

SPARSE MATRIX TECHNOLOGY

electronic edition

Sergio Pissanetzky

Sparse Matrix Technology

electronic edition

Sergio Pissanetzky

Copyright © 2007 by Sergio Pissanetzky and SciControls.com. All rights reserved. No part of the contents of this book can be reproduced without the written permission of the publisher.

Professionally typeset by L^AT_EX. This work is in compliance with the mathematical typesetting conventions established by the [International Organization for Standardization](#) (ISO).

Dr. Pissanetzky retired after a rewarding career as an Entrepreneur, Professor, Research Scientist and Consultant. He was the founder of Magnus Software Corporation, where he focused on development of specialized applications for the Magnetic Resonance Imaging (MRI) and the High Energy Particle Accelerator industries. He has served as Member of the International Editorial Board of the “International Journal for Computation in Electrical and Electronic Engineering”, as a Member of the International Advisory Committee of the International Journal “Métodos Numéricos para Cálculo y Diseño en Ingeniería”, and as a member of the International Committee for Nuclear Resonance Spectroscopy, Tokyo, Japan. Dr. Pissanetzky has held professorships in Physics at Texas A&M University and the Universities of Buenos Aires, Córdoba and Cuyo, Argentina. He has also held positions as a Research Scientist with the Houston Advanced Research Center, as Chairman of the Computer Center of the Atomic Energy Commission, San Carlos de Bariloche, Argentina, and as a Scientific Consultant at Brookhaven National Laboratory. Dr. Pissanetzky holds several US and European patents and is the author of two books and numerous peer reviewed technical papers. Dr. Pissanetzky earned his Ph.D. in Physics at the Balseiro Institute, University of Cuyo, in 1965. Dr. Pissanetzky has 35 years of teaching experience and 30 years of programming experience in languages such as Fortran, Basic, C and C++. Dr. Pissanetzky now lives in a quite suburban neighborhood in Texas.

Website: <http://www.SciControls.com>

ISBN 978-0-9762775-3-8

Contents

Preface	xiii
Introduction	1
1 Fundamentals	5
1.1 Introduction	5
1.2 Storage of arrays, lists, stacks and queues	5
1.3 Storage of lists of integers	8
1.4 Representation and storage of graphs	10
1.5 Diagonal storage of band matrices	12
1.6 Envelope storage of symmetric matrices	14
1.7 Linked sparse storage schemes	15
1.8 The sparse row-wise format	19
1.9 Ordered and unordered representations	20
1.10 Sherman's compression	22
1.11 Storage of block-partitioned matrices	24
1.12 Symbolic processing and dynamic storage schemes	26
1.13 Merging sparse lists of integers	28
1.14 The multiple switch technique	29
1.15 Addition of sparse vectors with the help of an expanded real accumulator	30
1.16 Addition of sparse vectors with the help of an expanded integer array of pointers	32
1.17 Scalar product of two sparse vectors with the help of an array of pointers	33
2 Linear Algebraic Equations	35
2.1 Introduction	35
2.2 Some definitions and properties	37
2.3 Elementary matrices and triangular matrices	40
2.4 Some properties of elementary matrices	41
2.5 Some properties of triangular matrices	42
2.6 Permutation matrices	44

2.7	Gauss elimination by columns	45
2.8	Gauss elimination by rows	49
2.9	Gauss-Jordan elimination	50
2.10	Relation between the elimination form of the inverse and the product form of the inverse	52
2.11	Cholesky factorization of a symmetric positive definite matrix	53
2.12	Practical implementation of Cholesky factorization	55
2.13	Forward and backward substitution	56
2.14	Cost considerations	57
2.15	Numerical examples	59
3	Numerical Errors in Gauss Elimination	63
3.1	Introduction	63
3.2	Numerical errors in floating point operations	65
3.3	Numerical errors in sparse factorization	68
3.4	Numerical errors in sparse substitution	73
3.5	The control of numerical errors	77
3.6	Numerical stability and pivot selection	78
3.7	Monitoring or estimating element growth	82
3.8	Scaling	83
4	Ordering for Gauss Elimination: Symmetric Matrices	85
4.1	Introduction: Statement of the problem	85
4.2	Basic notions of graph theory	87
4.3	Breadth-first search and adjacency level structures	93
4.4	Finding a pseudoperipheral vertex and a narrow level structure of a graph	95
4.5	Reducing the bandwidth of a symmetric matrix	96
4.6	Reducing the profile of a symmetric matrix	98
4.7	Graph-theoretical background of symmetric Gauss elimination	101
4.8	The minimum degree algorithm	104
4.9	Tree partitioning of a symmetric sparse matrix	109
4.10	Nested dissection	113
4.11	Properties of nested dissection orderings	118
4.12	Generalized nested dissection	121
4.13	One-way dissection of finite element problems	122
4.14	Orderings for the finite element method	127
4.15	Depth-first search of an undirected graph	132
4.16	Lexicographic search	136
4.17	Symmetric indefinite matrices	140

5	Ordering for Gauss Elimination: General Matrices	143
5.1	Introduction: Statement of the problem	143
5.2	Graph theory for unsymmetric matrices	146
5.3	The strong components of a digraph	148
5.4	Depth-first search of a digraph	151
5.5	Breadth-first search of a digraph and directed adjacency level structures	155
5.6	Finding a maximal set of vertex disjoint paths in an acyclic digraph	157
5.7	Finding a transversal: the algorithm of Hall	158
5.8	Finding a transversal: the algorithm of Hopcroft and Karp	161
5.9	The algorithm of Sargent and Westerberg for finding the strong components of a digraph	167
5.10	The algorithm of Tarjan for finding the strong components of a digraph	168
5.11	Pivoting strategies for unsymmetric matrices	172
5.12	Other methods and available software	175
6	Sparse Eigenanalysis	177
6.1	Introduction	177
6.2	The Rayleigh quotient	180
6.3	Bounds for eigenvalues	182
6.4	The bisection method for eigenvalue calculations	184
6.5	Reduction of a general matrix	185
6.6	Reduction of a symmetric band matrix to tridiagonal form	188
6.7	Eigenanalysis of tridiagonal and Hessenberg matrices	189
6.8	Direct and inverse iteration	190
6.9	Subspaces and invariant subspaces	193
6.10	Simultaneous iteration	196
6.11	Lanczos algorithm	199
6.12	Lanczos algorithm in practice	203
6.13	Block Lanczos and band Lanczos algorithms	206
6.14	Trace minimization	208
6.15	Eigenanalysis of hermitian matrices	209
6.16	Unsymmetric eigenproblems	210
7	Sparse Matrix Algebra	211
7.1	Introduction	211
7.2	Transposition of a sparse matrix	213
7.3	Algorithm for the transposition of a general sparse matrix	215
7.4	Ordering a sparse representation	216
7.5	Permutation of rows or columns of a sparse matrix: First procedure	217

7.6	Permutation of rows or columns of a sparse matrix: Second procedure	218
7.7	Ordering of the upper representation of a sparse symmetric matrix	218
7.8	Addition of sparse matrices	219
7.9	Example of addition of two sparse matrices	220
7.10	Algorithm for the symbolic addition of two sparse matrices with N rows and M columns	222
7.11	Algorithm for the numerical addition of two sparse matrices with N rows	223
7.12	Product of a general sparse matrix by a column vector	224
7.13	Algorithm for the product of a general sparse matrix by a full column vector	225
7.14	Product of a row vector by a general sparse matrix.	226
7.15	Example of product of a full row vector by a general sparse matrix.	226
7.16	Algorithm for the product of a full row vector by a general sparse matrix.	227
7.17	Product of a symmetric sparse matrix by a column vector.	228
7.18	Algorithm for the product of a symmetric sparse matrix by a full column vector	229
7.19	Multiplication of sparse matrices	230
7.20	Example of product of two matrices which are stored by rows.	231
7.21	Algorithm for the symbolic multiplication of two sparse matrices given in row-wise format	233
7.22	Algorithm for the numerical multiplication of two sparse matrices given in row-wise format	234
7.23	Triangular factorization of a sparse symmetric matrix given in row-wise format	235
7.24	Numerical triangular factorization of a sparse symmetric matrix given in row-wise format	238
7.25	Algorithm for the symbolic triangular factorization of a symmetric sparse matrix A	240
7.26	Algorithm for the numerical triangular factorization of a symmetric positive definite sparse matrix A	242
7.27	Example of forward and backward substitution	245
7.28	Algorithm for the solution of the system $U^T D U \mathbf{x} = \mathbf{b}$	246
8	Connectivity and Nodal Assembly	249
8.1	Introduction	249
8.2	Boundary conditions for scalar problems	251
8.3	Boundary conditions for vector problems	252
8.4	Example of a connectivity matrix	256
8.5	Example of a nodal assembly matrix	257
8.6	Algorithm for the symbolic assembly of a symmetric nodal assembly matrix	259
8.7	Algorithm for the numerical assembly of an element matrix and vector into the nodal assembly matrix A and right-hand vector b : Symmetric case	261
8.8	Algorithm for the numerical assembly of an element matrix and vector into the nodal assembly matrix A and right-hand vector b : General case	264

9	General Purpose Algorithms	267
9.1	Introduction	267
9.2	Multiplication of the inverse of a lower triangular matrix by a general matrix	268
9.3	Algorithm for the symbolic multiplication of the inverse of a lower triangular matrix U^{-T} by a general matrix B	269
9.4	Algorithm for the numerical multiplication of the inverse of a lower triangular matrix U^{-T} by a general matrix B	270
9.5	Algorithm for the multiplication of the inverse of an upper triangular unit diagonal matrix U by a full vector x	272
9.6	Algorithm for the multiplication of the transpose inverse of an upper triangular unit diagonal matrix U by a full vector	273
9.7	Solution of linear equations by the Gauss-Seidel iterative method	274
9.8	Algorithm for the iterative solution of linear equations by the Gauss-Seidel method	275
9.9	Checking the representation of a sparse matrix	276
9.10	Printing and displaying a sparse matrix	277
9.11	Algorithm for transforming a $RR(C)U$ of a symmetric matrix into a $RR(U)U$ of the same matrix	278
9.12	Algorithm for the pre-multiplication of a sparse matrix A by a diagonal matrix D	279
9.13	Algorithm for copying a sparse matrix from IA, JA, AN to IB, JB, BN	279
	Bibliography and Index	281

Preface to the Electronic Edition

This is an electronic edition of the classic book *Sparse Matrix Technology* by Sergio Pissanetzky, originally published in English by Academic Press, London, in 1984, and later translated into Russian and published by MIR, Moscow, in 1988. The electronic edition has been typed from the original, with only minor changes of format where dictated by electronics.

Preface

As computers grow in power and speed, matrices grow in size. In 1968, practical production calculations with linear algebraic systems of order 5,000 were commonplace, while a “large” system was one of order 10,000 or more.^a

In 1978, an over determined problem with 2.5 million equations in 400,000 unknowns was reported;^b in 1981, the magnitude of the same problem had grown: it had 6,000,000 equations, still in 400,000 unknowns.^c The matrix of coefficients had 2.4×10^{12} entries, most of which were zero: it was a *sparse* matrix. A similar trend toward increasing size is observed in eigenvalue calculations, where a “large” matrix is one of order 4,900 or 12,000.^d Will matrix problems continue to grow even further? Will our ability to solve them increase at a sufficiently high rate?

But this is only one side of the question. The other side concerns the microcomputer explosion. Microcomputers now have about the same power as large computers had two decades ago. Are users constrained to solving matrix problems of the same size as those of twenty years ago?

The owner of a microcomputer may not care too much about the cost of computation; the main difficulty is storage. On a large machine, the cost of solving a matrix problem increases rapidly if the size of the problem does, because both storage and labor grow. The overall cost becomes a primary consideration. How can such cost be minimized for a given problem and installation?

