Chapter F04

Simultaneous Linear Equations

Contents

1 Scope of the Chapter

This chapter is concerned with the solution of the matrix equation $AX = B$, where B may be a single vector or a matrix of multiple right-hand sides. The matrix A may be real, complex, symmetric, Hermitian, positive-definite, positive-definite Toeplitz or banded. It may also be rectangular, in which case a least-squares solution is obtained.

For a general introduction to sparse systems of equations, see the F11 Chapter Introduction, which currently provides routines for sparse symmetric systems. Some routines for sparse problems are also included in this chapter; they are described in Section 3.4.

2 Background to the Problems

A set of linear equations may be written in the form

$$Ax = b$$

where the known matrix A, with real or complex coefficients, is of size m by n, (m rows and n columns), the known right-hand vector b has m components (m rows and one column), and the required solution vector x has n components (n rows and one column). There may also be p vectors b_i, $i = 1, 2, \ldots, p$ on the right-hand side and the equations may then be written as

$$AX = B,$$

the required matrix X having as its p columns the solutions of $Ax_i = b_i$, $i = 1, 2, \ldots, p$. Some routines deal with the latter case, but for clarity only the case $p = 1$ is discussed here.

The most common problem, the determination of the unique solution of $Ax = b$, occurs when $m = n$ and A is not singular, that is $\mathrm{rank}(A) = n$. This is discussed in Section 2.1 below. The next most common problem, discussed in Section 2.2 below, is the determination of the least-squares solution of $Ax \simeq b$ required when $m > n$ and $\mathrm{rank}(A) = n$, i.e., the determination of the vector x which minimizes the norm of the residual vector $r = b - Ax$. All other cases are rank deficient, and they are treated in Section 2.3.

Most of the routines of the chapter are based on those published in the book edited by Wilkinson and Reinsch [3]. We are very grateful to the late Dr J H Wilkinson FRS for his help and interest during the implementation of this chapter of the Library. notintro="yes"

2.1 Unique Solution of $Ax = b$

Most routines in this chapter solve this particular problem. The computation starts with the triangular decomposition $A = PLU$, where L and U are respectively lower and upper triangular matrices and P is a permutation matrix, chosen so as to ensure that the decomposition is numerically stable. The solution is then obtained by solving in succession the simpler equations

$$\begin{aligned} Ly &= P^T b \\ Ux &= y \end{aligned}$$

the first by forward-substitution and the second by back-substitution.

If A is real symmetric and positive-definite, $U = L^T$, while if A is complex Hermitian and positive-definite, $U = L^H$; in both these cases P is the identity matrix (i.e., no permutations are necessary). In all other cases either U or L has unit diagonal elements.

Due to rounding errors the computed 'solution' x_0, say, is only an approximation to the true solution x. This approximation will sometimes be satisfactory, agreeing with x to several figures, but if the problem is ill-conditioned then x and x_0 may have few or even no figures in common, and at this stage there is no means of estimating the 'accuracy' of x_0.

There are three possible approaches to estimating the accuracy of a computed solution.

One way to do so, and to 'correct' x_0 when this is meaningful (see next paragraph), involves computing the residual vector $r = b - Ax_0$ in extended precision arithmetic, and obtaining a correction vector d by solving $PLUd = r$. The new approximate solution $x_0 + d$ is usually more accurate and the correcting process is repeated until (a) further corrections are negligible or (b) they show no further decrease.

It must be emphasised that the 'true' solution x may not be meaningful, that is correct to all figures quoted, if the elements of A and b are known with certainty only to say p figures, where p is smaller than the word-length of the computer. The first correction vector d will then give some useful information about the number of figures in the 'solution' which probably remain unchanged with respect to maximum possible uncertainties in the coefficients.

An alternative approach to assessing the accuracy of the solution is to compute or estimate the **condition number** of A, defined as

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

Roughly speaking, errors or uncertainties in A or b may be amplified in the solution by a factor $\kappa(A)$. Thus, for example, if the data in A and b are only accurate to 5 digits and $\kappa(A) \approx 10^3$, then the solution cannot be guaranteed to have more than 2 correct digits. If $\kappa(A) \geq 10^5$, the solution may have no meaningful digits.

