edge. A graph is a way of representing a binary relation between objects of a set V. For example, V can represent the major cities of the world. A line is drawn between any two cities that are linked by a nonstop airline connection. Such a graph will represent the relation "there is a nonstop flight from city (A) to city (B)." In this particular example, the binary relation is likely to be symmetric, i.e., when there is a nonstop flight from (A) to (B) there is also a nonstop flight from (B) to (A). In such situations, the graph is said to be undirected, as opposed to a general graph which is directed.

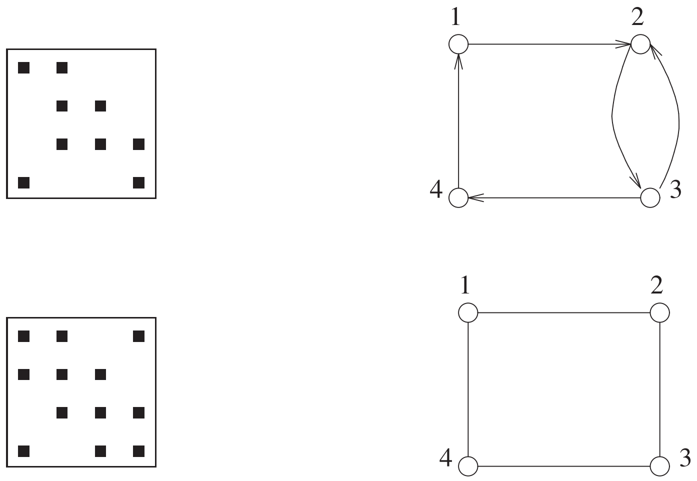


Figure 3.3 *Graphs of two* 4×4 *sparse matrices.*

Going back to sparse matrices, the *adjacency graph* of a sparse matrix is a graph $G = (V, E)$, whose n vertices in V represent the n unknowns. Its edges represent the binary relations established by the equations in the following manner: There is an edge from node i to node j when $a_{ij} \neq 0$. This edge will therefore represent the binary relation *equation* i *involves unknown* j. Note that the graph is directed, except when the matrix has a symmetric pattern ($a_{ij} \neq 0$ iff $a_{ji} \neq 0$ for all $1 \leq i, j \leq n$).

When a matrix has a symmetric nonzero pattern, i.e., when a_{ij} and a_{ji} are always nonzero at the same time, then the graph is *undirected*. Thus, for undirected graphs, every edge points in both directions. As a result, undirected graphs can be represented with nonoriented edges.

As an example of the use of graph models, parallelism in Gaussian elimination can be extracted by finding unknowns that are independent at a given stage of the elimination. These are unknowns which do not depend on each other according to the above binary relation. The rows corresponding to such unknowns can then be used as pivots simultaneously. Thus, in one extreme, when the matrix is diagonal, then all unknowns are independent. Conversely, when a matrix is dense, each unknown will depend on all other unknowns. Sparse matrices lie somewhere between these two extremes.

There are a few interesting simple properties of adjacency graphs. The graph of A^2 can be interpreted as an n-vertex graph whose edges are the pairs (i, j) for which there exists at least one path of length exactly two from node i to node j in the original graph of A. Similarly, the graph of A^k consists of edges which represent the binary relation "there is at least one path of length k from node i to node j." For details, see Exercise 4.

3.2.2 GRAPHS OF PDE MATRICES

For Partial Differential Equations involving only one physical unknown per mesh point, the adjacency graph of the matrix arising from the discretization is often the graph represented by the mesh itself. However, it is common to have several unknowns per mesh point. For example, the equations modeling fluid flow may involve the two velocity components of the fluid (in two dimensions) as well as energy and momentum at each mesh point. In such situations, there are two choices when labeling the unknowns. They can be labeled contiguously at each mesh point. Thus, for the example just mentioned, we can label all four variables (two velocities followed by momentum and then pressure) at a given mesh point as $u(k)$, ..., $u(k{+}3)$. Alternatively, all unknowns associated with one type of variable can be labeled first (e.g., first velocity components), followed by those associated with the second type of variables (e.g., second velocity components), etc. In either case, it is clear that there is redundant information in the graph of the adjacency matrix. The *quotient* graph corresponding to the *physical mesh* can be used instead. This results in substantial savings in storage and computation. In the fluid flow example mentioned above, the storage can be reduced by a factor of almost 16 for the integer arrays needed to represent the graph. This is because the number of edges has been reduced by this much, while the number of vertices, which is usually much smaller, remains the same.

3.3 PERMUTATIONS AND REORDERINGS

Permuting the rows or the columns, or both the rows and columns, of a sparse matrix is a common operation. In fact, *reordering* rows and columns is one of the most important ingredients used in *parallel* implementations of both direct and iterative solution techniques. This section introduces the ideas related to these reordering techniques and their relations to the adjacency graphs of the matrices. Recall the notation introduced in Chapter 1 that the j-th column of a matrix is denoted by a_{*j} and the i-th row by a_{i*}.

3.3.1 BASIC CONCEPTS

We begin with a definition and new notation.

DEFINITION 3.1 *Let A be a matrix and $\pi = \{i_1, i_2, \ldots, i_n\}$ a permutation of the set $\{1, 2, \ldots, n\}$. Then the matrices*

$$A_{\pi,*} = \{a_{\pi(i),j}\}_{i=1,\ldots,n;j=1,\ldots,m},$$
$$A_{*,\pi} = \{a_{i,\pi(j)}\}_{i=1,\ldots,n;j=1,\ldots,m}$$

are called row π-permutation and column π-permutation of A, respectively.

It is well known that any permutation of the set $\{1, 2, \ldots, n\}$ results from at most n interchanges, i.e., elementary permutations in which only two entries have been interchanged. An *interchange matrix* is the identity matrix with two of its rows interchanged. Denote by X_{ij} such matrices, with i and j being the numbers of the interchanged rows. Note that in order to interchange rows i and j of a matrix A, we only need to premultiply it by the matrix X_{ij}. Let $\pi = \{i_1, i_2, \ldots, i_n\}$ be an arbitrary permutation. This permutation is the product of a sequence of n consecutive interchanges $\sigma(i_k, j_k), k = 1, \ldots, n$. Then the rows of a matrix can be permuted by interchanging rows i_1, j_1, then rows i_2, j_2 of the resulting matrix, etc., and finally by interchanging i_n, j_n of the resulting matrix. Each of these operations can be achieved by a premultiplication by X_{i_k,j_k}. The same observation can be made regarding the columns of a matrix: In order to interchange columns i and j of a matrix, postmultiply it by X_{ij}. The following proposition follows from these observations.

PROPOSITION 3.1 *Let π be a permutation resulting from the product of the interchanges $\sigma(i_k, j_k)$, $k = 1, \ldots, n$. Then,*

$$A_{\pi,*} = P_\pi A, \quad A_{*,\pi} = AQ_\pi,$$

where

$$P_\pi = X_{i_n,j_n} X_{i_{n-1},j_{n-1}} \ldots X_{i_1,j_1}, \tag{3.1}$$

$$Q_\pi = X_{i_1,j_1} X_{i_2,j_2} \ldots X_{i_n,j_n}. \tag{3.2}$$

Products of interchange matrices are called *permutation matrices*. Clearly, a permutation matrix is nothing but the identity matrix with its rows (or columns) permuted.

