Introduction to Part I

Linear Systems, Least Squares and Linear Programming

by J. H. WILKINSON

1. Introduction

Algorithms associated with linear systems may be roughly classified under three headings:

i) Solution of non-singular systems of linear equations, matrix inversion and determinant evaluation.

ii) Linear least squares problems and calculation of generalized inverses.

iii) Linear programming

though the last two overlap to some extent with the first.

Nearly all the methods we describe for solving $n \times n$ systems of equations start by performing some factorization of the matrix A of coefficients and as a by-product the determinant of A is readily available. By taking the n columns of the unit matrix as right-hand sides, the inverse of A can be computed, though the volume of work can be reduced somewhat by taking account of the special nature of these n columns.

It was decided that iterative techniques for the solution of linear algebraic equations should be excluded from this volume since they have mainly been developed in connexion with the solution of partial differential equations.

It will be appreciated that the selection of the most efficient algorithms may be to some extent machine dependent. This is usually true, for example, when storage considerations become relevant. In the algorithms presented in this book no reference is made to the use of a backing store such as a magnetic tape or disc. For this reason the storage requirements in practice may be different from what they appear to be. For example, if iterative refinement of the solution of a system of linear equations is required, a copy of the original system must be retained. In practice this could be held in the backing store.

While it is hoped that many users will wish to take advantage of the tested algorithms in exactly the form in which they are published, others will probably wish to adapt them to suit their own special requirements. The descriptions accompanying the algorithms should make this easier to do.

2. List of Procedures

To facilitate reference to the procedures they are listed below in alphabetical order, together with the chapters in which they are described. Sets of related procedures are grouped together. We also include a number of basic procedures associated with complex arithmetic, etc. in this list.

acc inverse, acc solve, I/2
bandet1, bansol1, I/6
bandet2, bansol2, I/6
cabs, cdiv, csqrt, II/13
cg, I/5
chobanddet, chobandsol, I/4
choldet1, cholsol1, cholinversion1, I/1 and I/2
choldet2, cholsol2, cholinversion2, I/1
compdet, compsol, cx acc solve, I/7
gjdef1, gjdef2, I/3
innerprod, cx innerprod, I/2 and I/7
least squares solution, I/8
lp, I/11
minfit, svd, I/10
ortho1, ortho2, ortholin1, ortholin2, I/9
svd, minfit, I/10
symdet, symsol, syminversion, I/1
unsym acc solve, unsymdet, unsymsol, I/7

3. Positive Definite Symmetric Matrices

The solution of an $n \times n$ system of linear equations with a positive definite symmetric matrix of coefficients is a simpler problem than that of a system with a matrix of general form or even with a non-positive definite symmetric matrix. Most of the algorithms we give for positive definite symmetric matrices are based on the Cholesky factorization $A = LL^T$, where L is a lower triangular matrix, or on the related factorization $A = LDL^T$ where L is unit lower triangular and D is a positive diagonal matrix. Such algorithms are numerically stable and very economical as regards the number of arithmetic operations. There is no significant difference between the effectiveness of the LL^T and LDL^T factorizations, but the former uses square roots and the latter does not. For some purposes the Cholesky factor L is specifically required.

3.1. Dense Positive Definite Matrices

a) Cholesky Factorizations

There are two main sets associated with the Cholesky factorization of a dense $n \times n$ positive definite matrix. They are *choldet1, cholsol1* and *cholinversion1*; and *choldet2, cholsol2* and *cholinversion2*. The second of these sets achieves economy of storage by working only with a linear array of $\frac{1}{2}n(n+1)$ elements consisting initially of the lower triangle of A which is later overwritten with L. This second set is marginally slower than the first, but should be used when storage considerations are paramount.

In each case *choldet* gives the LL^T factorization of A and computes the determinant as a by-product. After using *choldet*, systems of equations having A as the matrix of coefficients may be solved by using the corresponding *cholsol* and the inverse of A may be found by using the corresponding *cholinversion*.

Although the algorithms based on the Cholesky factorization are very accurate, indeed of optimal accuracy having regard to the precision of computation, the computed solutions or the computed inverse may not be of sufficient accuracy if the matrix A is ill-conditioned. The accuracy of the solutions or of the inverse can be improved while still using the original factorization of A. This is achieved in the algorithms *acc solve* and *acc inverse*. These algorithms enable us to determine by means of an iterative refinement process solutions or inverses which are "correct to working accuracy" provided A is not "singular to working accuracy". Iterative refinement ultimately gives a computed solution $\bar{x}$ such that

$$\|x - \bar{x}\|_\infty / \|x\|_\infty \leqq k \times macheps,$$

where *macheps* is the machine precision and k is a constant of the order of unity.