To be more precise, suppose that

$$Ax = b \text{ and } (A + \delta A)(x + \delta x) = b + \delta b.$$

Here δA and δb represent perturbations to the matrices A and b which cause a perturbation δx in the solution. We can define measures of the relative sizes of the perturbations in A, b and x as

$$\rho_A = \frac{\|\delta A\|}{\|A\|}, \quad \rho_b = \frac{\|\delta b\|}{\|b\|} \text{ and } \rho_x = \frac{\|\delta x\|}{\|x\|} \text{ respectively.}$$

Then

$$\rho_x \leq \frac{\kappa(A)}{1 - \kappa(A)\rho_A}(\rho_A + \rho_b)$$

provided that $\kappa(A)\rho_A < 1$. Often $\kappa(A)\rho_A \ll 1$ and then the bound effectively simplifies to

$$\rho_x \leq \kappa(A)(\rho_A + \rho_b).$$

Hence, if we know $\kappa(A)$, ρ_A and ρ_b, we can compute a bound on the relative errors in the solution. Note that ρ_A, ρ_b and ρ_x are defined in terms of the norms of A, b and x. If A, b or x contains elements of widely differing magnitude, then ρ_A, ρ_b and ρ_x will be dominated by the errors in the larger elements, and ρ_x will give no information about the relative accuracy of smaller elements of x.

A third way to obtain useful information about the accuracy of a solution is to solve two sets of equations, one with the given coefficients, which are assumed to be known with certainty to p figures, and one with the coefficients rounded to $(p-1)$ figures, and to count the number of figures to which the two solutions agree. In ill-conditioned problems this can be surprisingly small and even zero.

2.2 The Least-squares Solution of $Ax \simeq b$, $m > n$, $\mathbf{rank}(A) = n$

The least-squares solution is the vector $\hat{x}$ which minimizes the sum of the squares of the residuals,

$$S = (b - A\hat{x})^T(b - A\hat{x}) = \|b - A\hat{x}\|_2^2.$$

The solution is obtained in two steps:

(i) Householder Transformations are used to reduce A to 'simpler form' via the equation $QA = R$, where R has the appearance

$$\begin{pmatrix} \hat{R} \\ \hline 0 \end{pmatrix}$$

with $\hat{R}$ a non-singular upper triangular n by n matrix and 0 a zero matrix of shape $(m-n)$ by n. Similar operations convert b to $Qb = c$, where

$$c = \begin{pmatrix} c_1 \\ \hline c_2 \end{pmatrix}$$

with c_1 having n rows and c_2 having $(m-n)$ rows.

(ii) The required least-squares solution is obtained by back-substitution in the equation

$$\hat{R}\hat{x} = c_1.$$

Again due to rounding errors the computed $\hat{x}_0$ is only an approximation to the required $\hat{x}$, but as in Section 2.1, this can be improved by 'iterative refinement'. The first correction d is the solution of the least-squares problem

$$Ad = b - A\hat{x}_0 = r$$

and since the matrix A is unchanged, this computation takes less time than that of the original $\hat{x}_0$. The process can be repeated until further corrections are (a) negligible or (b) show no further decrease.

2.3 Rank-deficient Cases

If, in the least-squares problem just discussed, $\text{rank}(A) < n$, then a least-squares solution exists but it is not unique. In this situation it is usual to ask for the least-squares solution 'of minimal length', i.e., the vector x which minimizes $\|x\|_2$, among all those x for which $\|b - Ax\|_2$ is a minimum.

This can be computed from the Singular Value Decomposition (SVD) of A, in which A is factorized as

$$A = QDP^T$$

where Q is an m by n matrix with orthonormal columns, P is an n by n orthogonal matrix and D is an n by n diagonal matrix. The diagonal elements of D are called the 'singular values' of A; they are non-negative and can be arranged in decreasing order of magnitude:

$$d_1 \geq d_2 \geq \ldots \geq d_n \geq 0.$$

The columns of Q and P are called respectively the left and right singular vectors of A. If the singular values $d_{r+1}, \ldots, d_n$ are zero or negligible, but d_r is not negligible, then the rank of A is taken to be r (see also Section 2.4) and the minimal length least-squares solution of $Ax \simeq b$ is given by

$$\hat{x} = D^{\dagger}Q^T b$$

where $D^{\dagger}$ is the diagonal matrix with diagonal elements $d_1^{-1}, d_2^{-1}, \ldots, d_r^{-1}, 0, \ldots, 0$.

The SVD may also be used to find solutions to the homogeneous system of equations $Ax = 0$, where A is m by n. Such solutions exist if and only if $\text{rank}(A) < n$, and are given by

$$x = \sum_{i=r+1}^{n} \alpha_i p_i$$

where the α_i are arbitrary numbers and the p_i are the columns of P which correspond to negligible elements of D.

The general solution to the rank-deficient least-squares problem is given by $\hat{x} + x$, where $\hat{x}$ is the minimal length least-squares solution and x is any solution of the homogeneous system of equations $Ax = 0$.

2.4 The Rank of a Matrix

In theory the rank is r if $n - r$ elements of the diagonal matrix D of the singular value decomposition are exactly zero. In practice, due to rounding and/or experimental errors, some of these elements have very small values which usually can and should be treated as zero.