Observe that $X_{i,j}^2 = I$, i.e., the square of an interchange matrix is the identity, or equivalently, the inverse of an interchange matrix is equal to itself, a property which is intuitively clear. It is easy to see that the matrices (3.1) and (3.2) satisfy

$$P_\pi Q_\pi = X_{i_n,j_n} X_{i_{n-1},j_{n-1}} \ldots X_{i_1,j_1} \times X_{i_1,j_1} X_{i_2,j_2} \ldots X_{i_n,j_n} = I,$$

which shows that the two matrices Q_π and P_π are nonsingular and that they are the inverse of one another. In other words, permuting the rows and the columns of a matrix, *using the same permutation*, actually performs a similarity transformation. Another important consequence arises because the products involved in the definitions (3.1) and (3.2) of P_π and Q_π occur in reverse order. Since each of the elementary matrices X_{i_k,j_k} is symmetric, the matrix Q_π is the transpose of P_π. Therefore,

$$Q_\pi = P_\pi^T = P_\pi^{-1}.$$

Since the inverse of the matrix P_π is its own transpose, permutation matrices are unitary.

Another way of deriving the above relationships is to express the permutation matrices P_π and P_π^T in terms of the identity matrix, whose columns or rows are permuted. It can easily be seen (See Exercise 3) that

$$P_\pi = I_{\pi,*}, \quad P_\pi^T = I_{*,\pi}.$$

It is then possible to verify directly that

$$A_{\pi,*} = I_{\pi,*} A = P_\pi A, \quad A_{*,\pi} = AI_{*,\pi} = AP_\pi^T.$$

It is important to interpret permutation operations for the linear systems to be solved. When the rows of a matrix are permuted, the order in which the equations are written is changed. On the other hand, when the columns are permuted, the unknowns are in effect *relabeled*, or *reordered*.

Example 3.1 Consider, for example, the linear system $Ax = b$ where

$$A = \begin{pmatrix} a_{11} & 0 & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}$$

and $\pi = \{1, 3, 2, 4\}$, then the (column-) permuted linear system is

$$\begin{pmatrix} a_{11} & a_{13} & 0 & 0 \\ 0 & a_{23} & a_{22} & a_{24} \\ a_{31} & a_{33} & a_{32} & 0 \\ 0 & 0 & a_{42} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}.$$

Note that only the unknowns have been permuted, not the equations, and in particular, the right-hand side has not changed.

In the above example, only the columns of A have been permuted. Such one-sided permutations are not as common as two-sided permutations in sparse matrix techniques. In reality, this is often related to the fact that the diagonal elements in linear systems play a distinct and important role. For instance, diagonal elements are typically large in PDE applications and it may be desirable to preserve this important property in the permuted matrix. In order to do so, it is typical to apply the same permutation to both the columns and the rows of A. Such operations are called *symmetric permutations*, and if denoted by $A_{\pi,\pi}$, then the result of such symmetric permutations satisfies the relation

$$A_{\pi,\pi} = P_\pi^T A P_\pi.$$

The interpretation of the symmetric permutation is quite simple. The resulting matrix corresponds to renaming, or relabeling, or reordering the unknowns and then reordering the equations in the same manner.

Example 3.2 For the previous example, if the rows are permuted with the same permutation as the columns, the linear system obtained is

$$\begin{pmatrix} a_{11} & a_{13} & 0 & 0 \\ a_{31} & a_{33} & a_{32} & 0 \\ 0 & a_{23} & a_{22} & a_{24} \\ 0 & 0 & a_{42} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ b_2 \\ b_4 \end{pmatrix}.$$

Observe that the diagonal elements are now diagonal elements from the original matrix, placed in a different order on the main diagonal.

3.3.2 RELATIONS WITH THE ADJACENCY GRAPH

From the point of view of graph theory, another important interpretation of a symmetric permutation is that *it is equivalent to relabeling the vertices of the graph* without altering the edges. Indeed, let (i, j) be an edge in the adjacency graph of the original matrix A and let A' be the permuted matrix. Then $a'_{ij} = a_{\pi(i),\pi(j)}$ and a result (i, j) is an edge in the adjacency graph of the permuted matrix A', if and only if $(\pi(i), \pi(j))$ is an edge in the graph of the original matrix A. Thus, the graph of the permuted matrix has not changed; rather, the labeling of the vertices has. In contrast, nonsymmetric permutations do not preserve the graph. In fact, they can transform an indirected graph into a directed one. Symmetric permutations may have a tremendous impact on the structure of the matrix even though the general graph of the adjacency matrix is identical.

Example 3.3 Consider the matrix illustrated in Figure 3.4 together with its adjacency graph. Such matrices are sometimes called "arrow" matrices because of their shape, but it would probably be more accurate to term them "star" matrices because of the structure of their graphs.

If the equations are reordered using the permutation $9, 8, \ldots, 1$, the matrix and graph shown in Figure 3.5 are obtained. Although the difference between the two graphs may seem slight, the matrices have a completely different structure, which may have a significant impact on the algorithms. As an example, if Gaussian elimination is used on the reordered matrix, no fill-in will occur, i.e., the L and U parts of the LU factorization will have the same structure as the lower and upper parts of A, respectively. On the other hand, Gaussian elimination on the original matrix results in disastrous fill-ins. Specifically, the L and U parts of the LU factorization are now dense matrices after the first step of Gaussian elimination. With direct sparse matrix techniques, it is important to find permutations of the matrix that will have the effect of reducing fill-ins during the Gaussian elimination process.

To conclude this section, it should be mentioned that two-sided nonsymmetric permutations may also arise in practice. However, they are more common in the context of direct methods.

3.3.3 COMMON REORDERINGS

The type of reordering, or permutations, used in applications depends on whether a direct or an iterative method is being considered. The following is a sample of such reorderings which are more useful for iterative methods.

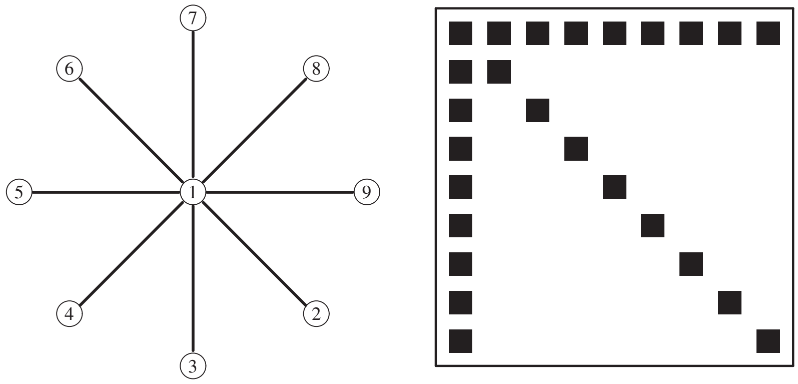


Figure 3.4 *Pattern of a 9 × 9 arrow matrix and its adjacency graph.*

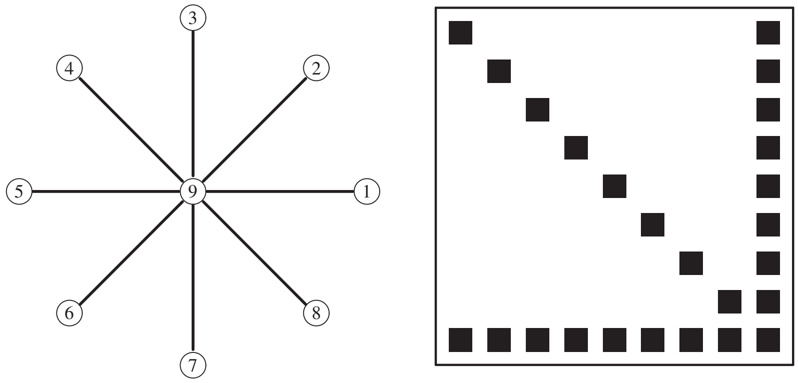


Figure 3.5 *Adjacency graph and matrix obtained from above figure after permuting the nodes in reverse order.*