Answers to these and other related questions are given in this book for the following classes of matrix problems: direct solution of sparse linear algebraic equations, solution of sparse standard and generalized eigenvalue problems, and sparse matrix algebra. Methods are described which range from very simple yet surprisingly effective ideas to highly sophisticated algorithms. Sparse matrix technology is now a well established discipline, which was defined as “the art of handling sparse matrices”.^e It is composed of a beautiful blend of theoretical developments, numerical experience and practical considerations. It is not only an important computational tool in a broad spectrum

^aIn the Preface, the pertinent references are given as footnotes, because this enhances clarity. The full list of references is given at the end of the book. Tinney, 1969,²³⁷ p.28; Willoughby, 1971,²⁵⁰ p.271.

^bKolata, 1978.¹⁴⁴

^cGolub and Plemmons, 1981, ¹⁰⁶ p.3.

^d Cullum and Willoughby, 1981,⁴² p. 329; Parlett, 1980,¹⁷⁵ p. XIII.

^eHarary, 1971.¹²⁴

of computational areas,^f but also is in itself a valuable contribution to the general development of computer software. The new ideas developed during the last fifteen years were used to devise nearly optimum algorithms for a variety of matrix problems. Research in the field is currently very active and the spectrum of applications broadens continuously. Sparse matrix technology is here and will stay.

The concept expressing the nature of our concern is contained in the title of the book. Technology is applied science, the science or study of the practical or industrial arts.^g The phrase “sparse matrix technology” was an everyday saying in the early nineteen seventies at the IBM T. J. Watson Research Center.^h Nowadays it seems to be in desuetude. The material for the book was selected from the several Symposia and Congresses on large matrix problems regularly held since 1968.ⁱ Major sources of inspiration were: an advanced course with four review articles,^j excellent survey articles^k and books,^l a collection of papers,^m and many publications which are cited where pertinent. Several basic ideas can be found in the literature published before 1973.ⁿ No attempt is made, however, to cover such an important amount of material. Rather, the fundamental methods and procedures are introduced and described in detail, the discussion reaching the point where the reader can understand the specialized literature on each subject. A unified treatment is provided whenever possible, although, like any field of human knowledge which grows fast, sparse matrix technology has grown unevenly. Some areas are well developed, while other areas lack further research. We have not included proofs of all the theorems, except when they are closely related to practical techniques which are used subsequently. The concepts and methods are introduced at an elementary level, in many cases with the help of simple examples. Many fundamental algorithms are described and carefully discussed. Ready-to-use very efficient and professional algorithms are given in Fortran. The reader is assumed to be familiar with this popular language. The algorithms, however, are explained so clearly that even a person with a limited knowledge of Fortran can understand them and eventually translate them into other languages. Linear algebra and graph theory are used extensively in the book. No particular acquaintance with these subjects is necessary because all definitions and properties are introduced from the beginning, although some preparation

^fRose and Willoughby, 1972.¹⁹⁸

^gWebster’s Dictionary, second edition, 1957.

^hWilloughby, 1971;²⁵⁰ Rose and Willoughby, 1972,¹⁹⁸ Preface; Willoughby, 1972;²⁵¹ Hachtel, 1976,¹¹⁷ p. 349.

ⁱWilloughby, 1969;²⁴⁹ Reid, 1971a;¹⁸⁷ Rose and Willoughby, 1972,¹⁹⁸ Bunch and Rose, 1976;²⁸ Duff and Stewart, 1979;⁶⁸ Duff, 1981b.⁶¹ The Proceedings of the Symposium held at Fairfield Glade, Tennessee, in 1982, will be published as a special issue of the SIAM Journal on Scientific and Statistical Computing, and possibly other SIAM journals, to appear in 1983. The Software Catalog prepared in conjunction with the Symposium is available (Heath, 1982.¹²⁶)

^jBarker, 1977.¹⁰

^kDuff, 1977,⁵⁵ 1982.⁶²

^lWilkinson, 1965;²⁴⁷ Parlett, 1980;¹⁷⁵ George and Liu, 1981.⁹⁷

^mBjörck *et al.* 1981¹⁶

ⁿBrayton *et al.* 1970;¹⁹ Willoughby, 1972;²⁵¹ Tewarson, 1973.²³⁵

may be helpful. An extensive bibliography and a survey of the relevant literature are included in many sections. The book fills the gap between books on the design of computer algorithms and specialized literature on sparse matrix techniques, on the one side, and user needs and application oriented requirements on the other.

The purpose of the book is to bring sparse matrix technology within reach of engineers, programmers, analysts, teachers and students. This book will be found helpful by everyone who wishes to develop his own sparse matrix software, or who is using it and wishes to understand better how it operates, or who is planning to acquire a sparse matrix package and wishes to improve his understanding of the subject. Teachers who need an elementary presentation of sparse matrix methods and ideas and many examples of application at a professional level, will find such material in this book.

Chapter 1 covers all fundamental material such as storage schemes, basic definitions and computational techniques needed for sparse matrix technology. It is very convenient to read at least Sections 1 to 9 and Section 12 of Chapter 1 first. The first reading may, however, be superficial. The reader will feel motivated to examine this material in more detail while reading other chapters of the book, where numerous references to sections of Chapter 1 are found.

Chapters 2 to 5 deal with the solution of linear algebraic equations. They are not independent. The material in Chapter 2 is rather elementary, but its form of presentation serves as an introduction for Chapters 4 and 5, which contain the important material. Chapter 3 deals with numerical errors in the case where the linear system is sparse, and also serves as an introduction to Chapters 4 and 5. This material is not standard in the literature. Sparse matrix methods and algorithms for the direct solution of linear equations are presented in Chapters 4 and 5. Chapter 4 deals with symmetric matrices, and Chapter 5 with general matrices.

The calculation of eigenvalues and eigenvectors of a sparse matrix, or of a pair of sparse matrices in the case of a generalized eigenvalue problem, is discussed in Chapter 6. Chapter 6 can be read independently, except that some references are made to material in Chapters 1 and 7.

Chapters 7, 8 and 9 deal with sparse matrices stored in row-wise format. Algorithms for algebraic operations, triangular factorization and back substitution are explicitly given in Fortran and carefully discussed in Chapter 7. The material in Chapter 1 is a prerequisite, particularly Sections 8, 9 and 10 and 12 to 17. In addition, Chapter 2 is a prerequisite for Sections 23 to 28 of Chapter 7. Chapter 8 covers the sparse matrix techniques associated with mesh problems, in particular with the finite element method, and in Chapter 9 we present some general purpose Fortran algorithms.

Sparse matrix technology has been applied to almost every area where matrices are employed. Anyone interested in a particular application may find it helpful to read the literature where the application is described in detail, in addition to the relevant chapters of this book. A list of bibliographical references sorted by application was published^o and many papers describing a

^oDuff, 1977,⁵⁵ p. 501.

variety of applications can be found in the Proceedings of the 1980 IMA Conference^p and in other publications^q

Good, robust sparse matrix software is now commercially available. The Sparse Matrix Software Catalog^r lists more than 120 programs. Many subroutines are described in the Harwell Catalogue^s and two surveys have also been published.^t Producing a good piece of sparse matrix software is not an easy task. It requires expert programming skills. As in any field of engineering, the software designer must build a prototype, test it carefully^u and improve it before the final product is obtained and mass production starts. In software engineering, mass production is equivalent to obtaining multiple copies of a program and implementing them in many different installations. This requires transportability. From the point of view of the user, the software engineer must assume responsibility for choosing the right program and file structures and installing them into the computer. For the user, the product is not the program but the result. The desirable attributes of a good program are not easily achieved.^v In this book, the characteristics and availability of software for each particular application are discussed in the corresponding sections.

I would like to acknowledge the collaboration of Neil Callwood. He has read the manuscript several times, correcting many of my grammatical infelicities, and is responsible for the “British flavour” that the reader may find in some passages. I would also like to acknowledge the patience and dedication Mrs. Carlota R. Glücklich while typing the manuscript and coping with our revisions.

Sergio Pissanetzky
Sergio@SciControls.com

January 1984

^pDuff, 1981b.⁶¹

^qBunch and Rose, 1976;²⁸ Duff and Stewart, 1979.⁶⁸

^rHeath, 1982.¹²⁶

^sHopper, 1980.¹³⁰

^tDuff, 1982;⁶² Parlett, 1983.¹⁷⁶

^uDuff, 1979;⁵⁷ Eisenstat *et al.* 1979;⁷⁵ Duff *et al.* 1982.⁶⁹

^vGentleman and George, 1976;⁸⁷ Silvester, 1980.²¹⁵

$$A = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & A_{11} & & A_{13} & A_{14} & \\ 2 & & A_{22} & & & A_{25} \\ 3 & & & A_{33} & & A_{35} \\ 4 & & & & A_{44} & \\ 5 & \text{symmetric} & & & & A_{55} \end{array}$$

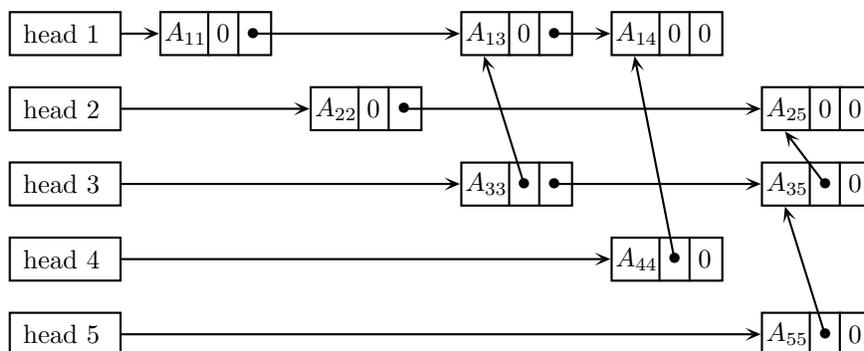


Figure 1.4: Larcombe's version of Knuth's storage scheme for symmetric matrices with no zero elements on the diagonal.

1.8 The sparse row-wise format

The *sparse row-wise format* (Chang, 1969;²⁹ Curtis and Reid, 1971b;⁴⁶ Gustavson, 1972¹¹²) to be described here is one of the most commonly used storage schemes for sparse matrices. The scheme has minimal storage requirements and at the same time it has proved to be very convenient for several important operations such as addition, multiplication, permutation and transposition of sparse matrices, the solution of linear equations with a sparse matrix of coefficients by either direct or iterative methods, etc. In this scheme, the values of the nonzero elements of the matrix are stored by rows, along with their corresponding column indices, in two arrays, say AN and JA, respectively. An array of pointers, say IA, is also provided to indicate the locations in AN and JA where the description of each row begins. An extra entry in IA contains a pointer to the first empty position in JA and AN. An example is convenient at this point. Consider the matrix:

$$A = \begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 0 & 0 & 1. & 3. & 0 & 0 & 0 & 5. & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 7. & 0 & 1. & 0 & 0 \end{array}$$

A is represented as follows:

$$\begin{array}{rcccccc}
 & & 1 & 2 & 3 & 4 & 5 & 6 \\
 \text{IA} & = & 1 & 4 & 4 & 6 & & \\
 \text{JA} & = & 3 & 4 & 8 & 6 & 8 & \\
 \text{AN} & = & 1. & 3. & 5. & 7. & 1. &
 \end{array}
 \qquad \text{RR(C)O}$$

The description of row 1 of A begins at the position $\text{IA}(1) = 1$ of AN and JA. Since the description of row 2 begins at $\text{IA}(2) = 4$, this means that row 1 of A is described in positions 1, 2 and 3 of AN and JA. In this example:

$$\begin{array}{ll}
 \text{IA}(1) & = 1 \quad \text{first row begins at JA}(1) \text{ and AN}(1). \\
 \text{IA}(2) & = 4 \quad \text{second row begins at JA}(4) \text{ and AN}(4) \\
 \text{IA}(3) & = 4 \quad \text{third row begins at JA}(4) \text{ and AN}(4). \text{ Since this is the same position at} \\
 & \quad \text{which row 2 begins, this means that row 2 is empty.} \\
 \text{IA}(4) & = 6 \quad \text{this is the first empty location in JA and AN. The description of row 3} \\
 & \quad \text{thus ends at position } 6 - 1 = 5 \text{ of JA and AN.}
 \end{array}$$

In general, row r of A is described in positions $\text{IA}(r)$ to $\text{IA}(r+1) - 1$ of JA and AN, except when $\text{IA}(r+1) = \text{IA}(r)$ in which case row r is empty. If matrix A has m rows, then IA has $m + 1$ positions.