For example, the following 5 by 8 matrix has rank 3 in exact arithmetic

$$\begin{pmatrix} 22 & 14 & -1 & -3 & 9 & 9 & 2 & 4 \\ 10 & 7 & 13 & -2 & 8 & 1 & -6 & 5 \\ 2 & 10 & -1 & 13 & 1 & -7 & 6 & 0 \\ 3 & 0 & -11 & -2 & -2 & 5 & 5 & -2 \\ 7 & 8 & 3 & 4 & 4 & -1 & 1 & 2 \end{pmatrix}.$$

On a computer with 7 decimal digits of precision the computed singular values were:

$$3.5 \times 10^1, \quad 2.0 \times 10^1, \quad 2.0 \times 10^1, \quad 1.3 \times 10^{-6}, \quad 5.5 \times 10^{-7}$$

and the rank would be correctly taken to be 3.

It is not, however, always certain that small computed singular values are really zero. With the 7 by 7 Hilbert matrix, for example, where $a_{ij} = 1/(i+j-1)$, the singular values are

$$1.7, \quad 2.7 \times 10^{-1}, \quad 2.1 \times 10^{-2}, \quad 1.0 \times 10^{-3}, \quad 2.9 \times 10^{-5}, \quad 4.9 \times 10^{-7}, \quad 3.5 \times 10^{-9}.$$

Here there is no clear cut-off between small (i.e., negligible) singular values and larger ones. In fact, in exact arithmetic, the matrix is known to have full rank and none of its singular values is zero. On a computer with 7 decimal digits of precision, the matrix is effectively singular, but should its rank be taken to be 6, or 5, or 4?

It is therefore impossible to give an infallible rule, but generally the rank can be taken to be the number of singular values which are neither zero nor very small compared with other singular values. For example, if there is a sharp decrease in singular values from numbers of order unity to numbers of order 10^{-7}, then the latter will almost certainly be zero in a machine in which 7 significant decimal figures is the maximum accuracy. Similarly for a least-squares problem in which the data is known to about four significant figures and the largest singular value is of order unity then a singular value of order 10^{-4} or less should almost certainly be regarded as zero.

It should be emphasised that rank determination and least-squares solutions can be sensitive to the scaling of the matrix. If at all possible the units of measurement should be chosen so that the elements of the matrix have data errors of approximately equal magnitude.

2.5 Generalized Linear Least Squares Problems

The simple type of linear least-squares problem described in Section 2.2 can be generalized in various ways.

(1) linear least-squares problems with **equality constraints**:

$$\text{find } x \text{ to minimize } \quad S = \|c - Ax\|_2^2 \quad \text{subject to } \quad Bx = d,$$

where A is m by n and B is p by n, with $p \leq n \leq m + p$. The equations $Bx = d$ may be regarded as a set of equality constraints on the problem of minimizing S. Alternatively the problem may be regarded as solving an overdetermined system of equations

$$\begin{pmatrix} A \\ B \end{pmatrix} x = \begin{pmatrix} c \\ d \end{pmatrix},$$

where some of the equations (those involving B) are to be solved exactly, and the others (those involving A) are to be solved in a least-squares sense. The problem has a unique solution on the assumptions that B has full row rank p and the matrix $\begin{pmatrix} A \\ B \end{pmatrix}$ has full column rank n. (For linear least-squares problems with **inequality constraints**, refer to Chapter E04.)

(2) **general Gauss–Markov linear model problems**:

$$\text{minimize } \quad \|y\|_2 \quad \text{subject to } \quad d = Ax + By,$$

where A is m by n and B is m by p, with $n \leq m \leq n + p$. When $B = I$, the problem reduces to an ordinary linear least-squares problem. When B is square and nonsingular, it is equivalent to a **weighted linear least-squares problem**:

$$\text{find } \; x \; \text{ to minimize } \quad \|B^{-1}(d - Ax)\|_2.$$

The problem has a unique solution on the assumptions that A has full column rank n, and the matrix (A, B) has full row rank m.

2.6 Calculating the Inverse of a Matrix

The routines in this chapter can also be used to calculate the inverse of a square matrix A by solving the equation

$$AX = I$$

where I is the identity matrix. However, solving the equations $AX = B$ by calculation of the inverse of the coefficient matrix A, i.e., by $X = A^{-1}B$, is **definitely not recommended**.

Similar remarks apply to the calculation of the pseudo inverse of a singular or rectangular matrix.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Black Box and General Purpose Routines

Most of the routines in this chapter are categorised as Black Box routines or General Purpose routines.