Level-set orderings. This class of orderings contains a number of techniques that are based on traversing the graph by *level sets*. A level set is defined recursively as the set of all unmarked neighbors of all the nodes of a previous level set. Initially, a level set consists of one node, although strategies with several starting nodes are also important and will be considered later. As soon as a level set is traversed, its nodes are marked and numbered. They can, for example, be numbered in the order in which they are traversed. In addition, the order in which each level itself is traversed gives rise to different orderings. For instance, the nodes of a certain level can be visited in the natural order in which they are listed. The neighbors of each of these nodes are then inspected. Each time, a neighbor of a visited vertex that is not numbered is encountered, it is added to the list and labeled as the next element of the next level set. This simple strategy is called *Breadth First Search* (BFS) traversal in graph theory. The ordering will depend on the way in which the nodes are traversed in each level set. In BFS the elements of a level set are always traversed in the natural order in which they are listed. In the *Cuthill-McKee ordering* the elements of a level set are traversed from the nodes of lowest degree to those of highest degree.

ALGORITHM 3.1: Cuthill-McKee Ordering

1. *Input: initial node* i_1; *Output: permutation array* `iperm`.
2. *Start: Set* $levset := \{i_1\}$; $next = 2$;
3.   *Set* $marker(i_1) = 1$; `iperm` $(1) = i_1$
4. *While* $(next < n)$ *Do:*
5. $Next\_levset = \emptyset$
6.   *Traverse* $levset$ *in order of increasing degree and*
7.   *for each visited node Do:*
8.   *For each neighbor* i *of* j *such that* $marker(i) = 0$ *Do:*
9.   *Add* i *to the set* $Next\_levset$
10. $marker(i) := 1$; `iperm` $(next) = i$
11. $next = next + 1$
12.   *EndDo*
13.   *EndDo*
14. $levset := Next\_levset$
15. *EndWhile*

The `iperm` array obtained from the procedure lists the nodes in the order in which they are visited and can, in a practical implementation, be used to store the level sets in succession. A pointer is needed to indicate where each set starts. The array `iperm` thus constructed does in fact represent the permutation array π defined earlier.

In 1971, George [103] observed that *reversing* the Cuthill-McKee ordering yields a better scheme for sparse Gaussian elimination. The simplest way to understand this is to look at the two graphs produced by these orderings. The results of the standard and reversed Cuthill-McKee orderings on the sample finite element mesh problem seen earlier are shown in Figures 3.6 and 3.7, when the initial node is $i_1 = 3$ (relative to the labeling of the original ordering of Figure 2.10). The case of the figure, corresponds to a variant of CMK in which the traversals in Line 6, is done in a random order instead of according to the degree. A large part of the structure of the two matrices consists of little "arrow" submatrices, similar to the ones seen in Example 3.3. In the case of the regular CMK ordering, these arrows point upward, as in Figure 3.4, a consequence of the level set labeling. These blocks are similar the star matrices of Figure 3.4. As a result, Gaussian elimination will essentially fill in the square blocks which they span. As was indicated in Example 3.3, a remedy is to reorder the nodes backward, as is done globally in the reverse Cuthill-McKee strategy. For the reverse CMK ordering, the arrows are pointing downward, as in Figure 3.5, and Gaussian elimination yields much less fill-in.

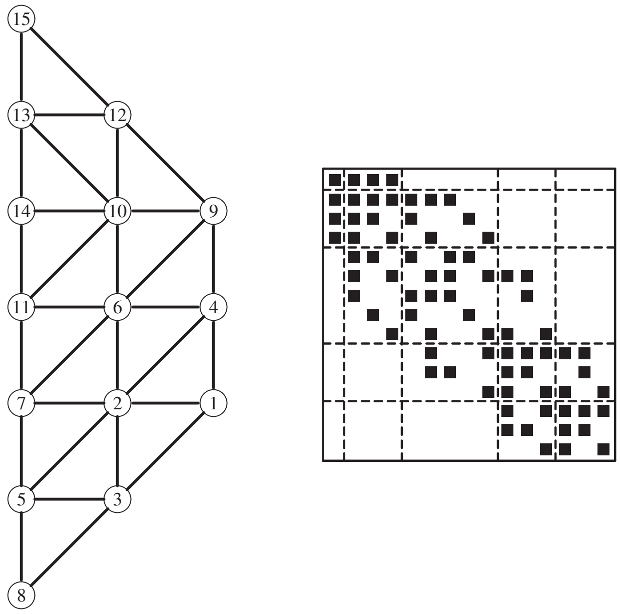


Figure 3.6 *Cuthill-McKee ordering.*

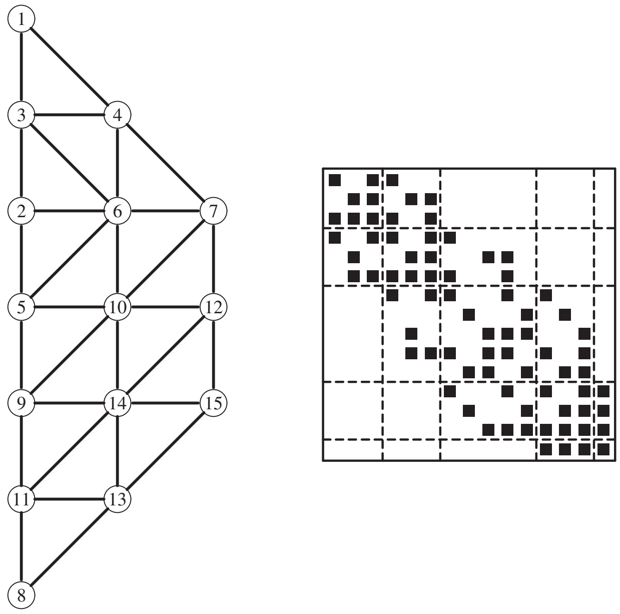


Figure 3.7 *Reverse Cuthill-McKee ordering.*

Example 3.4 The choice of the initial node in the CMK and RCMK orderings may be important. Referring to the original ordering of Figure 2.10, the previous illustration used $i_1 = 3$. However, it is clearly a poor choice if matrices with small bandwidth or *profile* are desired. If $i_1 = 1$ is selected instead, then the reverse Cuthill-McKee algorithm produces the matrix in Figure 3.8, which is more suitable for banded or *skyline* solvers.

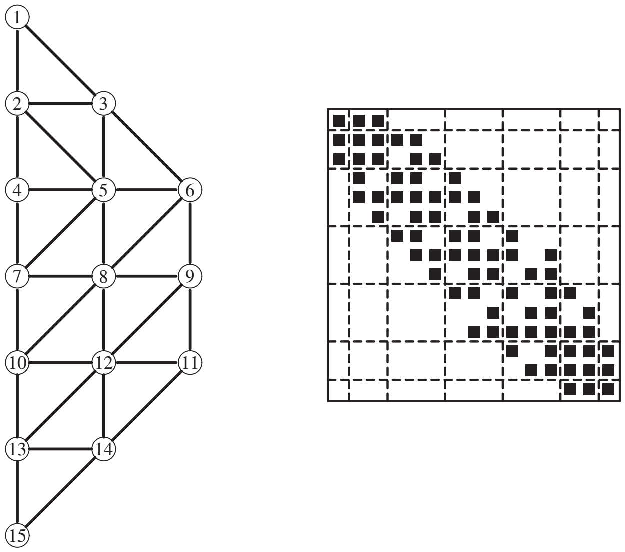


Figure 3.8 *Reverse Cuthill-McKee ordering starting with* $i_1 = 1$.