This representation is said to be *complete* because the entire matrix A is represented, and *ordered* because the elements of each row are stored in the ascending order of their column indices. It is thus a **R**ow-wise **R**epresentation **C**omplete and **O**rdered, or RR(C)O.

The arrays IA and JA represent the structure of A , given as the set of the adjacency lists of the graph associated with A . If an algorithm is divided into a symbolic section and a numerical section (Section 1.12), the arrays IA and JA are computed by the symbolic section, and the array AN by the numerical section.

Gustavson (1972)¹¹² also proposed a variant of row-wise storage, suitable for applications requiring both row and column operations. A is stored row-wise as described, and in addition the structure of A^T is computed and also stored row-wise. A row-wise representation of the structure of A^T is identical to a column-wise representation of the structure of A . It can be obtained by transposition of the row-wise structure of A (Chapter 7). This scheme has been used, for example, for linear programming applications (Reid, 1976).¹⁸⁹

A much simpler row-oriented scheme was proposed by Key (1973)¹⁴¹ for unsymmetric matrices. The nonzeros are held in a two-dimensional array of size n by m , where n is the order of the matrix and m the maximum number of nonzeros in a row. This scheme is easy to manipulate but has the disadvantage that m may not be predictable and may turn out to be large.

1.9 Ordered and unordered representations

Sparse matrix representations do not necessarily have to be ordered, in the sense that the elements of each row can be stored in any order while still preserving the order of the rows. The matrix A

The k th step consists of the elimination of the nonzeros on column k of $\mathbf{A}^{(k)}$ both above and below the diagonal. Row k is first normalized by dividing all its elements by the diagonal element. Then, convenient multiples of the normalized row k are subtracted from all those rows which have a nonzero in column k either above or below the diagonal. The matrix $\mathbf{A}^{(k+1)}$ is thus obtained with zeros in its k initial columns. This process is continued until, at the end of step n , the identity matrix $\mathbf{A}^{(n+1)} \equiv \mathbf{I}$ is obtained. The k th step of Gauss-Jordan elimination by columns is equivalent to pre-multiplication of $\mathbf{A}^{(k)}$ by \mathbf{D}_k^{-1} and by the complete column elementary matrix $(\mathbf{T}_k^C)^{-1}$:

$$\mathbf{A}^{(k+1)} = (\mathbf{T}_k^C)^{-1} \mathbf{D}_k^{-1} \mathbf{A}^{(k)} \tag{2.37}$$

where $\mathbf{A}^{(1)} \equiv \mathbf{A}$ and:

$$\begin{aligned} (\mathbf{D}_k)_{kk} &= A_{kk}^{(k)} \\ (\mathbf{T}_k^C)_{ik} &= A_{ik}^{(k)} \quad \text{for all } i \neq k \end{aligned} \tag{2.38}$$

Thus, we have:

$$(\mathbf{T}_n^C)^{-1} \mathbf{D}_n^{-1} \dots (\mathbf{T}_2^C)^{-1} \mathbf{D}_2^{-1} (\mathbf{T}_1^C)^{-1} \mathbf{D}_1^{-1} \mathbf{A} = \mathbf{I}. \tag{2.39}$$

The factorized form of \mathbf{A} is:

$$\mathbf{A} = \mathbf{D}_1 \mathbf{T}_1^C \mathbf{D}_2 \mathbf{T}_2^C \dots \mathbf{D}_n \mathbf{T}_n^C, \tag{2.40}$$

and the *product form of the inverse* in terms of column matrices is:

$$\mathbf{A}^{-1} = (\mathbf{T}_n^C)^{-1} \mathbf{D}_n^{-1} \dots (\mathbf{T}_2^C)^{-1} \mathbf{D}_2^{-1} (\mathbf{T}_1^C)^{-1} \mathbf{D}_1^{-1}. \tag{2.41}$$

The close relationship between this expression and the elimination form of the inverse, Expression (2.24), will be discussed in Section 2.10. The results of the elimination are usually recorded as a table of factors:

$$\begin{array}{ccc} (\mathbf{D}_1)_{11}^{-1} & (\mathbf{T}_2^C)_{12} & (\mathbf{T}_3^C)_{13} \dots \\ (\mathbf{T}_1^C)_{21} & (\mathbf{D}_2)_{22}^{-1} & (\mathbf{T}_3^C)_{23} \\ (\mathbf{T}_1^C)_{31} & (\mathbf{T}_2^C)_{32} & (\mathbf{D}_3)_{33}^{-1} \\ \vdots & \vdots & \vdots \end{array} \tag{2.42}$$

By Equation (2.38), this table is formed simply by leaving each off-diagonal $A_{ik}^{(k)}$ where it is obtained. The diagonal is obtained, as in Gauss elimination, by storing the reciprocals of the diagonal elements used to normalize each row. The lower triangle and diagonal of this table are thus identical to those of the Gauss table. Expressions (2.40) and (2.41) indicate how to use the table (2.42). When solving linear equations by means of $\mathbf{x} = \mathbf{A}^{-1} \mathbf{b}$, Equation (2.41) is used, with the matrices $(\mathbf{T}_k^C)^{-1}$ obtained from the table by reversing the signs of the off-diagonal elements of

column k (Property 2.4(d)). The matrices D_k^{-1} are directly available from the table. The product of A with any matrix or vector can also be computed using the table, as indicated by Equation (2.40).

Gauss-Jordan elimination can also be performed by rows. The version by columns requires the addition of multiples of row k to all other rows in order to cancel the off-diagonal elements of column k . This process can be understood conceptually as the construction of new equations which are linear combinations of the original ones. On the other hand, in Gauss-Jordan elimination by rows, we add multiples of *column* k to all other columns, in such a way that the off-diagonal elements of *row* k become zero. This process can be viewed as the construction of new *unknowns* which are linear combinations of the original ones and which satisfy linear equations with some zero coefficients. Alternatively, we can forget about the system of linear equations and view the row algorithm as the triangularization of A^T , the transpose of A , by columns. Doing this, we obtain the equivalent of Expression (2.41):

$$(A^T)^{-1} = (T_n^C)^{-1}(D_n')^{-1} \dots (T_2^C)^{-1}(D_2')^{-1}(T_1^C)^{-1}(D_1')^{-1}, \quad (2.43)$$

which by transposition and using $(A^T)^{-1} = (A^{-1})^T$ yields:

$$A^{-1} = (D_1')^{-1}(T_1^R)^{-1}(D_2')^{-1}(T_2^R)^{-1} \dots (D_n')^{-1}(T_n^R)^{-1} \quad (2.44)$$

Equation (2.44) is the product form of the inverse in terms of row matrices. The elimination by rows is equivalent to multiplying A *from the right* by Expression (2.44). The nontrivial elements of the matrices of Expression (2.44) are recorded as a table of factors in the usual way, and the table can be used to solve linear equations or to multiply either A or A^{-1} by any matrix or vector.

2.10 Relation between the elimination form of the inverse and the product form of the inverse

From the preceding section it should be clear that Gauss-Jordan elimination by columns can be performed equally well if we first eliminate all nonzeros from the lower triangle of A , and then all nonzeros from the upper triangle of A . In fact, when we start at the upper left-hand corner of A , we can eliminate lower and upper portions of columns in any order, provided only that upper portions are eliminated in order, lower portions are also eliminated in order, and the upper portion of any column k is eliminated *after* the lower portion of the preceding column. This statement holds true due to the fact that a row $k + 1$ is obtained in final form immediately after the lower portions of columns 1 to k have been eliminated and row $k + 1$ has been normalized; row $k + 1$ can then be used either immediately or at any later stage to eliminate the upper portion of column $k + 1$, provided that the upper portions of columns 1 to k have been previously eliminated. These facts can be stated formally using the properties of the elementary matrices (Section 2.4). We use Property 2.4(c) to express T_k^C as follows:

$$T_k^C = L_k^C U_k^C, \quad (2.45)$$

Table 3.1. Bounds for the norms of \mathbf{L} , expression for n_{ij} (see Equation 2.16), and bounds for the norm of the error matrix \mathbf{E} for the factorization $\mathbf{LU} = \mathbf{A} + \mathbf{E}$, where all matrices are of order n . The bandwidth of band matrices is assumed not to exceed n .

A	Bounds for \mathbf{L}	n_{ij}	Error bounds for factorization
Sparse	$\ \mathbf{L}\ _1 \leq a_M (\max_j c_j^L + 1)$	$\sum_{k=1}^m n_{ij}^{(k)}$	$\ \mathbf{E}\ _1 \leq 3.01\varepsilon_M a_M \max_j \sum_{i=1}^n n_{ij}$
	$\ \mathbf{L}\ _\infty \leq a_M (\max_i r_i^L + 1)$	$m = \min(i, j)$	$\ \mathbf{E}\ _\infty \leq 3.01\varepsilon_M a_M \max_i \sum_{j=1}^n n_{ij}$
Full	$\ \mathbf{L}\ _1 \leq a_M n$	$\min(i, j)$	$\ \mathbf{E}\ _1, \ \mathbf{E}\ _\infty \leq \frac{3.01}{2} \varepsilon_M a_M n(n+1)$
	$\ \mathbf{L}\ _\infty \leq a_M n$		
Band $ \setminus\beta\setminus\beta\setminus $	$\ \mathbf{L}\ _1 \leq a_M(\beta + 1)$	$\max[0, \min(i, j, i - j + \beta + 1, j - i + \beta + 1)]$	$\ \mathbf{E}\ _1, \ \mathbf{E}\ _\infty \leq 3.01\varepsilon_M a_M(\beta + 1)^2$
	$\ \mathbf{L}\ _\infty \leq a_M(\beta + 1)$		
Band $ \setminus\beta\setminus 2\beta\setminus $	$\ \mathbf{L}\ _1 \leq a_M(\beta + 1)$	$\max[0, \min(i, j, i - j + 2\beta + 1, j - i + \beta + 1, \beta + 1)]$	$\ \mathbf{E}\ _1, \ \mathbf{E}\ _\infty \leq 3.01\varepsilon_M a_M(\beta + 1) \times (2\beta + 1)$
	$\ \mathbf{L}\ _\infty \leq a_M(\beta + 1)$		