In order to achieve this high accuracy these algorithms employ a procedure *innerprod* which calculates to double precision the inner-product of two single precision vectors. It is assumed that this is done in machine code. For consistency alternative versions of *choldet1* and *cholsol1* are given in which all inner-products are accumulated in double precision. For computers on which the accumulation of inner-products is inefficient, the standard versions of *choldet1* and *cholsol1* may be used, but the accumulation of inner-products in the computation of the residuals is an *essential* feature of the iterative refinement process.

The procedures *acc solve* and *acc inverse* are of great value even when more accurate solutions are not required. If $\bar{x}^{(1)}$ is the original solution and $\bar{x}^{(2)}$ is the first improved solution, then $\bar{x}^{(2)} - \bar{x}^{(1)}$ usually gives a very good estimate of the effect of end figure changes in the data (i.e. end figure changes from the point of view of *machine* word length) and enables one to assess the significance of the solution when the given matrix A is subject to errors. On a computer which accumulates double precision inner-products efficiently this important information is obtained very economically as regards computing time. However, it is necessary to retain A and the right-hand sides, though this could be done on the backing store.

b) The LDL^T Factorization

The LDL^T factorization is used in the set of procedures *symdet, symsol* and *syminversion*. The factorization of A is performed in *symdet* and *det*(A) is produced as a by-product. After using *symdet*, systems of equations having A as the matrix of coefficients may be solved using *symsol* and the inverse of A may be found by using *syminversion*. It may be mentioned that algorithms analogous to *choldet2* etc. could be produced using the LDL^T decomposition and that versions of *acc solve* and *acc inverse* could be based on *symdet* and *symsol*. There is little to choose between corresponding algorithms based on the LL^T and LDL^T decompositions.

c) Gauss-Jordan Inversion

Two algorithms *gjdef1* and *gjdef2* are given for inverting a positive definite matrix *in situ*. The second of these economises in storage in much the same way as *choldet2*. The Gauss-Jordan algorithm should not be used for solving linear equations since it is less efficient than the *choldet-cholsol* combinations. For matrix inversion it provides an elegant algorithm of much the same accuracy as the *cholinversions*.

3.2. Positive Definite Band Matrices

Many physical problems give rise to systems of equations for which the matrix is positive definite and of band form. For such matrices the Cholesky factorization is very effective since it preserves this band form. The procedure *chobanddet* gives the Cholesky factors and the determinant of such a matrix, A, and subsequent use of *chobandsol* provides the solution of systems of equations having A as the matrix of coefficients. It is rare for the inverse of such a matrix to be required (the inverse will be full even when A is of band form) and hence no procedure *choband inverse* is provided. Iterative refinement of the solution could be provided by a procedure analogous to *acc solve*. An alternative method for the solution of banded equations is discussed in the next section.

3.3. Sparse Positive Definite Matrices of Special Form

Many problems in mathematical physics give rise to sparse positive definite matrices in which the non-zero elements a_{ij} are defined in a systematic way as functions of i and j. This is particularly true of systems arising from finite difference approximations to partial differential equations. If the matrix is sufficiently sparse and the functions of i and j are sufficiently simple the procedure *cg* based on the conjugate gradient algorithm may be the most efficient method of solution.

Although the conjugate gradient method is sometimes thought of as an iterative method it would, with exact computation, give the exact solution in a finite number of steps. It has therefore been included, though *true* iterative methods have been excluded.

The *cg* algorithm is optimal in respect of storage since the matrix A is, in general, not stored at all. All that is needed is an auxiliary procedure for determining $A\,x$ from a given vector x. When A is of very high order the use of the conjugate gradient method (or of one of the iterative methods not included in this volume) may be mandatory since it may not be possible to store A even if it is of narrow band form. This is a common situation in connexion with systems arising from partial differential equations. They give rise to narrow banded matrices but in addition most of the elements within the band itself are zero and usually the procedure for computing $A\,x$ is very simple.

It cannot be too strongly emphasized that the procedure *cg* is seldom, if ever, to be preferred to Cholesky type algorithms on the grounds of accuracy. The *cg* method is less competitive in terms of speed if a number of different right-hand sides are involved since each solution proceeds independently and r solutions involve r times as much work. With Cholesky type methods the factorization of A is required only once, however many right-hand sides are involved.