Independent set orderings. The matrices that arise in the model finite element problems seen in Figures 2.7, 2.10, and 3.2 are all characterized by an upper-left block that is diagonal, i.e., they have the structure

$$A = \begin{pmatrix} D & E \\ F & C \end{pmatrix}, \tag{3.3}$$

in which D is diagonal and C, E, and F are sparse matrices. The upper-diagonal block corresponds to unknowns from the previous levels of refinement and its presence is due to the ordering of the equations in use. As new vertices are created in the refined grid, they are given new numbers and the initial numbering of the vertices is unchanged. Since the old connected vertices are "cut" by new ones, they are no longer related by equations. Sets such as these are called *independent sets*. Independent sets are especially useful in parallel computing, for implementing both direct and iterative methods.

Referring to the adjacency graph $G = (V, E)$ of the matrix, and denoting by (x, y) the edge from vertex x to vertex y, an *independent set* S is a subset of the vertex set V such that

$$\text{if } x \in S, \quad \text{then} \quad \{(x, y) \in E \text{ or } (y, x) \in E\} \rightarrow y \notin S.$$

To explain this in words: Elements of S are not allowed to be connected to other elements of S either by incoming or outgoing edges. An independent set is *maximal* if it cannot be augmented by elements in its complement to form a larger independent set. Note that a maximal independent set is by no means the largest possible independent set that can be found. In fact, finding the independent set of maximum cardinal is NP-hard [132]. In the following, the term *independent set* always refers to *maximal independent set*.

There are a number of simple and inexpensive heuristics for finding large maximal independent sets. A greedy heuristic traverses the nodes in a given order, and if a node is not already marked, it selects the node as a new member of S. Then this node is marked along with its nearest neighbors. Here, a nearest neighbor of a node x means any node linked to x by an incoming or an outgoing edge.

ALGORITHM 3.2: Greedy Algorithm for ISO

1. *Set* $S = \emptyset$.
2. *For* $j = 1, 2, \ldots, n$ *Do:*
3. *If node* j *is not marked then*
4. $S = S \cup \{j\}$
5. *Mark* j *and all its nearest neighbors*
6. *EndIf*
7. *EndDo*

In the above algorithm, the nodes are traversed in the natural order $1, 2, \ldots, n$, but they can also be traversed in any permutation $\{i_1, \ldots, i_n\}$ of $\{1, 2, \ldots, n\}$. Since the size of the reduced system is $n - |S|$, it is reasonable to try to maximize the size of S in order to obtain a small reduced system. It is possible to give a rough idea of the size of S. Assume that the maximum degree of each node does not exceed ν. Whenever the above algorithm accepts a node as a new member of S, it potentially puts all its nearest neighbors, i.e., at most ν nodes, in the complement of S. Therefore, if s is the size of S, the size of its complement, $n - s$, is such that $n - s \leq \nu s$, and as a result,

$$s \geq \frac{n}{1+\nu}.$$

This lower bound can be improved slightly by replacing ν with the maximum degree ν_S of *all the vertices that constitute* S. This results in the inequality

$$s \geq \frac{n}{1+\nu_S},$$

which suggests that it may be a good idea to first visit the nodes with smaller degrees. In fact, this observation leads to a general heuristic regarding a good order of traversal. The algorithm can be viewed as follows: Each time a node is visited, remove it and its nearest neighbors from the graph, and then visit a node from the remaining graph. Continue in the same manner until all nodes are exhausted. Every node that is visited is a member of S and its nearest neighbors are members of $\bar{S}$. As result, if ν_i is the degree of the node visited at step i, adjusted for all the edge deletions resulting from the previous visitation steps, then the number n_i of nodes that are left at step i satisfies the relation

$$n_i = n_{i-1} - \nu_i - 1.$$

The process adds a new element to the set S at each step and stops when $n_i = 0$. In order to maximize $|S|$, the number of steps in the procedure must be maximized. The difficulty in the analysis arises from the fact that the degrees are updated at each step i because of the removal of the edges associated with the removed nodes. If the process is to be lengthened, a rule of thumb would be to visit the nodes that have the smallest degrees first.

ALGORITHM 3.3: Increasing Degree Traversal for ISO

1. *Set $S = \emptyset$. Find an ordering $i_1, \ldots, i_n$ of the nodes by increasing degree.*
2. *For $j = 1, 2, \ldots n$, Do:*
3. *If node i_j is not marked then*
4. $S = S \cup \{i_j\}$
5. *Mark i_j and all its nearest neighbors*
6. *EndIf*
7. *EndDo*

A refinement to the above algorithm would be to update the degrees of all nodes involved in a removal, and dynamically select the one with the smallest degree as the next node to be visited. This can be implemented efficiently using a min-heap data structure. A different heuristic is to attempt to maximize the number of elements in S by a form of local optimization which determines the order of traversal dynamically. In the following, removing a vertex from a graph means deleting the vertex and all edges incident to/from this vertex.

Example 3.5 The algorithms described in this section were tested on the same example used before, namely, the finite element mesh problem of Figure 2.10. Here, all strategies used yield the initial independent set in the matrix itself, which corresponds to the nodes of all the previous levels of refinement. This may well be optimal in this case, i.e., a larger independent set may not exist.

Multicolor orderings. Graph coloring is a familiar problem in computer science which refers to the process of labeling (coloring) the nodes of a graph in such a way that no two adjacent nodes have the same label (color). The goal of graph coloring is to obtain a colored graph which uses the smallest possible number of colors. However, optimality in the context of numerical linear algebra is a secondary issue and simple heuristics do provide adequate colorings.

Basic methods for obtaining a multicoloring of an arbitrary grid are quite simple. They rely on greedy techniques, a simple version of which is as follows.

ALGORITHM 3.4: Greedy Multicoloring Algorithm

1. *For $i = 1, \ldots, n$ Do: set $Color(i) = 0$.*
2. *For $i = 1, 2, \ldots, n$ Do:*
3. *Set Color*$(i) = \min \{k > 0 \mid k \neq$ *Color*$(j), \forall\, j \in \mathrm{Adj}(i))\}$
4. *EndDo*

Here, Adj(i) represents the set of nodes that are adjacent to node i. The color assigned to node i in line 3 is the smallest *allowable* color number which can be assigned to node i. Here, allowable means different from the colors of the nearest neighbors and positive. This procedure is illustrated in Figure 3.9. The node being colored in the figure is indicated by an arrow. It will be assigned color number 3, the smallest positive integer different from 1, 2, 4, 5.