Then, the computed result \mathbf{w} satisfies the exact relation:

$$\mathbf{L}\mathbf{w} = \mathbf{b} + \delta\mathbf{b} \quad (3.52)$$

where, from Equations 3.47 and 3.50, the following bounds hold for the components of $\delta\mathbf{b}$:

$$|\delta b_i| \leq 3.01\varepsilon_M b_{Mi}(r_i^L + 1). \quad (3.53)$$

A less tight but simpler bound is obtained if b_M is the absolute value of the largest element of all the vectors $\mathbf{b}^{(k)}$, so that $b_{Mi} \leq b_M$ and:

$$|b_i^{(k)}| \leq b_M; \quad i = 1, 2, \dots, n; \quad k \leq i. \quad (3.54)$$

Then:

$$|\delta b_i| \leq 3.01\varepsilon_M b_M(r_i^L + 1). \quad (3.55)$$

Backward substitution is the solution of $\mathbf{U}\mathbf{x} = \mathbf{w}$. It can be viewed as an algorithm with n steps, where the sequence of vectors $\mathbf{w}^{(n)} \equiv \mathbf{w}, \mathbf{w}^{(n-1)}, \dots, \mathbf{w}^{(2)}, \mathbf{w}^{(1)}$ is computed, with $\mathbf{w}^{(k)}$ and $\mathbf{w}^{(k-1)}$ having their components k to n identical. Step k , $k = n, n-1, \dots, 1$, is:

$$\begin{aligned} x_k &= w_k^{(k)} \\ w_i^{(k-1)} &= w_i^{(k)} - U_{ik}x_k + g_i^{(k)}; \quad i = 1, \dots, k-1, \end{aligned} \quad (3.56)$$

where $g_i^{(k)}$ is the error introduced by the floating point computation. The operations performed on an element $w_i, i < n$, are:

$$w_i - U_{in}x_n + g_i^{(n)} + g_i^{(n)} - U_{i,n-1}x_{n-1} + g_i^{(n-1)} - \dots - U_{i,i+1}x_{i+1} + g_i^{(i+1)} = x_i \quad (3.57)$$

or:

$$w_i + \sum_{k=i+1}^n g_i^{(k)} = \sum_{k=i}^n U_{ik}x_k; \quad i < n, \quad (3.58)$$

Thus, if we define the error vector $\delta\mathbf{w}$:

$$\begin{aligned} \delta w_i &= \sum_{k=i+1}^n g_i^{(k)}; \quad i < n \\ \delta w_n &= 0, \end{aligned} \quad (3.59)$$

we have the following exact relation between the computed numbers:

$$\mathbf{U}\mathbf{x} = \mathbf{w} + \delta\mathbf{w}. \quad (3.60)$$

In order to obtain bounds for $\delta\mathbf{w}$, we let $w_{Mi} = \max_k |w_i^{(k)}|$, so that:

$$|w_i^{(k)}| \leq w_{Mi}; \quad 1 \leq i \leq n; \quad i \leq k \leq n. \quad (3.61)$$

In particular, for $k = i, w_i^{(i)} = x_i$, so that $|x_i| \leq w_{Mi}$. We also let w_M be the largest w_{Mi} ; therefore:

$$|w_i^{(k)}| \leq w_M; \quad i = 1, 2, \dots, n; \quad k \geq i. \quad (3.62)$$

Then, using Equation 3.22:

$$|g_i^{(k)}| \leq 3.01\varepsilon_M w_{Mi}; \quad k > i \quad (3.63)$$

and

$$|\delta w_i| \leq 3.01\varepsilon_M w_{Mi} r_i^U, \quad (3.64)$$

where r_i^U is the number of off-diagonal nonzeros in row i of \mathbf{U} . Alternatively, using Equation 3.62:

$$|\delta w_i| \leq 3.01\varepsilon_M w_M r_i^U. \quad (3.65)$$

Finally, we consider the residual

$$\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b} \quad (3.66)$$

obtained when the solution \mathbf{x} of System 3.1 is computed using floating point arithmetic. Using Equations 3.41, 3.52 and 3.60, we obtain:

$$\mathbf{r} = -\mathbf{E}\mathbf{x} + \mathbf{L}\delta\mathbf{w} + \delta\mathbf{b}. \quad (3.67)$$

Taking the 1-norm or the ∞ -norm, we have:

$$\|\mathbf{r}\| \leq \|\mathbf{E}\|\|\mathbf{x}\| + \|\mathbf{L}\|\|\delta\mathbf{w}\| + \|\delta\mathbf{b}\|. \quad (3.68)$$

From Equation 3.62 we obtain bounds for the norms of \mathbf{x} :

$$\begin{aligned} \|\mathbf{x}\|_1 &\leq nw_M \\ \|\mathbf{x}\|_\infty &\leq w_M. \end{aligned} \quad (3.69)$$

Bounds for the norms of \mathbf{E} and \mathbf{L} are given in table 3.1. Bounds for the norms of $\delta\mathbf{w}$ and $\delta\mathbf{b}$ were obtained from Equations 3.65 and 3.55, respectively, and are listed in Table 3.2. Thus, a bound for $\|\mathbf{r}\|$ can be computed using Equation 3.68.

The residual \mathbf{r} has another interpretation. Let $\tilde{\mathbf{x}}$ be the exact solution of Equation 3.1; then $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b}$ and

$$\mathbf{r} = \mathbf{A}(\mathbf{x} - \tilde{\mathbf{x}}), \quad (3.70)$$

Table 3.2. Values of some parameters and bounds for the norms $\delta\mathbf{b}$ and $\delta\mathbf{w}$ for forward and backward substitution.

A	Parameters of Section 3.4	Forward substitution	Backward substitution
Sparse	See Section 3.4	$\ \delta\mathbf{b}\ _1 \leq 3.01\varepsilon_M b_M n_L$ $\ \delta\mathbf{b}\ _\infty \leq 3.01\varepsilon_M b_M (\max_i r_i^L + 1)$	$\ \delta\mathbf{w}\ _1 \leq 3.01\varepsilon_M w_M n'_U$ $\ \delta\mathbf{w}\ _\infty \leq 3.01\varepsilon_M w_M \max_i r_i^U$
Full	$r_i^L = i - 1$ $r_i^U = n - i$ $n_L = n(n + 1)/2$ $n'_U = n(n - 1)/2$	$\ \delta\mathbf{b}\ _1 \leq (3.01/2)\varepsilon_M b_M n(n + 1)$ $\ \delta\mathbf{b}\ _\infty \leq 3.01\varepsilon_M b_M n$	$\ \delta\mathbf{w}\ _1 \leq (3.01/2)\varepsilon_M w_M n(n - 1)$ $\ \delta\mathbf{w}\ _\infty \leq 3.01\varepsilon_M w_M (n - 1)$
Band $\setminus \setminus \beta \setminus \setminus \beta \setminus \setminus$	$r_i^L = \min(i - 1, \beta)$ $r_i^U = \min(n - i, \beta)$ $n_L = (n - \beta/2)(\beta + 1)$ $n'_U = (n - \beta/2 - 1/2)\beta$	$\ \delta\mathbf{b}\ _1 \leq 3.01\varepsilon_M b_M (n - \beta/2)(\beta + 1)$ $\ \delta\mathbf{b}\ _\infty \leq 3.01\varepsilon_M b_M (\beta + 1)$	$\ \delta\mathbf{w}\ _1 \leq 3.01\varepsilon_M w_M (n - \beta/2 - 1/2)\beta$ $\ \delta\mathbf{w}\ _\infty \leq 3.01\varepsilon_M w_M \beta$
Band $\setminus \setminus \beta \setminus 2\beta \setminus \setminus$	$r_i^L = \min(i - 1, \beta)$ $r_i^U = \min(n - i, 2\beta)$ $n_L = (n - \beta/2)(\beta + 1)$ $n'_U = (2n - 2\beta - 1)\beta$	$\ \delta\mathbf{b}\ _1 \leq 3.01\varepsilon_M b_M (n - \beta/2)(\beta + 1)$ $\ \delta\mathbf{b}\ _\infty \leq 3.01\varepsilon_M b_M (\beta + 1)$	$\ \delta\mathbf{w}\ _1 \leq 3.01\varepsilon_M w_M (2n - 2\beta - 1)\beta$ $\ \delta\mathbf{w}\ _\infty \leq 6.02\varepsilon_M w_M \beta$

spondence between fill-ins and new edges added to the graph is evident. The reader can finish the exercise.

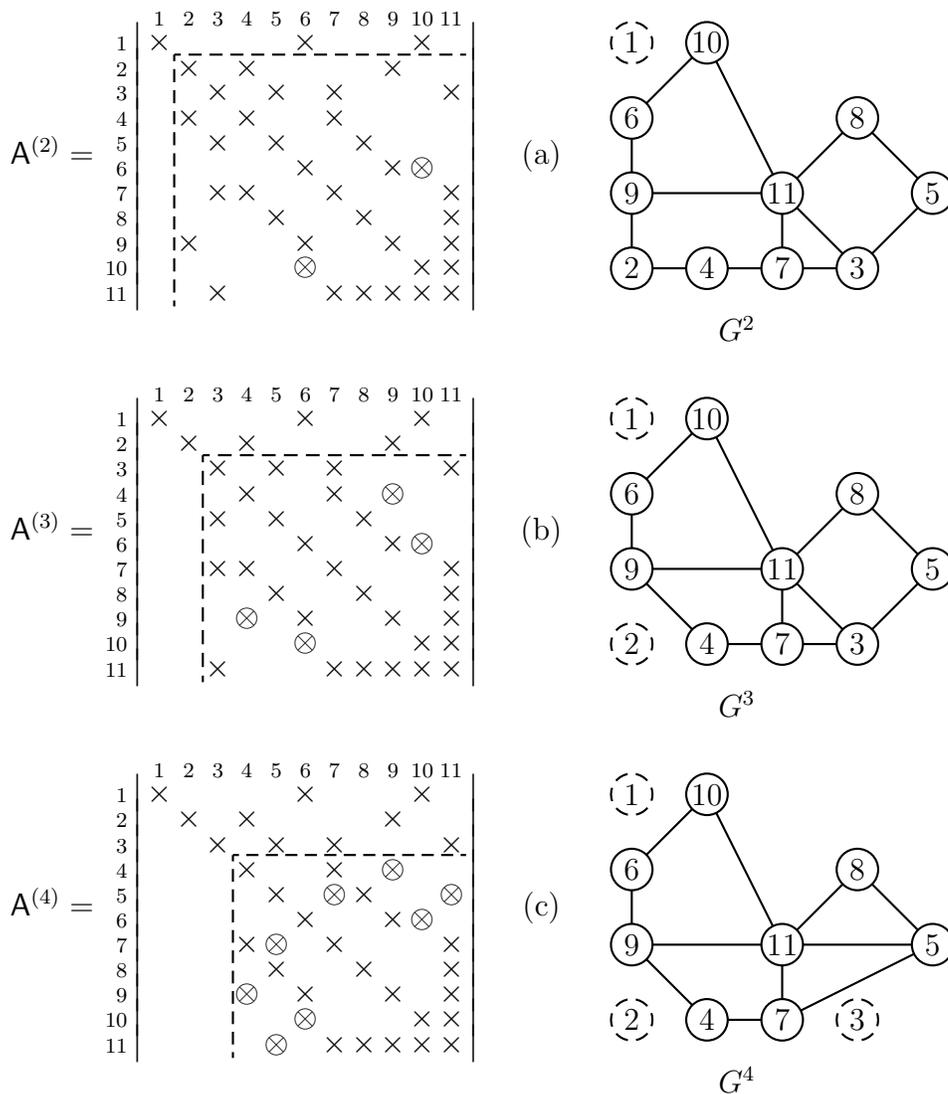


Figure 4.8: The three initial elimination steps and the corresponding elimination graphs for the matrix of Fig. 4.1(a). Fill-ins are encircled.

In terms of graph theory, Parter's rule says that the adjacent set of vertex k becomes a clique when vertex k is eliminated. Thus, Gauss elimination generates cliques systematically. Later, as elimination progresses, cliques grow or sets of cliques join to form larger cliques, a process known

$Y = \{14, 16, 1, 7\}$ and finds that Y has adjacent vertices in L_5 which have not yet been placed in any partition. Thus $S = \{7\}$ is pushed onto the stack and the algorithm branches to Step 5, where, picking $v_5 = 13$, it is found that the path can not be prolonged any longer, so $t = 1$. Letting $S = \{13\}$, the algorithm continues with Step 1, where S is not modified, and with Step 2, where Y is determined to be $\{13, 15\}$, which becomes the third partition member.