4. Non-Positive Definite Symmetric Matrices

There are no algorithms in this volume designed to take advantage of symmetry in the solution of dense symmetric matrices and they must be solved using the procedures described in later sections. The two sets of procedures *bandet1*, *bansol1* and *bandet2*, *bansol2* may be used to deal with symmetric band matrices

but they do not take advantage of symmetry and indeed this is destroyed during the factorization. Either of the procedures *bandet1, 2* may be used to factorize A and to find its determinant and subsequently *bansol1, 2* may be used to solve sets of equations having A as the matrix of coefficients. The first set of procedures is appreciably more efficient than the second, but *when A is symmetric, bandet2 also computes the number of positive eigenvalues.* It can therefore be used to find the number of eigenvalues greater than a given value p by working with $A - pI$.

5. Non-Hermitian Matrices

The solution of a system of equations having a dense non-Hermitian matrix of coefficients is a more severe problem than that for a symmetric positive definite system of equations. In particular the problem of scaling presents difficulties which have not been solved in an entirely satisfactory way. The procedures given here are in this respect something of a compromise between what is desirable and what is readily attainable.

5.1. Dense Matrices (Real and Complex)

The procedures depend on the LU factorization of the matrix A, where L is lower triangular and U is unit upper triangular. *Partial pivoting* is used in the factorization so that the algorithms effectively "correspond" to Gaussian elimination in which the pivotal element at each stage is chosen to be the largest element in the leading column of the remaining matrix, though this would give rise to a unit lower triangular L.

The procedure *unsymdet* produces the LU factorization of a real dense matrix A and produces the determinant as a by-product; subsequently any number of linear systems having A as the matrix of coefficients may be solved by using the procedure *unsymsol.* The procedures *compdet* and *compsol* perform the analogous operations for a complex matrix A.

To cover the cases when A is too ill-conditioned for the computed solution to be sufficiently accurate the procedures *unsym acc solve* and *cx acc solve* perform iterative refinement of the solution, using the initial factorization of A given by *unsymdet* or *compdet* respectively. Provided A is not almost singular to working accuracy iterative refinement will ultimately give a solution $\bar{x}$ which is correct to working accuracy, i.e. such that $\|x - \bar{x}\|_\infty / \|x\|_\infty \leqq k \times macheps$, where *macheps* is the machine precision and k is a constant of order unity. Notice that "small" components of x may well have a low relative accuracy. We stress once again that one step of iterative refinement is of great value in giving an estimate of the sensitivity of the system of equations. If $\bar{x}^{(1)}$ and $\bar{x}^{(2)}$ are the first two solutions, then $\bar{x}^{(2)} - \bar{x}^{(1)}$ usually gives a good estimate of the effect of end figure changes in the data. (See earlier comments in 3.1 (a).)

5.2. Unsymmetric Band Matrices

The procedures *bandet1* and *bansol1* are designed to deal with real unsymmetric matrices of band form. They deal efficiently with banded matrices having a different number of non-zero diagonal columns on the two sides. An

LU decomposition with partial pivoting is used. (They can be used for symmetric band matrices; see 4.) *bandet1* gives the factorization of A and its determinant and subsequently *bansol1* may be used to give solutions corresponding to any number of right-hand sides.

bandet2 and *bansol2* can also be used for unsymmetric matrices but are less efficient; they were designed primarily for use with symmetric band matrices (see 4.).

6. Least Squares and Related Problems

a) The Standard Least Squares Problem

There are effectively two procedures designed for the solution of the least squares problem associated with a real $m \times n$ matrix, A, such that $m \geqq n$ and $rank(A) = n$.

The first of these, *least squares solution*, is based on the determination of an orthogonal matrix Q such that $Q\tilde{A} = R$ where R is an $m \times n$ upper triangular matrix and $\tilde{A}$ is A with its columns suitably permuted as determined by a column pivoting strategy. The matrix Q is obtained as the product of n elementary orthogonal matrices of the type $I - 2ww^T$, where $\|w\|_2 = 1$. The vector $\tilde{x}$ such that $\|b - \tilde{A}\tilde{x}\|_2$ is a minimum is then determined from the equation $R\tilde{x} = Qb$. Any number of right-hand sides may be treated simultaneously. If the m columns of the $m \times m$ unit matrix are taken successively as right-hand sides the $n \times m$ matrix $(A^H A)^{-1} A^H$ is produced; this is the pseudo-inverse of A, the latter being assumed to be of rank n.

The procedure incorporates an iterative refinement process analogous to that described earlier for $n \times n$ systems of linear equations. For a compatible $m \times n$ system, i.e. a system for which $\min \|b - Ax\|_2 = 0$, iterative refinement leads to a solution which is "correct to working accuracy", but for an incompatible system the final accuracy may fall appreciably short of this. Least squares solutions are frequently required for systems which are far from compatible and for such systems iterative refinement serves little purpose.