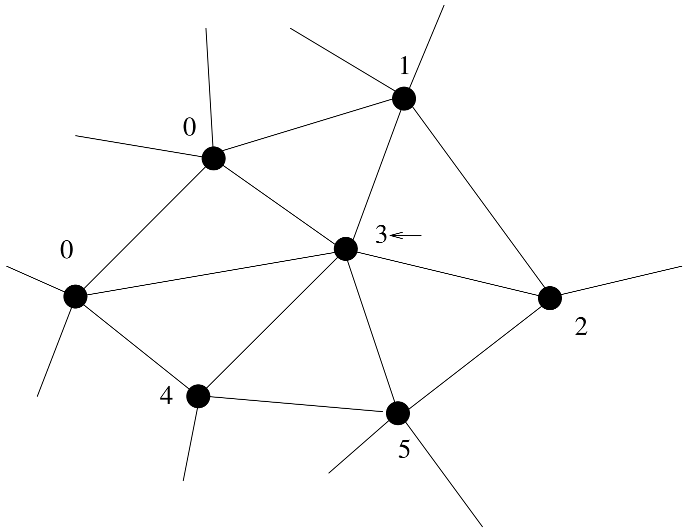


Figure 3.9 *The greedy multicoloring algorithm.*

In the above algorithm, the order $1, 2, \ldots, n$ has been arbitrarily selected for traversing the nodes and coloring them. Instead, the nodes can be traversed in any order $\{i_1, i_2, \ldots, i_n\}$. If a graph is *bipartite*, i.e., if it can be colored with two colors, then the algorithm will find the optimal two-color (Red-Black) ordering for *Breadth-First* traversals. In addition, if a graph is bipartite, it is easy to show that the algorithm will find two colors for any traversal which, at a given step, visits an unmarked node that is adjacent to at least one visited node. In general, the number of colors needed does not exceed the maximum degree of each node +1. These properties are the subject of Exercises 9 and 8.

Example 3.6 Figure 3.10 illustrates the algorithm for the same example used earlier, i.e., the finite element mesh problem of Figure 2.10. The dashed lines separate the different color sets found. Four colors are found in this example.

Once the colors have been found, the matrix can be permuted to have a block structure in which the diagonal blocks are diagonal. Alternatively, the color sets $S_j = [i_1^{(j)}, \ldots, i_{n_j}^{(j)}]$ and the permutation array in the algorithms can be used.

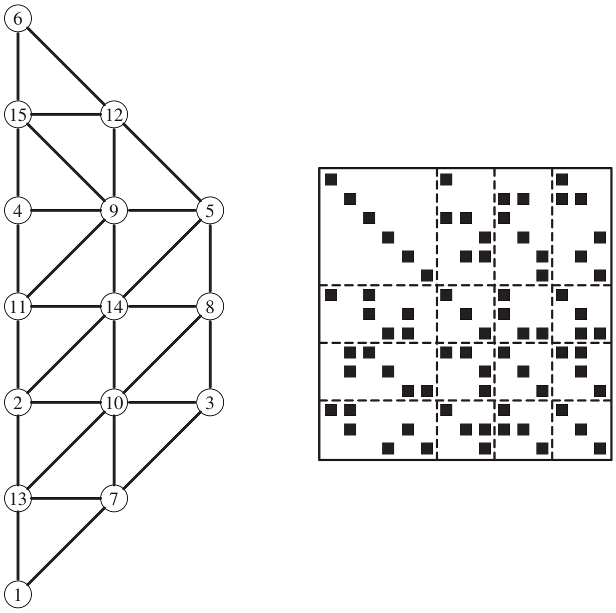


Figure 3.10 *Graph and matrix corresponding to mesh of Figure 2.10 after multicolor ordering.*

3.3.4 IRREDUCIBILITY

Remember that a *path* in a graph is a sequence of vertices $v_1, v_2, \ldots, v_k$, which are such that (v_i, v_{i+1}) is an edge for $i = 1, \ldots, k-1$. Also, a graph is said to be *connected* if there is a path between any pair of two vertices in V. A *connected component* in a graph is a *maximal subset* of vertices which all can be connected to one another by paths in the graph. Now consider matrices whose graphs may be *directed*. A matrix is *reducible* if its graph is not connected, and *irreducible* otherwise. When a matrix is reducible, then it can be permuted by means of *symmetric* permutations into a block upper triangular matrix of the form

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & \ldots \\ & A_{22} & A_{23} & \ldots \\ & & \ddots & \vdots \\ & & & A_{pp} \end{pmatrix},$$

where each partition corresponds to a connected component. It is clear that linear systems with the above matrix can be solved through a sequence of subsystems with the matrices $A_{ii}, i = p, p-1, \ldots, 1$.

3.4 STORAGE SCHEMES

In order to take advantage of the large number of zero elements, special schemes are required to store sparse matrices. The main goal is to represent only the nonzero elements, and to be able to perform the common matrix operations. In the following, Nz denotes the total number of nonzero elements. Only the most popular schemes are covered here, but additional details can be found in books such as Duff, Erisman, and Reid [77].

The simplest storage scheme for sparse matrices is the so-called coordinate format. The data structure consists of three arrays: (1) a real array containing all the real (or complex) values of the nonzero elements of A in any order; (2) an integer array containing their row indices; and (3) a second integer array containing their column indices. All three arrays are of length Nz, the number of nonzero elements.

Example 3.7 The matrix

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

will be represented (for example) by

AA	12.	9.	7.	5.	1.	2.	11.	3.	6.	4.	8.	10.
JR	5	3	3	2	1	1	4	2	3	2	3	4
JC	5	5	3	4	1	4	4	1	1	2	4	3

In the above example, the elements are listed in an arbitrary order. In fact, they are usually listed by row or columns. If the elements were listed by row, the array JC which contains redundant information might be replaced by an array which points to the beginning of each row instead. This would involve nonnegligible savings in storage. The new data structure has three arrays with the following functions:

- A real array AA contains the real values a_{ij} stored row by row, from row 1 to n. The length of AA is Nz.
- An integer array JA contains the column indices of the elements a_{ij} as stored in the array AA. The length of JA is Nz.
- An integer array IA contains the pointers to the beginning of each row in the arrays AA and JA. Thus, the content of $IA(i)$ is the position in arrays AA and JA where the i-th row starts. The length of IA is $n+1$ with $IA(n+1)$ containing the number $IA(1) + Nz$, i.e., the address in A and JA of the beginning of a fictitious row number $n+1$.

Thus, the above matrix may be stored as follows:

AA	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
JA	1	4	1	2	4	1	3	4	5	3	4	5
IA	1	3	6	10	12	13						

This format is probably the most popular for storing general sparse matrices. It is called the *Compressed Sparse Row* (CSR) format. This scheme is preferred over the coordinate scheme because it is often more useful for performing typical computations. On the other hand, the coordinate scheme is advantageous for its simplicity and its flexibility. It is often used as an "entry" format in sparse matrix software packages.

There are a number of variations for the Compressed Sparse Row format. The most obvious variation is storing the columns instead of the rows. The corresponding scheme is known as the *Compressed Sparse Column* (CSC) scheme.

Another common variation exploits the fact that the diagonal elements of many matrices are all usually nonzero and/or that they are accessed more often than the rest of the elements. As a result, they can be stored separately. The *Modified Sparse Row* (MSR) format has only two arrays: a real array AA and an integer array JA. The first n positions in AA contain the diagonal elements of the matrix in order. The position $n+1$ of the array AA is not used, but may sometimes be used to carry other information concerning the matrix. Starting at position $n+2$, the nonzero elements of AA, excluding its diagonal elements, are stored by row. For each element $AA(k)$, the integer $JA(k)$ represents its column index on the matrix. The $n+1$ first positions of JA contain the pointer to the beginning of each row in AA and JA. Thus, for the above example, the two arrays will be as follows:

AA	1.	4.	7.	11.	12.	*	2.	3.	5.	6.	8.	9.	10.
JA	7	8	10	13	14	14	4	1	4	1	4	5	3

The star denotes an unused location. Notice that $JA(n) = JA(n+1) = 14$, indicating that the last row is a zero row, once the diagonal element has been removed.