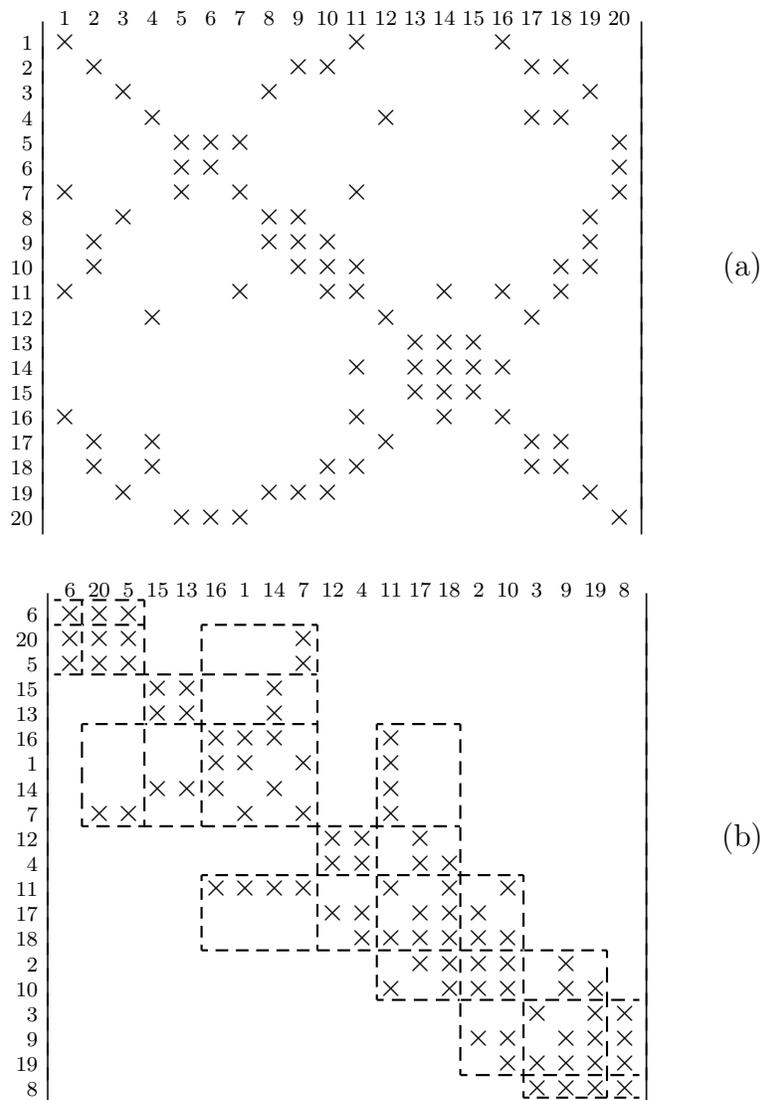


Figure 4.12: The Refined Quotient Tree algorithm. (a) Structure of the matrix corresponding to the graph of Fig. 4.2(a). (b) The permuted block matrix corresponding to the quotient tree of Fig. 4.2(c).

Table 4.1

Class of graphs	Bound for fill-in	Bound for multiplication count	Observations and references
Any, such that $\sigma = 1/2$	$c_3 n \log_2 n + O(n)$	$c_7 n^{3/2} + O[n(\log n)^2]$	Ordering time is $O[(m+n) \log n]$ if separators can be found in $O(m+n)$ time. c_3 and c_7 given by Eq. 4.23 (Lipton <i>et al.</i> , 1977 ¹⁵²)
Planar graphs (in this case $\sigma = 1/2, \alpha = 2/3, \beta = 2\sqrt{2}$)	$c_3 n \log n + O(n)$	$c_7 n^{3/2} + O[n(\log n)^2]$	$c_3 \leq 129, c_7 \leq 4002$. Ordering time is $O(n \log n)$ (Lipton and Tarjan, 1979 ¹⁵¹ ; Lipton <i>et al.</i> , 1979 ¹⁵³)
Two-dimensional finite element graphs (in this case $\sigma = 1/2, \alpha = 2/3, \beta = 4\lfloor k/2 \rfloor$)	$O(k^2 n \log n)$	$O(k^3 n^{3/2})$	k is the maximum number of boundary nodes of the elements. Ordering time is $O(n \log n)$ (Lipton <i>et al.</i> , 1979 ¹⁵³)
Regular planar grid	$\frac{31}{8} n \log_2 n + O(n)$	$\frac{829}{84} n^{3/2} + O(n \log_2 n)$	(George and Liu, 1981 ⁹⁷)
Any such that $\sigma > 1/2$	$O(n^{2\sigma})$	$O(n^{3\sigma})$	(Lipton <i>et al.</i> , 1979 ¹⁵³)
Three-dimensional grid graphs (in this case $\sigma = 2/3$)	$O(n^{4/3})$	$O(n^2)$	(Lipton <i>et al.</i> , 1979 ¹⁵³)
Any, such that $1/3 < \sigma < 1/2$	$O(n)$	$O(n^{3\sigma})$	(Lipton <i>et al.</i> , 1979 ¹⁵³)
Any, such that $\sigma = 1/3$	$O(n)$	$O(n \log_2 n)$	(Lipton <i>et al.</i> , 1979 ¹⁵³)
Any, such that $\sigma < 1/3$	$O(n)$	$O(n)$	(Lipton <i>et al.</i> , 1979 ¹⁵³)

The idea is illustrated in Fig. 4.19(a), where the rectangle represents the set of nodes of a two-dimensional finite element grid. Choose σ small separators ($\sigma = 3$ in the figure) which consist of grid lines and dissect the grid into $\sigma + 1$ blocks R_1, R_2, \dots of comparable size. If all separators are considered to form another single block, a tree partitioning is obtained as shown by the quotient tree of Fig. 4.19(b). The advantages of tree partitioning regarding the reduction of fill-in and operation count were discussed in Section 4.9. Now, let us number the nodes of each R -set sequentially, following lines from left to right as closely as possible, and starting at the bottom left as indicated by the arrows. When all R -sets have been numbered, the separators are also numbered sequentially, as the arrows show. The numbering corresponds to a monotone ordering of the tree. The matrix associated with the finite element grid is partitioned into blocks as shown in Fig. 4.19(c), where all nonzeros are confined to the cross-hatched areas. If Gauss elimination is performed on this matrix, fill-in will result only inside the cross-hatched areas and in the dotted areas. Besides, the hatched blocks are not completely full. For example, the four leading diagonal blocks are banded.

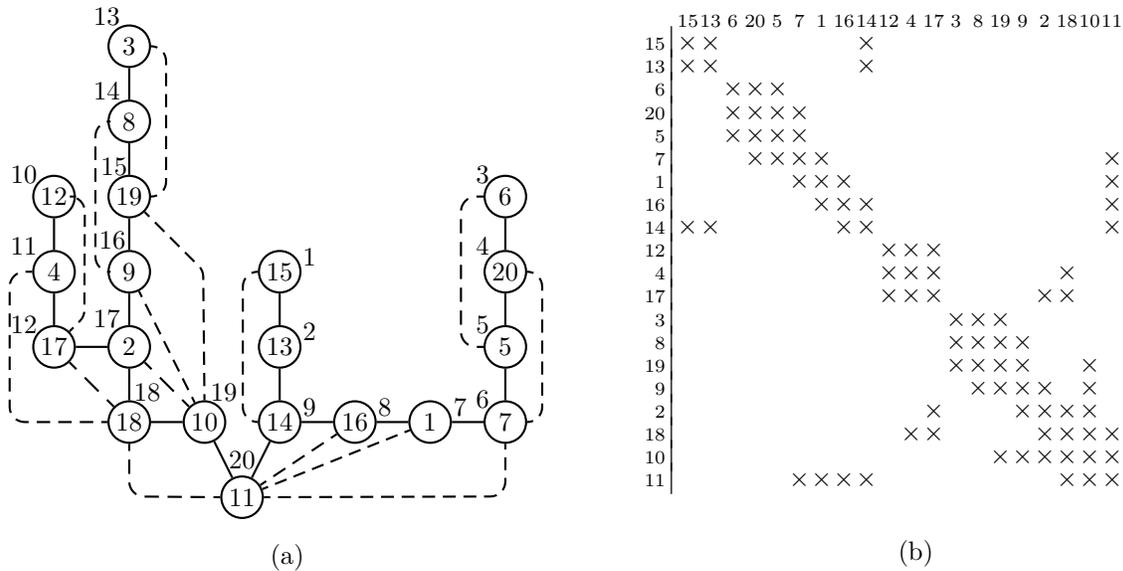


Figure 4.25: Reverse depth-first ordering, short frond strategy, for the graph of Fig. 4.2(a).

in favor of vertex 19, which is adjacent to two visited vertices: 9 and 10. The reader may continue the search and verify that the spanning tree and reverse depth-first ordering shown in Fig. 4.25(a) may be obtained. The separators (11), (10, 18, 2) and (14) can be immediately identified. The corresponding permuted matrix is shown in Fig. 4.25(b). No fill-in at all is produced by elimination on this matrix, a result obtained at a very low computational cost. The reason why an ordering with no fill-in exists for the graph of Fig. 4.2(a) is that this graph is triangulated (Rose, 1970¹⁹⁴), see Section 4.16.

Now consider the application of the long frond strategy to the same graph. Again 11 is the starting vertex. Vertices 10 and 18 are the next candidates, both of degree 5. We arbitrarily select vertex 10. At this point $V_v = \{11, 10\}$, and vertices 18, 2, 9 and 19 all have three edges leading to vertices not in V_v . Vertex 18 is discarded because it is adjacent to both visited vertices, while 2, 9 and 19 are adjacent to only one of the visited vertices. Let us choose vertex 2 to be the next vertex to visit.

At this point $V_v = \{11, 10, 2\}$ and $|\text{Adj}(w) - V_v|$ is equal to 3, 2 and 2 for vertices 17, 18 and 9, respectively. Thus, we select vertex 17. Next is vertex 4, which introduces two new edges (while 12 or 18 would have introduced only one), and finally vertex 12, which is adjacent to only two visited vertices (while 18 is adjacent to five). On backtracking to vertex 4 we find the tree arc (4, 18). Figure 4.26(a) shows one possible ordering obtained in this way. The four separators (11), (10, 2), (17, 4) and (14) can be identified. As expected, this strategy has produced more separators than the short frond strategy. The corresponding permuted matrix is shown in Fig. 4.26(b). Elimination would produce 10 fill-ins in this matrix.

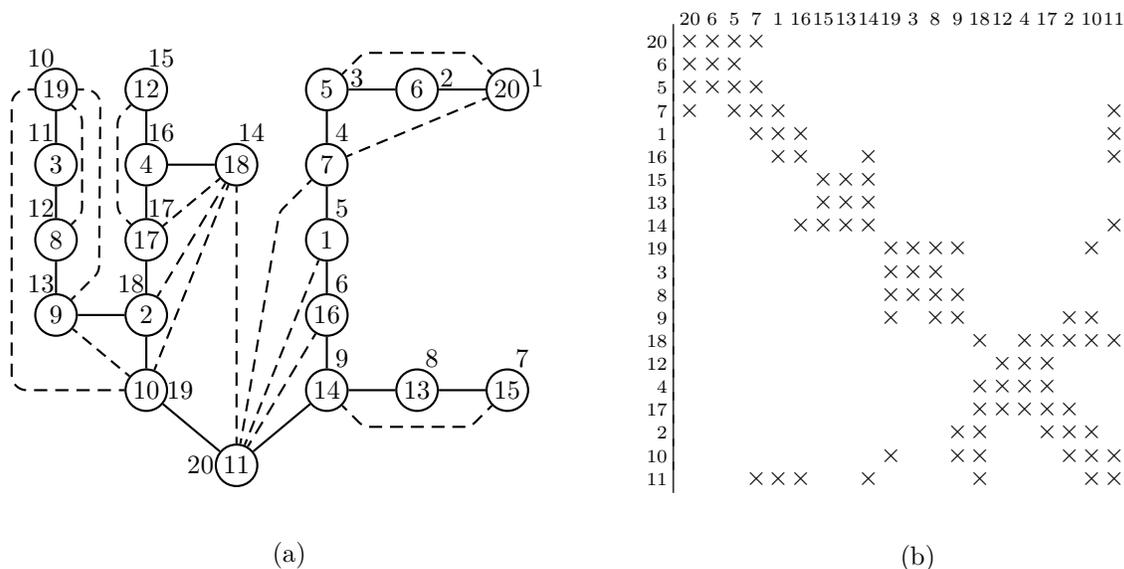


Figure 4.26: Reverse depth-first ordering, long frond strategy, for the graph of Fig. 4.2(a).