The procedure can be used with $m = n$, in which case it gives the solution of a non-singular system of linear equations. It can therefore be used in the obvious way to perform matrix inversion. Although the orthogonal factorization is *slightly* superior to the LU factorization as regards numerical stability this superiority is hardly such as to justify the use of *least squares solution* in place of *unsymdet* and *unsymsol* when one is primarily interested in non-singular $n \times n$ systems.

The alternative to *least squares solution* is again based on an orthogonal triangularization of A, effectively using a variant of the Gram-Schmidt process. This gives an orthogonal basis for the columns of A and each right-hand side b is expressed in terms of this basis and a residual vector r which is orthogonal to it.

There is a set of four procedures based on this factorization. *ortholin1* gives the solution of the least squares problem $AX = B$ where B consists of k right-hand sides. *ortholin2* is provided for the special case when there is only one right-hand side, while *ortho1* gives the inverse of an $n \times n$ matrix. These three procedures are all provided with an iterative refinement section. The fourth procedure *ortho2* gives the inverse of an $n \times n$ matrix without iterative refinement and economises on storage by overwriting the inverse on the space originally holding A. The

speed and performance of these procedures is roughly comparable with that of *least squares solution*. In particular iterative refinement does not generally give solutions correct to working accuracy for incompatible systems of $m \times n$ equations.

b) The General Least Squares Problem and Pseudo-Inverses

When the $m \times n$ matrix A is of rank less than n, the least squares problem does not have a unique solution and one is often interested in the solution x of the least squares problem such that $\|x\|_2$ is a minimum. This solution is given by $x = A^+ b$ where A^+ is the pseudo-inverse of A. There are two procedures associated with this problem, both based on the *singular value decomposition* of the matrix A. For the case $m \geqq n$ we may express this decomposition in the form $A = U \Sigma V^T$ where $U^T U = V^T V = V V^T = I_n$ and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, the σ_i being the *singular values* of A. (U is an $m \times n$ matrix with orthogonal columns and V is an orthogonal matrix.) The corresponding expression for the pseudo-inverse is $V \Sigma^+ U^T$ where $\Sigma^+ = \mathrm{diag}(\sigma_1^+, \ldots, \sigma_n^+)$ and

$$\sigma_i^+ = \begin{cases} \sigma_i^{-1} & \text{if} \quad \sigma_i > 0 \\ 0 & \text{if} \quad \sigma_i = 0. \end{cases}$$

The singular value decomposition gives the most reliable determination of the "rank" of a matrix A, the elements of which are subject to errors. The computed singular values are always exact for some matrix which is equal to A almost to working accuracy and these values are of decisive importance in rank determination. Alternative simpler factorizations can in rare cases give an "incorrect" rank determination.

The procedure *svd* gives the set of singular values and the matrices U and V. The procedure *minfit* gives the solutions $A^+ b$ corresponding to each of p given right-hand sides b. The two procedures may be used to solve a variety of problems related to least squares problems and the calculation of the pseudo-inverse.

7. The Linear Programming Problem

Research in the field of linear programming has had surprisingly little contact with that in the main stream of numerical linear algebra, possibly for historical reasons. This is unfortunate since the two fields have a great deal in common. It is hoped that the inclusion of a linear programming procedure will serve to emphasize this fact and help to bring together the two groups of research workers.

In view of the fact that *lp* is the only procedure concerned with linear programming in this volume, it was felt that a fairly comprehensive treatment was warranted. *lp* may be used to solve the general problem:

Minimise

$$c_0 + c_{-m} x_{-m} + \cdots + c_{-1} x_{-1} + c_1 x_1 + \cdots + c_n x_n$$

subject to the relations

$$x_{-i} + \sum_{k=1}^{n} a_{ik} x_k = b_i, \qquad i = 1, 2, \ldots, m$$

$$x_i \geqq 0 \quad (i \in I^+), \qquad x_i = 0 \quad (i \in I^0),$$

where I^+ and I^0 are disjoint subsets of $\{-m, \ldots, -1, 1, \ldots, n\}$.

The procedure is a variant of the simplex method based on a triangular factorization of the current basis A_J of the form $LA_J = R$, where R is a non-singular upper triangular matrix. Numerical stability is preserved by using a variant of the exchange algorithm independently suggested by Golub and Stoer, and developed by Bartels.

It is recognized that linear programming procedures are very machine dependent. Since very large problems are encountered, the use of the backing store is often mandatory and it is also important to deal efficiently with sparse matrices. These last two problems are to some extent bypassed by the use of two subprocedures *ap* and *p*, the first of which deals with the handling of the matrix A and the second with operations on the matrices R and L and the various modified versions $\bar{B}$ of the right-hand sides. Specific versions of *ap* and *p* are presented to deal with the simple cases when the matrices $A, R, L, \bar{B}$ are stored as standard rectangular arrays.