Diagonally structured matrices are matrices whose nonzero elements are located along a small number of diagonals. These diagonals can be stored in a rectangular array `DIAG(1:n,1:Nd)`, where `Nd` is the number of diagonals. The offsets of each of the diagonals with respect to the main diagonal must be known. These will be stored in an array `IOFF(1:Nd)`. Thus, the element $a_{i,i+\text{ioff}(j)}$ of the original matrix is located in position (i,j) of the array `DIAG`, i.e.,

$$\texttt{DIAG}(i,j) \leftarrow a_{i,i+\text{ioff}(j)}.$$

The order in which the diagonals are stored in the columns of `DIAG` is generally unimportant, though if several more operations are performed with the main diagonal, storing it in the first column may be slightly advantageous. Note also that all the diagonals except the main diagonal have fewer than n elements, so there are positions in `DIAG` that will not be used.

Example 3.8 For example, the following matrix which has three diagonals

$$A = \begin{pmatrix} 1. & 0. & 2. & 0. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 0. & 6. & 7. & 0. & 8. \\ 0. & 0. & 9. & 10. & 0. \\ 0. & 0. & 0. & 11. & 12. \end{pmatrix}$$

will be represented by the two arrays

$$\mathtt{DIAG} = \begin{array}{|ccc|} \hline * & 1. & 2. \\ 3. & 4. & 5. \\ 6. & 7. & 8. \\ 9. & 10. & * \\ 11 & 12. & * \\ \hline \end{array} \qquad \mathtt{IOFF} = \boxed{\text{-1 0 2}}\,.$$

A more general scheme which is popular on vector machines is the so-called Ellpack-Itpack format. The assumption in this scheme is that there are at most Nd nonzero elements per row, where `Nd` is small. Then two rectangular arrays of dimension `n` $\times$ `Nd` each are required (one real and one integer). The first, `COEF`, is similar to `DIAG` and contains the nonzero elements of A. The nonzero elements of each row of the matrix can be stored in a row of the array `COEF(1:n,1:Nd)`, completing the row by zeros as necessary. Together with `COEF`, an integer array `JCOEF(1:n,1:Nd)` must be stored which contains the column positions of each entry in `COEF`.

Example 3.9 Thus, for the matrix of the previous example, the Ellpack-Itpack storage scheme is

$$\mathtt{COEF} = \begin{array}{|ccc|} \hline 1. & 2. & 0. \\ 3. & 4. & 5. \\ 6. & 7. & 8. \\ 9. & 10. & 0. \\ 11 & 12. & 0. \\ \hline \end{array} \qquad \mathtt{JCOEF} = \begin{array}{|ccc|} \hline 1 & 3 & 1 \\ 1 & 2 & 4 \\ 2 & 3 & 5 \\ 3 & 4 & 4 \\ 4 & 5 & 5 \\ \hline \end{array}\,.$$

A certain column number must be chosen for each of the zero elements that must be added to pad the shorter rows of A, i.e., rows 1, 4, and 5. In this example, those integers are selected to be equal to the row numbers, as can be seen in the `JCOEF` array. This is somewhat arbitrary, and in fact, any integer between 1 and n would be acceptable. However, there may be good reasons for not inserting the same integers too often, e.g. a constant number, for performance considerations.

3.5 BASIC SPARSE MATRIX OPERATIONS

The matrix-by-vector product is an important operation which is required in most of the iterative solution algorithms for solving sparse linear systems. This section shows how these can be implemented for a small subset of the storage schemes considered earlier.

The following Fortran 90 segment shows the main loop of the matrix-by-vector operation for matrices stored in the Compressed Sparse Row stored format.

```
DO I=1, N
   K1 = IA(I)
   K2 = IA(I+1)-1
   Y(I) = DOTPRODUCT(A(K1:K2),X(JA(K1:K2)))
ENDDO
```

Notice that each iteration of the loop computes a different component of the resulting vector. This is advantageous because each of these components can be computed independently. If the matrix is stored by columns, then the following code could be used instead:

```
DO J=1, N
   K1 = IA(J)
   K2 = IA(J+1)-1
   Y(JA(K1:K2)) = Y(JA(K1:K2))+X(J)*A(K1:K2)
ENDDO
```

In each iteration of the loop, a multiple of the j-th column is added to the result, which is assumed to have been initially set to zero. Notice now that the outer loop is no longer parallelizable. An alternative to improve parallelization is to try to split the vector operation in each inner loop. The inner loop has few operations, in general, so this is unlikely to be a sound approach. This comparison demonstrates that data structures may have to change to improve performance when dealing with high performance computers.

Now consider the matrix-by-vector product in diagonal storage.

```
DO J=1, N
   JOFF = IOFF(J)
   DO I=1, N
      Y(I) = Y(I) +DIAG(I,J)*X(JOFF+I)
   ENDDO
ENDDO
```

Here, each of the diagonals is multiplied by the vector x and the result added to the vector y. It is again assumed that the vector y has been filled with zeros at the start of the loop. From the point of view of parallelization and/or vectorization, the above code is probably the better to use. On the other hand, it is not general enough.

Solving a lower or upper triangular system is another important “kernel” in sparse matrix computations. The following segment of code shows a simple routine for solving a unit lower triangular system $Lx = y$ for the CSR storage format.

```
X(1) = Y(1)
DO I = 2, N
   K1 = IAL(I)
   K2 = IAL(I+1)-1
   X(I)=Y(I)-DOTPRODUCT(AL(K1:K2),X(JAL(K1:K2)))
ENDDO
```

At each step, the inner product of the current solution x with the i-th row is computed and subtracted from $y(i)$. This gives the value of $x(i)$. The `dotproduct` function computes the dot product of two arbitrary vectors `u(k1:k2)` and `v(k1:k2)`. The vector `AL(K1:K2)` is the i-th row of the matrix L in sparse format and `X(JAL(K1:K2))` is the vector of the components of `X` *gathered* into a short vector which is consistent with the column indices of the elements in the row `AL(K1:K2)`.

3.6 SPARSE DIRECT SOLUTION METHODS

Most direct methods for sparse linear systems perform an LU factorization of the original matrix and try to reduce cost by minimizing fill-ins, i.e., nonzero elements introduced during the elimination process in positions which were initially zeros. The data structures employed are rather complicated. The early codes relied heavily on *linked lists* which are convenient for inserting new nonzero elements. Linked-list data structures were dropped in favor of other more dynamic schemes that leave some initial elbow room in each row for the insertions, and then adjust the structure as more fill-ins are introduced.

A typical sparse direct solution solver for positive definite matrices consists of four phases. First, preordering is applied to minimizing fill-in. Two popular methods are used: minimal degree ordering and nested-dissection ordering. Second, a symbolic factorization is performed. This means that the factorization is processed only symbolically, i.e., without numerical values. Third, the numerical factorization, in which the actual factors L and U are formed, is processed. Finally, the forward and backward triangular sweeps are executed for each different right-hand side. In a code where numerical pivoting is necessary, the symbolic phase cannot be separated from the numerical factorization.

3.7 TEST PROBLEMS

For comparison purposes it is important to use a common set of test matrices that represent a wide spectrum of applications. There are two distinct ways of providing such data sets. The first approach is to collect sparse matrices in a well-specified standard format from various applications. This approach is used in the Harwell-Boeing collection of test matrices. The second approach is to generate these matrices with a few sample programs such as those provided in the SPARSKIT library [179]. The coming chapters will use examples from these two sources. In particular, five test problems will be emphasized for their varying degrees of difficulty.