When the user is dealing with a large problem, a sophisticated ordering algorithm may be convenient, and may even determine whether the problem is tractable or not. For a medium-size problem, a simple ordering technique may often produce a large improvement as compared with no ordering at all, at a low programming cost.

4.16 Lexicographic search

In this section we continue the analysis of low fill orderings for symmetric matrices, but now from a different point of view. We consider a special class of matrices which can be ordered in such a way that Gauss elimination would cause no fill-in. Then we take advantage of the properties of such matrices to give a procedure which finds a low fill ordering for any symmetric matrix. As usual, we discuss the ideas in terms of graph theory. Let $G^A = (V, E)$ be the undirected graph associated with a symmetric matrix A , and let $G^F = (V, E \cup F)$ be the corresponding filled graph associated with $U + U^T$, where $A = U^T D U$ is the factorization of A and F is the set of new edges (or nonzeros of U) introduced during factorization. If the graph G^A has an elimination ordering for which $F = \emptyset$, i.e., no fill-in is produced if elimination is carried out in that order, we say that G^A is a *perfect elimination graph*. The ordering itself is called a *perfect elimination ordering*. Note that fill-in may result if we eliminate in a different order, even when G^A is a perfect elimination graph. Note also that every elimination graph G^F is a perfect elimination graph since no fill-in would result if elimination were performed again in the same order.

Chapter 6

Sparse Eigenanalysis

6.1 Introduction

The standard eigenvalue problem is defined by

$$\mathbf{A} \mathbf{x} = \lambda \mathbf{x} \tag{6.1}$$

where \mathbf{A} is the given n by n matrix. It is desired to find the *eigenpairs* (λ, \mathbf{x}) of \mathbf{A} , where λ is an *eigenvalue* and \mathbf{x} is the corresponding *eigenvector*. The generalized eigenvalue problem is

$$\mathbf{A} \mathbf{x} = \lambda \mathbf{B} \mathbf{x} \tag{6.2}$$

where \mathbf{A} and \mathbf{B} are given n by n matrices and again we wish to determine λ and \mathbf{x} . For historical reasons the pair \mathbf{A}, \mathbf{B} is called a *pencil* (Gantmacher, 1959⁸³). When $\mathbf{B} = \mathbf{I}$ the generalized problem reduces to the standard one.

Both for simplicity and to follow the general trend imposed by most of the literature and existing software, we restrict the analysis to the case where \mathbf{A} is real symmetric and \mathbf{B} is real symmetric and positive definite, except when stated otherwise. Almost all the results become valid for hermitian matrices when the conjugate transpose superscript H is written in place of the transpose superscript T . On the other hand, an eigenvalue problem where \mathbf{A} or \mathbf{A} and \mathbf{B} , are hermitian, can be solved using software for real matrices (Section 6.15).

Equation 6.1 has a nonzero solution \mathbf{x} when

$$\text{Det}(\mathbf{A} - \lambda \mathbf{I}) = 0. \tag{6.3}$$

This is a polynomial equation of the n th degree in λ , which has n roots $\lambda_1, \lambda_2, \dots, \lambda_n$. The roots are the eigenvalues of \mathbf{A} , and they may be either all different or there may be multiple roots with any *multiplicity*. When \mathbf{A} is real symmetric, the eigenvalues are all real. The simplest example is the identity matrix \mathbf{I} , which has an eigenvalue equal to 1 with multiplicity n . To each eigenvalue

array of pointers IC at lines 5 and 24. The multiple switch array IX is initialized to 0 at lines 2 and 3. I, defined at line 4, identifies each row. The DO 20 loop scans row I of the first given matrix: the column indices, if any, are stored in JC at line 11 and the row index is stored in IX at line 13, thus turning “on” the corresponding switch. The DO 40 loop runs over row I of the second matrix. For each column index J, defined at line 18, the multiple switch is tested at line 19: if the value of IX(J) is I, then the switch is on, which means that J has already been added to the list JC and should not be added again. Otherwise, J is added to JC at line 20. The reader may expect that the sentence IX(J)=I should appear between lines 21 and 22 in order to record the fact that the column index J has been added to the list JC. However, such a record is now not necessary because, during the processing of row I, the same value of J will never be found again: there are no repeated column indices in the representation of row I in the array JB.

7.11 Algorithm for the numerical addition of two sparse matrices with N rows

Input:	IA, JA, AN	first given matrix in RR(C)U.
	IB, JB, BN	second given matrix in RR(C)U.
	IC, JC	structure of the resulting matrix in RR(C)U.
	N	number of rows of the matrices.
Output:	CN	numerical values of the nonzeros of the resulting matrix.
Workspace:	X	expanded array used to accumulate the nonzeros; the dimension of X is M, the number of columns of the matrices.

```

1.      DO 70 I=1,N
2.      IH=I+1
3.      ICA=IC(I)
4.      ICB=IC(IH)-1
5.      IF(ICB.LT.ICA)GO TO 70
6.      DO 10 IP=ICA,ICB
7.  10   X(JC(IP))=0.
8.      IAA=IA(I)
9.      IAB=IA(IH)-1
10.     IF(IAB.LT.IAA)GO TO 30
11.     DO 20 IP=IAA,IAB
12.  20   X(JA(IP))=AN(IP)

```

- 213 Sherman, A. H. (1975). On the efficient solution of sparse systems of linear and nonlinear equations. Ph.D. Thesis, Department of Computer Science, Yale University, New Haven, CT. Report No. 46.
- 214 Shirey, R. W. (1969). "Implementation and analysis of efficient graph planarity testing algorithms." Ph.D. Dissertation, University of Wisconsin, Madison, WI.
- 215 Silvester, P. P. (1980). Software engineering aspects of finite elements. In Chari and Silvester (1980) ³⁰, pp. 69-85.
- 216 Skeel, R. D. (1981). Effect of equilibration on residual size for partial pivoting. *SIAM J. Numer. Anal.* **18**, 449-454.
- 217 Smith, B. T., Boyle, J. M., Dongarra, J. J., Garbow, B. S., Ikebe, Y., Klema, V. C. and Moler, C. B. (1976). *Matrix Eigensystem Routines – EISPACK Guide*. Lecture Notes in Computer Science, Vol. 6, 2nd. edn. Springer-Verlag: Berlin.
- 218 Speelpenning, B. (1978). "The generalized element method." Department of Computer Science, University of Illinois, Urbana, Champaign, Il. Report UIUCDCS-R-78.
- 219 Stewart, G. W. (1973). *Introduction to Matrix Computations*. Academic Press; New York.
- 220 Stewart, G. W. (1974). Modifying pivot elements in Gaussian elimination. *Math. Comput.* **28**, 537-542.
- 221 Stewart, G. W. (1976a). A bibliographical tour of the large sparse generalized eigenvalue problem. In Bunch and Rose (1976) ²⁸, pp. 113-130.
- 222 Stewart, G. W. (1976b). Simultaneous iteration for computing invariant subspaces of non-hermitian matrices. *Numer. Math.* **25**, 123-136.
- 223 Stewart, W. J. and Jennings, A. (1981). Algorithm 570.LOPSI: A simultaneous iteration algorithm for real matrices. *ACM Trans. Math. Software* **7**, 230-232.
- 224 Strassen, V. (1969). Gaussian elimination is not optimal. *Numer. Math.* **13**, 354-356.
- 225 Swift, G. (1960). A comment on matrix inversion by partitioning. *SIAM Rev.* **2**, 132-133.
- 226 Szyld, D. B. and Vishnepolsky, O. (1982). "Some operations on sparse matrices." Institute for Economic Analysis, New York University. Private communication.
- 227 Takahashi, H. and Natori, M. (1972). Eigenvalue problem of large sparse matrices. *Rep. Comput. Cent. Univ. Tokyo* **4**, 129-148.
- 228 Tarjan, R. E. (1971). "An efficient planarity algorithm." Computer Science Department, Stanford University, Stanford, CA. Technical Report 244.
- 229 Tarjan, R. E. (1972). Depth first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146-160.
- 230 Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *J. Assoc. Comput. Mach.* **22**, 215-225.
- 231 Tarjan, R. E. (1976). Graph theory and Gaussian elimination. In Bunch and Rose (1976) ²⁸, pp. 3-22.
- 232 Tewarson, R. P. (1967). On the product form of inverses of sparse matrices and graph theory. *SIAM Rev.* **9**, 91-99.
- 233 Tewarson, R. P. (1967). Row-column permutation of sparse matrices. *Comput. J.* **10**, 300-305.

Index

- n*-dimensional space, 185
- Accidental cancellation, 58, 69, 85, 143
- Accumulator
 - expanded, 30, 230, 234, 244
- Active column, 15
- Active elements, 78
- Active submatrix, 64
- Acyclic digraph, 150
 - paths in, 157
- Addition
 - of sparse matrices, 219
 - of sparse matrices, numerical algorithm., 223
 - of sparse matrices, symbolic algorithm., 222
 - of sparse vectors, 30, 32
- Adjacency level structure, 93
 - directed, 155
- Adjacency matrix, 12
- Adjacency structure of a graph, 10
- Adjacent set, 89, 94
- Adjacent vertex, 10
- Algebra for sparse matrices, 211
- Algebraic equations
 - linear, 35
- Algolw, 8
- Algorithm
 - conjugate gradient, 15
 - Lanczos, 15
- Almost-planar graph, 92, 122
- Amalgamation of cliques, 103
- Ancestor, 90
- Arc of tree, 93, 133, 134
 - redundant, 154
- Array, 5
 - data structure, 8
 - storage, 5
 - switch, 29
- Assembly
 - nodal, 249
- Assembly of element matrices, 127, 258
 - example, 257
 - numerical algorithm, 261
 - symbolic algorithm, 259
- Assignment
 - maximum, 148
- Augmented system, 132
- Augmenting path, 159, 161
 - shortest, 163
- Backsubstitution, 57
- Backtracking, 133, 151
- Backward error analysis, 67
- Backward substitution, 48, 56
 - example, 245
- Balancing, 84
- Band
 - generalized eigenproblem, reduction, 188
 - Lanczos algorithm, 206
 - matrix, 12
 - of a matrix, 12
 - variable, 14
- Band matrix
 - band, 12
 - bandwidth, 12
 - diagonal storage, 12, 13
 - half-bandwidth, 12

- semiband, 13
- Bandwidth, 12
 - reduction algorithm, 96
- Basis, 194
 - orthonormal, 194
- Bidirectional linked list, 7
- Bigraph, 146, 148
- Biorthogonalization Lanczos algorithm, 210
- Bipartite graph, 146, 148, 163
- Bireducible matrix, 145
- Bisection method, 184
- Block-partitioned matrix
 - storage of, 24
- Block-within-block-column storage, 25
- Block diagonal pivoting, 82
- Block Lanczos algorithm, 206
- Block lower triangular form, 144, 152, 157
- Block matrix, 110
- Block methods, 37
- Boundary conditions
 - for scalar problems, 251
 - for vector problems, 252
- Boundary element method, 249
- Bounds for eigenvalues, 182
- Boy, 148, 164
- Breadth-first search, 93
 - algorithm, 94
 - of a digraph, 155
 - of undirected graph, 93
- Bunch's error formulation, 67

- Cancellation
 - accidental, 69, 85
- Cardinality, 87
- Cell, 6
- Characteristic polynomial, 183
- Chebyshev acceleration, 198
- Checking a sparse representation, 276
- Cholesky factorization, 38, 53
 - implementation, 55
- Circular linked list, 7
- Circular storage, 17
- Clique, 12, 89
- Clique amalgamation, 103
- Codiagonal, 13
- Collapsing of vertex, 167
- Column
 - active, 15
 - equilibrated matrix, 83
 - graph, 148
 - head, 16
 - permutation, 217
- Column-wise representation, 20
 - complete, ordered, 21
- Column heads, 16
- Compact storage, 8, 30
- Compatible numbering, 118
- Complete pivoting, 78–80
- Complete representation, 20
- Component partitioning, 90
- Composite vertex, 91, 167
- Compression
 - Sherman's, 22
- Condensation, 150, 256
- Congruence transformation, 178
- Congruent pencils, 179
- Conjugate gradient algorithm, 15
- Connected component, 90
- Connected digraph, 147
- Connected graph, 89
- Connected nodes, 251
- Connection table, 11
- Connectivity level structure, 149
- Connectivity matrix, 12, 249, 256
- Consistent numbering, 118
- Control of numerical errors, 77
- Convenient ordering, 87
- Coordinate over-relaxation, 181
- Coordinate relaxation, 181
- Copying a sparse matrix, 279

- Cost considerations, 57
- CR(C)O, 21
- Cross-link, 93, 154, 155
- Cuthill and McKee algorithm, 96
 - reverse, 98
- Cutvertex, 90
- Cycle, 89
 - directed, 147

- Data structure, 26
 - array, 8
 - dynamic, 27
 - record, 8
 - static, 26
- Data structure set-up, 26
- Definitions and properties, 37
- Deflation, 190
- Degree, 11, 89
- Depth-first search, 93, 132
 - of a digraph, 151
 - of undirected graph, 132
- Descendant, 90
- Desirable forms, 27
- Determinant, 183
- Diagonal elementary matrix, 40
- Diagonally dominant matrix, 39
 - properly, 39
- Diagonal pivoting, 79
- Diagonal storage of band matrix, 12, 13
- Diakoptical system, 131
- Diameter of graph, 89
- Difference
 - symmetric, 162
- Digraph, 10, 87, 145
 - breadth-first search, 155
 - depth-first search, 151
- Dimension, 194
- Directed adjacency level structure, 155
- Directed adjacency level substructure, 156
- Directed cycle, 147

- Directed graph, 10, 87, 145
- Direct iteration, 190
 - for generalized eigenproblem, 193
- Dirichlet nodes, 251
- Disconnected graph, 89
- Disjoint list
 - multiple, 9
- Displaying a sparse matrix, 277
- Dissection
 - generalized nested, 121
 - height, 114
 - nested, 113
- Distance, 89, 147
- Dot product, 194
 - of sparse vectors, 33
- Dual structure, 150
- Dynamic storage, 26
 - allocation, 27
- Dynamic storage allocation, 27
- Dynamic structure, 27

- Eccentricity, 89
- Edge, 10, 87
- Eigenanalysis, 177
 - of hermitian matrix, 209
 - of Hessenberg matrix, 189
 - of tridiagonal matrix, 189
 - of tridiagonal unsymmetric matrix, 189
- Eigenpair, 177
- Eigenproblem
 - generalized, 177
 - standard, 177
 - unsymmetric, 210
- Eigenvalue, 177
 - economizer, 256
- Eigenvalues
 - bounds for, 182
- Eigenvector, 177
 - left-hand, 210
 - right-hand, 210

- Element
 - generalized, 130
 - super, 130
- Elementary matrices, 40
 - properties, 41
- Elementary matrix, 40
- Element matrix, 128
 - assembly, 128
- Elimination digraph, 147
- Elimination form
 - relation with product form, 52
- Elimination form of the inverse, 47, 52
- Elimination graph, 101
 - perfect, 136
- Elimination ordering
 - minimal, 138
 - minimum, 138
 - perfect, 136
- Embedding
 - planar, 88
- Entrance, 149
- Envelope, 14
- Envelope storage
 - of symmetric matrices, 14
- Equilibrated matrix, 83
- Equilibration, 83
- Error
 - estimator, 82
- Error bounds
 - for factorization, 74
 - for substitution, 76
- Error growth
 - monitoring, 82
- Errors
 - control, 77
 - in floating point operations, 65
 - in sparse factorization, 68
 - in sparse substitution, 73
- Estimator, 82
- Euclidean space, 193
- Exit, 149
- Expanded array of pointers, 33, 34
- Expanded real accumulator, 31
- Expanded storage, 9, 30
- Factorization
 - Cholesky, 38, 53
 - error bounds, 74
 - errors, 68
 - orthogonal, 37
 - triangular, 22, 35
- Fast Givens rotation, 186
- Fill, 26, 36
- Fill-in, 85
 - for nested dissection, 123
- Filled digraph, 147
- Filled graph, 103
- Finite element, 127
 - graph, 92
 - method, 92, 127, 249
 - ordering, 127
- Fixed length records, 16
- Floating point operations
 - errors, 65
- Flutter
 - aircraft analysis, 210
- Forest
 - spanning, 92, 152
- Format, sparse
 - column-wise, 20
 - row-wise, 19
- Forward and backward substitution
 - algorithm, 246
- Forward row-backward column storage, 18
- Forward substitution, 48, 56
 - example, 245
- Free vertex, 161
- FronD, 133, 154
- Front, 129
- Frontal method, 128

- for general matrix, 175
- Front of queue, 7
- Frontwidth, 15
- Fundamentals, 5

- Gauss-Jordan elimination
 - by columns, 50
 - by rows, 52
- Gauss-Seidel
 - algorithm, 275
 - method, 274
- Gauss elimination
 - and graph theory, 101
 - by columns, 45
 - by rows, 49
- Generalized eigenproblem, 177
- Generalized element, 130
- Generalized nested dissection, 121
- Gerschgorin
 - interval, 183
- Gerschgorin disc, 183
- Girl, 148, 164
- Givens rotation, 186
 - fast, 186
- Gram-Schmidt orthonormalization, 196, 206
- Graph, 87
 - adjacency structure, 10
 - bipartite, 146
 - connected, 89
 - directed, 10, 87
 - disconnected, 89
 - edge, 10
 - labelling, 10
 - planar, 88, 119
 - representation, 10
 - storage, 10
 - undirected, 10
 - vertex, 10
- Graphs
 - clique, 12

- Graph theory
 - and Gauss elimination, 101
 - for symmetric matrices, 87
 - for unsymmetric matrices, 146–148

- Hölder's inequality, 65
- Half-bandwidth, 12
- Hall's algorithm, 158
- Head, 6
 - of column, 16
 - of row, 16
- Height of dissection, 114
- Hermitian matrix, 40
 - eigenanalysis of, 209
- Hessenberg form, 37
- Hessenberg matrix, 185
 - eigenanalysis of, 189
- Hopcroft and Karp's algorithm, 161
- Householder's reflection, 186
 - matrix, 186
- Hypermatrix storage, 25

- Implicit storage, 24
- Incident edge, 89
- Indefinite matrix, 37
- Indefinite symmetric system, ordering, 140
- Indegree, 146
- Indistinguishable vertices, 107
- Inertia theorem, 184
- Infinite element, 127
- Inner product, 193
- Integer array of pointers, 32, 34
- Integers
 - merging of lists, 28
 - storage of lists, 8
- Interval of Gerschgorin, 183
- Introduction, 1, 5, 63
- Invariance of a graph, 88, 145, 148
- Invariant subspace, 194
- Inverse iteration, 190
 - for generalized eigenproblem, 193

- simultaneous, 198
- Isostructural rows, 24
- Jacobi rotation, 186
- Jenning's storage, 14
- Jungle, 156
- Key's storage, 20
- King's algorithm, 98
 - reverse, 101
- Knuth-Rheinboldt-Mesztenyi storage, 17
- Knuth sparse storage scheme, 16
- Knuth storage
 - column heads, 16
 - row heads, 16
- KRM circular storage, 17
- Krylov subspace, 199
- Label, 10
- Labelled graph, 10, 87
- Laguerre's iteration, 183
- Lanczos' basis, 199
- Lanczos algorithm, 15, 199
 - band, 206
 - biorthogonalization, 210
 - block, 206
 - block, for generalized eigenproblems, 208
 - for generalized eigenproblem, 203
 - in practice, 203
 - with no orthogonalization, 205
 - with periodic orthogonalization, 206
 - with reorthogonalization, 204
 - with selective orthogonalization, 204
- Left-hand eigenvector, 210
- Left row elementary matrix, 41
- Length
 - of level structure, 93, 155
 - of partitioning, 150
- Level structure, 90, 93
 - ajacency, 93
 - connectivity of, 149
 - directed adjacency, 155, 156
 - length of, 93
 - rooted, 93
 - width, 93
- Lexicographic search, 136
 - algorithm, 138
- Linear algebraic equations, 35
- Line of a matrix, 18
- Linked list, 6
 - bidirectional, 7
 - circular, 7
- Linked sparse storage schemes, 15
- Linked storage, 15
- List
 - head, 6
 - linked, 6
 - merging, 28
 - operations with, 5, 9
 - range, 8
 - sparse, 8
 - storage of, 5
 - terminator, 6
- Lists of integers, 8
 - compact storage, 8
 - expanded storage, 9
 - merging, 28
 - multiple disjoint, 9
 - range, 8
 - sparse, 8
 - storage, 8
- Long frond ordering, 134
- Loop, 88
- Lower column elementary matrix, 40
- Lower triangular matrix, 22, 40
 - block form, 144
- Lowlink, 168
- Machine precision, 65
- Markowitz's algorithm, 172
- Markowitz's pivoting, 172

- Matching, 148, 161
- Matrices
- elementary, 40
 - triangular, 40
- Matrix
- adjacency, 12
 - bireducible, 145
 - connectivity, 12
 - diagonally dominant, 39
 - elementary, 40
 - hermitian, 40
 - indefinite, 37
 - lower triangular, 22, 40
 - minor, 38
 - nondefinite, 37
 - nullity, 39
 - of rank one, 39
 - orthogonal, 38
 - permutation, 44
 - positive definite, 37
 - principal minor, 38
 - properly diagonally dominant, 39
 - rank, 39
 - rank deficiency, 39
 - singular, 39
 - structure, 16
 - triangular factorization, 38
 - unit diagonal, 40
 - unsymmetric, 37
 - upper triangular, 22, 40
 - zero-nonzero pattern, 16
- Maximal set, 157
- Maximum assignment, 148
- Maximum set, 157
- Merging sparse lists of integers, 28
- Mesh, 127
- Mesh generation, 250
- Minimal elimination ordering, 138
- Minimal nested dissection partitioning, 115
- Minimal separator, 90
- Minimization of trace, 208
- Minimum degree algorithm, 104
- Minimum elimination ordering, 138
- Minor, 38
- principal, 38
- Minrow-within-mincolumn pivoting, 174
- Modularity, 27
- Monitoring error growth, 82
- Monotone ordering, 90
- Multi-frontal method, 130
- Multiple disjoint lists, 9
- Multiple switch technique, 29
- Multiplication
- of diagonal matrix by matrix, 279
 - of general matrix by vector, 224
 - of sparse matrices, 230
 - of symmetric matrix by vector, 228, 229
 - of triangular matrix by matrix, 268–270
 - of triangular matrix by vector, 272
- Multiplicity, 177
- Nested dissection, 113
- algorithm, 116
 - fill-in, 123
 - generalized, 121
 - ordering, 113
 - properties, 118
 - tree, 114
- Nodal assembly matrix, 249, 251, 257
- Nodes, 127
- connected, 251
 - Dirichlet, 251
- Nonzero, 69, 85
- definition, 69
- Norm
- of matrix, 65, 72
 - of vector, 64
- Nullity, 39
- symbolic, 159
- Nullity of a matrix, 39