The SPARSKIT package can generate matrices arising from the discretization of the two- or three-dimensional Partial Differential Equations

$$-\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(b\frac{\partial u}{\partial y}\right)-\frac{\partial}{\partial z}\left(c\frac{\partial u}{\partial z}\right)$$
$$+\frac{\partial\,(du)}{\partial x}+\frac{\partial\,(eu)}{\partial y}+\frac{\partial\,(fu)}{\partial z}+gu=h$$

on rectangular regions with general mixed-type boundary conditions. In the test problems, the regions are the square $\Omega=(0,1)^2$, or the cube $\Omega=(0,1)^3$; the Dirichlet condition $u=0$ is always used on the boundary. Only the discretized matrix is of importance, since the right-hand side will be created artificially. Therefore, the right-hand side, h, is not relevant.

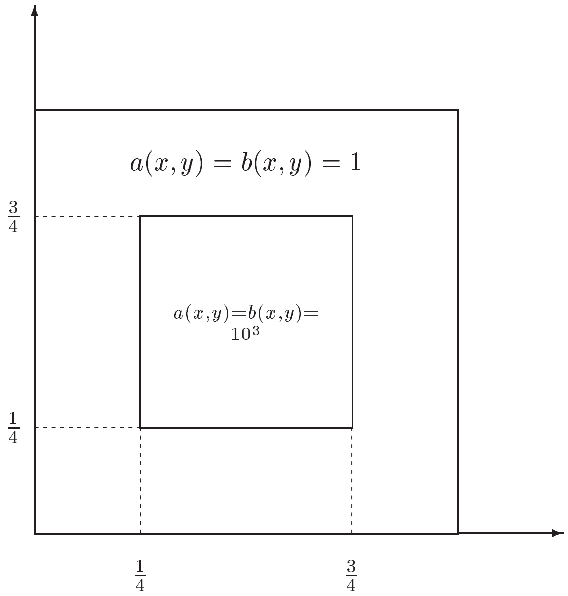


Figure 3.11 *Physical domain and coefficients for Problem 1.*

Problem 1: F2DA. In the first test problem which will be labeled F2DA, the domain is two-dimensional, with

$$a(x,y)=b(x,y)=1.0$$

and

$$d(x,y)=\gamma(x+y),\quad e(x,y)=\gamma(x-y),\quad f(x,y)=g(x,y)=0.0, \tag{3.4}$$

where the constant γ is equal to 10. The domain and coefficients for this problem are shown is Figure 3.11. If the number of points in each direction is 34, then there are $n_x=n_y=32$

interior points in each direction and a matrix of size $n = n_x \times n_y = 32^2 = 1024$ is obtained. In this test example, as well as the other ones described below, the right-hand side is generated as

$$b = Ae,$$

in which $e = (1, 1, \ldots, 1)^T$. The initial guess is always taken to be a vector of pseudo-random values.

Problem 2: F2DB. The second test problem is similar to the previous one but involves discontinuous coefficient functions a and b. Here, $n_x = n_y = 32$ and the functions d, e, f, g are also defined by (3.4). However, the functions a and b now both take the value 1,000 inside the subsquare of width $\frac{1}{2}$ centered at $(\frac{1}{2}, \frac{1}{2})$, and one elsewhere in the domain, i.e.,

$$a(x, y) = b(x, y) = \begin{cases} 10^3 & \text{if} \quad \frac{1}{4} < x, y < \frac{3}{4} \\ 1 & \text{otherwise} \end{cases} \quad .$$

Problem 3: F3D. The third test problem is three-dimensional with $n_x = n_y = n_z = 16$ internal mesh points in each direction leading to a problem of size $n = 4096$. In this case, we take

$$a(x, y, z) = b(x, y, z) = c(x, y, z) = 1$$

$$d(x, y, z) = \gamma e^{xy}, \quad e(x, y, z) = \gamma e^{-xy},$$

and

$$f(x, y, z) = g(x, y, z) = 0.0.$$

The constant γ is taken to be equal to 10.0 as before.

The Harwell-Boeing collection is a large data set consisting of test matrices which have been contributed by researchers and engineers from many different disciplines. These have often been used for test purposes in the literature [78]. The collection provides a data structure which constitutes an excellent medium for exchanging matrices. The matrices are stored as ASCII files with a very specific format consisting of a four- or five-line header. Then, the data containing the matrix is stored in CSC format together with any right-hand sides, initial guesses, or exact solutions when available. The SPARSKIT library also provides routines for reading and generating matrices in this format.

Only one matrix from the collection was selected for testing the algorithms described in the coming chapters. The matrices in the last two test examples are both irregularly structured.

Problem 4: ORS The matrix selected from the Harwell-Boeing collection is ORSIRR1. This matrix arises from a reservoir engineering problem. Its size is $n = 1030$ and it has a total of Nz =6,858 nonzero elements. The original problem is based on a $21 \times 21 \times 5$ irregular grid. In this case and the next one, the matrices are preprocessed by scaling their rows and columns.

Problem 5: FID This test matrix is extracted from the well known fluid flow simulation package FIDAP [84]. It is actually the test example number 36 from this package and features a two-dimensional Chemical Vapor Deposition in a Horizontal Reactor. The matrix has a size of $n = 3079$ and has $Nz = 53843$ nonzero elements. It has a symmetric pattern and few diagonally dominant rows or columns. The rows and columns are prescaled in the same way as in the previous example. Figure 3.12 shows the patterns of the matrices ORS and FID.

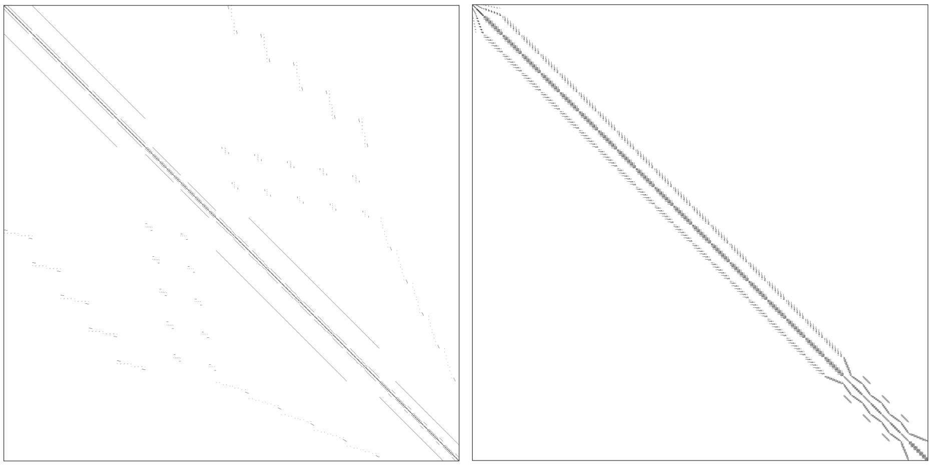


Figure 3.12 *Patterns of the matrices ORS (left) and FID (right).*

EXERCISES

1. Consider the mesh of a discretized PDE. In which situations is the graph representing this mesh the same as the adjacency graph of the matrix? Give examples from both Finite Difference and Finite Element discretizations.
2. Let A and B be two sparse (square) matrices of the same dimension. How can the graph of $C = A + B$ be characterized with respect to the graphs of A and B?
3. Consider the matrix defined as
$$P_\pi = I_{\pi,*}.$$
Show directly (without using Proposition 3.1 or interchange matrices) that the following three relations hold
$$\begin{aligned} A_{\pi,*} &= I_{\pi,*}A \\ I_{*,\pi} &= P_\pi^T \\ AP_\pi^T &= A_{*,\pi}. \end{aligned}$$