- Numbered graph, 87
- Numbering
 - compatible, 118
 - consistent, 118
- Numerical assembly algorithm, 261
- Numerical errors, 63, 65, *see also* Errors
- Numerical examples, 59
- Numerical processing, 26
- Numerical stability and pivot selection, 78–82
- Numerical triangular factorization algorithm, 242–245
 - in row-wise format, 238–239
- Offspring, 90
- One-way dissection, 122–126
 - algorithm, 125–126
- Operations with lists, 5
- Operations with sparse matrices, 211
- Ordered graph, 87
- Ordered representation, 20
- Ordering
 - convenient, 87
 - for pivot selection, 45
 - monotone, 90
- Origin shift, 191
- Orthogonal factorization, 37
- Orthogonal matrix, 38
- Orthogonal similarity, 178
- Orthonormal basis, 194
- Orthonormalization
 - Gram-Schmidt, 206
- Outdegree, 146
- Overhead storage, 15
- Overrelaxation
 - coordinate, 181
 - parameter, 181
- Packed storage, 30
- Partial pivoting, 80
- Partitioned matrix
 - storage, 24–26
- Partitioning, 90
 - component, 90
 - tree, 91
- Pascal, 8
- Path, 89
 - augmenting, 159, 161
 - length, 89
 - shortest augmenting, 162
- Paths in an acyclic digraph, 157–158
 - algorithm for finding, 157
- Pattern
 - zero-nonzero, 16
- Pedigree, 90
- Pencil, 177
 - congruent, 179
- Perfect elimination graph, 136
- Perfect elimination ordering, 136
- Peripheral vertex, 89
- Permutation
 - of columns, 217
 - of rows, 217
- Permutation matrix, 44
 - storage, 45
- Phase counter, 30
- Pivot, 45
- Pivoting, 78
 - block diagonal, 82
 - complete, 79
 - diagonal, 79
 - for symmetric matrix, 86, 141
 - for unsymmetric band matrix, 175
 - for unsymmetric matrix, 172–175
 - partial, 80
 - threshold, 80
- Pivot selection, 45, 77
 - and numerical stability, 78–83
- Planar embedding, 88
- Planar graph, 88, 119
- Plane in n -dimensional space, 185
- Plane rotation, 186

- Pop
 - item on stack, 7
- Positive definite matrix, 37
- Power method, 190–191
 - for generalized eigenproblem, 193
- Practical Cholesky factorization, 55
- Preface**, xiii
- Primary candidates, 78
- Principal minor, 38
- Printing a sparse matrix, 277
- Processing
 - numerical, 26
 - symbolic, 26
- Product, *see* Multiplication
- Product form of the inverse, 51–53
- Profile, 14
 - reduction, 98–101
- Properties and definitions, 37
- Properties of elementary matrices, 41
- Properties of triangular matrices, 42–44
- Pseudoperipheral vertex, 89
 - algorithm for finding, 95
- Push
 - item on stack, 7
- Queue, 7
 - front, 7
 - rear, 7
 - storage, 5
- Quotient graph, 90
- Quotient tree, 91
 - algorithm, refined, 110
- Radix sort, simultaneous, 217
- Range of list, 8
- Rank
 - deficiency, 39
 - of a matrix, 39
 - symbolic, 159
- Rank one matrix, 39
- Rayleigh-Ritz procedure, 195
- Rayleigh matrix, 182, 195
- Rayleigh quotient, 180–182
 - iteration, 193
 - iteration, for generalized eigenproblem, 193
 - procedure, 195
 - procedure, for generalized eigenproblem, 195
- Reachability matrix, 146
- Reachable set, 90
- Reachable vertex, 90, 146
- Real accumulator
 - expanded, 31
- Rear of queue, 7
- Receiver, 146
- Record data structure, 8
- Records
 - fixed length, 16
- Records of variable length, 28
- Reducible matrix, 145
- Reduction
 - of band generalized eigenproblem, 188
 - of band matrix, 188–189
 - of bandwidth, 96–98
 - of general matrix, 185–187
 - of profile, 98–101
- Reflection of Householder, 186
- Reflector, 186
- Relaxation coordinate, 181
- Representation, 19–20
 - complete, 20
 - of graphs, 10–12
 - ordered, 20
 - ordering, 215–216, 218
 - transforming, 278
 - unordered, 20
- Residual, 75, 84
- Residual matrix, 200
- Residual vector, 181
- Restriction, 194
- Reverse
 - depth-first ordering, 134

- monotone ordering, 133
- Right-hand eigenvector, 210
- Right row elementary matrix, 41
- Ritz values, 195
- Ritz vector, 195
 - threshold, 204
- Rooted level structure, 93
- Rooted substructure, 155
- Rooted tree, 90
- Root of strong component, 153
- Rotation
 - fast Givens, 186
 - Givens, 186
 - Jacobi, 186
 - plane, 186
- Row
 - equilibrated matrix, 83
 - graph, 148
 - head, 16
- Row-wise format, 19–20
- Row-wise representation, 19
 - complete, unordered, 21
 - complete and ordered, 20
 - diagonal and upper, ordered, 21
 - upper, ordered, 22
- Row heads, 16
- Row permutation, 217–218
- Rows
 - isostructural, 24
- RR(C)O, 20
- RR(C)U, 21
- RR(DU)O, 21
- RR(U)O, 22
- Sargent and Westerberg’s algorithm, 167–168
- Scalar product, 194
 - of sparse vectors, 33
- Scaling, 83–84
- Search, 93
- Section graph, 88
- Selection of pivots, 45, 77
- Selfedge, 88
- Semiband, 13
- Separator, 90
 - minimal, 90
- Set
 - notation, 162
 - operations, 162
- Shape functions, 250
- Sherman’s compression, 22–24
- Shift of origin, 191
- Short frond ordering, 134
- Similarity, orthogonal, 178
- Simultaneous iteration, 196–199
 - for generalized eigenproblem, 198–199
 - inverse, 198
- Simultaneous radix sort, 217
- Singular matrix, 39
- Singular values, 208
- Sink, 146
- Skeleton, 92
- Skyline storage, 15
- Software for unsymmetric systems, 175
- Source, 146
- Space, 193
- Span, 90
- Spanning forest, 92, 152
- Spanning set, 194
- Spanning tree, 92
- Sparse list, 8
- Sparse matrix
 - algebra, 211–247
 - operations, 211
- Sparse representation
 - checking of, 276
- Sparse row-wise format, 19–20
- Sparse substitution
 - errors in, 73–77
- Sparse tableau, 132
- Sparse vectors

- addition, 30–33
- dot product, 33
- scalar product, 33–34
- Spectral factorization, 178
- Stability
 - numerical, 78–82
- Stack, 7
 - pop an item, 7
 - push an item, 7
 - storage, 5
 - top, 7
- Standard eigenproblem, 177
- Static data structure, 26
- Stiffness matrix, 251
- Stodola’s iteration, 190
- Storage, 5–28
 - allocation, 26
 - block-partitioned matrix, 24
 - block-within-block-column, 25
 - compact, 30
 - compressed, 22
 - connection table, 11
 - dynamic allocation, 27
 - dynamic schemes, 26–28
 - expanded, 30
 - forward row-backward column, 18
 - hypermatrix, 25
 - implicit, 24
 - Knuth-Rheinboldt-Mesztenyi, 17
 - Knuth scheme, 16
 - KRM, 17
 - linked sparse schemes, 15
 - of arrays, 5
 - of band matrices, 12
 - of graphs, 10
 - of integers, 8
 - of lists, 5
 - of lists of integers, 8
 - of queues, 5
 - of stacks, 5
 - of symmetric matrices, 14
 - overhead, 15
 - packed, 30
 - skyline, 15
 - sparse row-wise format, 19
 - supersparse, 26
 - variable band, 14
- Strong component, 145, 148–151, 167, 168
 - root, 153
- Strongly connected component, 148
- Strongly connected digraph, 148
- Structure of a matrix, 16, 20
- Sturm sequence, 184
- Subgraph, 88
- Subspace, 194
 - invariant, 190, 194
 - iteration, 196
 - Krylov, 199
- Substitution
 - backward, 48, 56–57
 - error bounds for, 76
 - errors in, 73–77
 - forward, 48, 56–57
- Substructure
 - rooted, 155
- Substructuring, 130
- Super-element, 130
- Supersparse storage, 26
- Switch, 29
 - array, 29
 - multiple, 29
 - phase counter, 30
- Switch array, 29
- Switch technique
 - multiple, 29–30
- Sylvester’s inertia theorem, 184
- Symbolic
 - nullity, 159
- Symbolic assembly algorithm, 259–260
- Symbolic processing, 26

- Symbolic rank, [159](#)
- Symbolic section, [26](#)
- Symbolic singular matrix, [159](#)
- Symbolic triangular factorization
 - algorithm, [240–242](#)
 - in row-wise format, [235](#)
- Symmetric
 - difference, [162](#)
- Symmetric indefinite system
 - ordering, [140–141](#)
- Symmetric matrices
 - envelope storage, [14](#)
- Symmetric matrix, [37](#)
- System
 - augmented, [132](#)
 - diakoptical, [131](#)

- Table of factors, [43](#), [47](#), [51](#)
- Tarjan’s algorithm, [168–172](#)
- Terminal members, [114](#)
- Terminator, [6](#)
- Threshold pivoting, [80](#)
- Threshold vector, [204](#)
- Tolerance, [80](#)
- Top of stack, [7](#)
- Trace, [183](#)
- Trace minimization, [208](#)
- Transforming a representation, [278](#)
- Transmitter, [146](#)
- Transportability, [212](#)
- Transposition of a matrix, [213–215](#)
 - algorithm, [215–216](#)
- Transversal, [18](#), [145](#), [146](#), [158](#)
- Tree, [90](#)
 - nested dissection, [114](#)
 - partitioning, [109–113](#)
 - quotient, [90](#)
 - rooted, [90](#)
 - spanning, [92](#)
- Tree arc, [93](#), [133](#), [152](#), [155](#)
 - redundant, [154](#)
- Tree partitioning, [91](#)
- Triangular factorization, [22](#), [35](#)
 - in row-wise format, [235–245](#)
 - numerical, [238–239](#)
 - numerical algorithm, [242–245](#)
 - symbolic algorithm, [240–242](#)
- Triangular matrices, [40](#)
 - properties, [42–44](#)
- Triangulated graph, [137](#)
- Triangulation, [137](#)
- Tridiagonal matrix, [185](#)
 - eigenanalysis of, [189](#)
 - eigenanalysis of unsymmetric, [189](#)
- Triple, [16](#)

- Undirected graph, [10](#), [87](#)
 - breadth-first search, [93–95](#)
 - depth-first search, [132–136](#)
- Uni-frontal method, [130](#)
- Unit diagonal matrix, [40](#)
- Unordered representation, [20–22](#)
- Unsymmetric band matrix pivoting, [175](#)
- Unsymmetric eigenproblem, [210](#)
- Unsymmetric matrix, [37](#)
 - graph theory, [146–148](#)
 - pivoting strategies for, [172–175](#)
- Unsymmetric system software, [175](#)
- Upper almost-triangular matrix, [185](#)
- Upper column elementary matrix, [41](#)
- Upper Hessenberg matrix, [185](#)
- Upper triangular matrix, [22](#), [40](#)

- Variable band, [14](#)
- Variable length records, [28](#)
- Vectors
 - addition, [30–32](#)
- Vertex, [10](#), [87](#)
 - adjacent, [10](#)
 - degree, [11](#)
- Vertex collapsing, [167](#)

Wavefront, [15](#)

Width

of level structure, [155](#)

Zero-nonzero pattern, [16](#)

Zlatev's pivoting, [81](#)

improved, [81](#)