4. Consider the two matrices

$$A = \begin{pmatrix} \star & \star & 0 & \star & 0 & 0 \\ 0 & \star & 0 & 0 & 0 & \star \\ 0 & \star & \star & 0 & 0 & 0 \\ 0 & \star & 0 & 0 & \star & 0 \\ 0 & 0 & 0 & 0 & \star & 0 \\ 0 & 0 & 0 & 0 & 0 & \star \end{pmatrix} \quad B = \begin{pmatrix} \star & 0 & 0 & 0 & 0 & 0 \\ \star & 0 & \star & 0 & \star & 0 \\ 0 & \star & 0 & 0 & 0 & 0 \\ \star & \star & 0 & 0 & 0 & 0 \\ 0 & \star & 0 & \star & \star & 0 \\ 0 & 0 & \star & 0 & 0 & \star \end{pmatrix}$$

where a $\star$ represents an arbitrary nonzero element.

a. Show the adjacency graphs of the matrices A, B, AB, and BA. (Assume that there are no numerical cancellations in computing the products AB and BA). Since there are zero diagonal elements, represent explicitly the cycles corresponding to the (i,i) edges when they are present.

b. Consider the matrix $C = AB$. Give an interpretation of an edge in the graph of C in terms of edges in the graph of A and B. Verify this answer using the above matrices.

c. Consider the particular case in which $B = A$. Give an interpretation of an edge in the graph of C in terms of paths of length two in the graph of A. The paths must take into account the cycles corresponding to nonzero diagonal elements of A.

d. Now consider the case where $B = A^2$. Give an interpretation of an edge in the graph of $C = A^3$ in terms of paths of length three in the graph of A. Generalize the result to arbitrary powers of A.

5. Consider a 6×6 matrix which has the pattern

$$A = \begin{pmatrix} \star & \star & & & \star & \\ \star & \star & \star & & & \star \\ & \star & \star & & & \\ & & & \star & \star & \\ \star & & & \star & \star & \star \\ & \star & & & \star & \star \end{pmatrix}.$$

a. Show the adjacency graph of A.

b. Consider the permutation $\pi = \{1, 3, 4, 2, 5, 6\}$. Show the adjacency graph and new pattern for the matrix obtained from a symmetric permutation of A based on the permutation array π.

6. Consider a matrix which has the pattern

$$A = \begin{pmatrix} \star & \star & & & \star & & & \star \\ \star & \star & \star & & & \star & & \\ & \star & \star & \star & & & \star & \\ & & \star & \star & \star & & & \star \\ \star & & & \star & \star & \star & & \\ & \star & & & \star & \star & \star & \\ & & \star & & & \star & \star & \star \\ \star & & & \star & & & \star & \star \end{pmatrix}.$$

a. Show the adjacency graph of A. (Place the 8 vertices on a circle.)

b. Consider the permutation $\pi = \{1, 3, 5, 7, 2, 4, 6, 8\}$. Show the adjacency graph and new pattern for the matrix obtained from a symmetric permutation of A based on the permutation array π.

c. Show the adjacency graph and new pattern for the matrix obtained from a reverse Cuthill-McKee ordering of A starting with the node 1. (Assume the vertices adjacent to a given vertex are always listed in increasing order in the data structure that describes the graph.)

d. Find a multicolor ordering for A (give the vertex labels color 1, followed by those for color 2, etc.).

7. Given a five-point finite difference graph, show that the greedy algorithm will always find a coloring of the graph with two colors.

8. Prove that the total number of colors found by the greedy multicoloring algorithm does not exceed $\nu_{max} + 1$, where ν_{max} is the maximum degree of all the vertices of a graph (not counting the cycles (i, i) associated with diagonal elements).

9. Consider a graph that is bipartite, i.e., 2-colorable. Assume that the vertices of the graph are colored by a variant of Algorithm (3.4), in which the nodes are traversed in a certain order $i_1, i_2, \ldots, i_n$.

a. Is it true that for any permutation $i_1, \ldots, i_n$ the number of colors found will be two?

b. Consider now a permutation satisfying the following property: for each j at least one of the nodes $i_1, i_2, \ldots, i_{j-1}$ is adjacent to i_j. Show that the algorithm will find a 2-coloring of the graph.

c. Among the following traversals indicate which ones satisfy the property of the previous question: (1) Breadth-First Search, (2) random traversal, (3) traversal defined by i_j = any node adjacent to i_{j-1}.

10. Given a matrix that is irreducible and with a symmetric pattern, show that its structural inverse is dense. Structural inverse means the pattern of the inverse, regardless of the values, or otherwise stated, is the union of all patterns of the inverses for all possible values. [Hint: Use Cayley Hamilton's theorem and a well known result on powers of adjacency matrices mentioned at the end of Section 3.2.1.]

11. The most economical storage scheme in terms of memory usage is the following variation on the coordinate format: Store all nonzero values a_{ij} in a real array $AA[1 : Nz]$ and the corresponding "linear array address" $(i - 1) * n + j$ in an integer array $JA[1 : Nz]$. The order in which these corresponding entries are stored is unimportant as long as they are both in the same position in their respective arrays. What are the advantages and disadvantages of this data structure? Write a short routine for performing a matrix-by-vector product in this format.

12. Write a FORTRAN code segment to perform the matrix-by-vector product for matrices stored in Ellpack-Itpack format.

13. Write a small subroutine to perform the following operations on a sparse matrix in coordinate format, diagonal format, and CSR format:

a. Count the number of nonzero elements in the main diagonal;

b. Extract the diagonal whose offset is k;

c. Add a nonzero element in position (i, j) of the matrix (this position may initially contain a zero or a nonzero element);

d. Add a given diagonal to the matrix. What is the most convenient storage scheme for each of these operations?

14. Linked lists is another popular scheme often used for storing sparse matrices. These allow to link together k data items (e.g., elements of a given row) in a large linear array. A starting position is given in the array which contains the first element of the set. Then, a link to the next element in the array is provided from a LINK array.

a. Show how to implement this scheme. A linked list is to be used for each row.

b. What are the main advantages and disadvantages of linked lists?

c. Write an algorithm to perform a matrix-by-vector product in this format.

NOTES AND REFERENCES. Two good references on sparse matrix computations are the book by George and Liu [104] and the more recent volume by Duff, Erisman, and Reid [77]. These are geared toward direct solution methods and the first specializes in symmetric positive definite problems. Also of interest are [157] and [163] and the early survey by Duff [76].

Sparse matrix techniques have traditionally been associated with direct solution methods. Clearly, this is now changing rapidly since the sophistication of iterative solution packages is starting to equal that of direct solvers. The SPARSKIT library, a package for sparse matrix computations [179] is currently in its second version and is available through anonymous FTP (`http://www.cs.umn.edu/Research/arpa/SPARSKIT`). Another available software package which emphasizes object-oriented design with the goal of hiding complex data structures from users is PETSc [19]. A manipulation package for sparse matrices, similar to SPARSKIT in spirit, is SMMS developed by Alvarado [6].

The idea of the greedy multicoloring algorithm is known in Finite Element techniques (to color elements); see, e.g., Benantar and Flaherty [23]. Wu [229] presents the greedy algorithm for multicoloring vertices and uses it for SOR type iterations, see also [182]. The effect of multicoloring has been extensively studied by Adams [2, 3] and Poole and Ortega [164]. Interesting results regarding multicoloring in the context of finite elements based on quad-tree structures have been obtained by Benantar and Flaherty [23] who show, in particular, that with this structure a maximum of six colors is required. ■