Black Box routines solve the equations $Ax_i = b_i$, $i = 1, 2, \ldots, p$ in a single call with the matrix A and the right-hand sides b_i being supplied as data. These are the simplest routines to use and are suitable when all the right-hand sides are known in advance and do not occupy too much storage.

General Purpose routines, in general, require a previous call to a routine in Chapter F01 or Chapter F03 to factorize the matrix A. This factorization can then be used repeatedly to solve the equations for one or more right-hand sides which may be generated in the course of the computation. The Black Box routines simply call a factorization routine and then a general purpose routine to solve the equations.

The two routines F04MBF and F04QAF which use an iterative method for sparse systems of equations do not fit easily into this categorisation, but are classified as general purpose routines in the decision trees and indexes.

3.2 Systems of Linear Equations

Most of the routines in this chapter solve linear equations $Ax = b$ when A is n by n and a unique solution is expected (case 2.1). If this turns out to be untrue the routines go to a failure exit. The matrix A may be 'general' real or complex, or may have special structure or properties, e.g. it may be banded, tridiagonal, almost block-diagonal, sparse, symmetric, Hermitian, positive-definite (or various combinations of these).

It must be emphasised that it is a waste of computer time and space to use an inappropriate routine, for example one for the complex case when the equations are real. It is also unsatisfactory to use the special routines for a positive-definite matrix if this property is not known in advance.

Routines are given for calculating the **approximate solution**, that is solving the linear equations just once, and also for obtaining the **accurate solution** by successive iterative corrections of this first approximation, as described in Section 2.1. The latter, of course, are more costly in terms of time and storage, since each correction involves the solution of n sets of linear equations and since the original A and its LU decomposition must be stored together with the first and successively corrected approximations to the solution. In practice the storage requirements for the 'corrected' routines are about double those of the 'approximate' routines, though the extra computer time is not prohibitive since the same matrix and the same LU decomposition is used in every linear equation solution.

Two routines are provided – F04YCF for real matrices, F04ZCF for complex matrices – which can return a cheap but reliable estimate of $\|A^{-1}\|$, and hence an estimate of the **condition number** $\kappa(A)$ (see Section 2.1). These routines can be used in conjunction with most of the linear equation solving routines in this chapter: further advice is given in the routine documents.

Other routines for solving linear equation systems, computing inverse matrices, and estimating condition numbers can be found in Chapter F07, which contains LAPACK software.

3.3 Linear Least-squares Problems

For case 2.2, when $m \geq n$ and a unique least-squares solution is expected, there are two routines for a general real A, one of which (F04JGF) computes a first approximation and the other (F04AMF) computes iterative corrections. If it transpires that $\text{rank}(A) < n$, so that the least-squares solution is not unique, then F04AMF takes a failure exit, but F04JGF proceeds to compute the **minimal length** solution by using the SVD (see below).

If A is expected to be of less than full rank then one of the routines for calculating the minimal length solution may be used. Currently these routines are only for the 'approximate' solution. These routines determine the rank based upon a user-supplied tolerance to decide which elements are negligible, routines based upon the SVD providing the most reliable guide.

For $m \gg n$ the use of the SVD is not significantly more expensive than the use of routines based upon the QR factorization.

If A is complex and rank$(A) = n$, the problem can be solved by calling F04KMF with $p = 0$ (dummy arrays of dimension 1 must be supplied for the parameters B and D). If A is expected to be of less than full rank, the problem can be solved by calls to F02XEF (to compute the SVD of A) and F06SAF (CGEMV/ZGEMV).

Problems with linear **equality constraints** can be solved by F04JLF (for real data) or by F04KLF (for complex data), provided that the problems are of full rank. Problems with linear **inequality constraints** can be solved by E04NCF in Chapter E04.

General Gauss–Markov linear model problems, as formulated in Section 2.5, can be solved by F04JMF (for real data) or by F04KMF (for complex data).

3.4 Sparse Matrix Routines

Routines specifically for real sparse matrices should be used only when the number of non-zero elements is very small, less than, say, 10% of the n^2 elements of A, and the matrix does not have a relatively small band width.

Chapter F11 contains routines for the iterative solution of real sparse symmetric linear systems, as well as a routine F11JBF which may be used for direct solution of real sparse symmetric positive-definite problems. There are two routines in Chapter F04 for solving sparse linear equations (F04AXF and F04QAF). F04AXF utilizes a factorization of the matrix A obtained from F01BRF or F01BSF, while F04QAF uses an iterative technique and requires a user-supplied function to compute matrix-vector products Ac and $A^T c$ for any given vector c. F04AXF can be utilised to solve for several right-hand sides, but the original matrix has to be explicitly supplied and is overwritten by the factorization, and the storage requirements will usually be substantially more than those of the iterative routines.

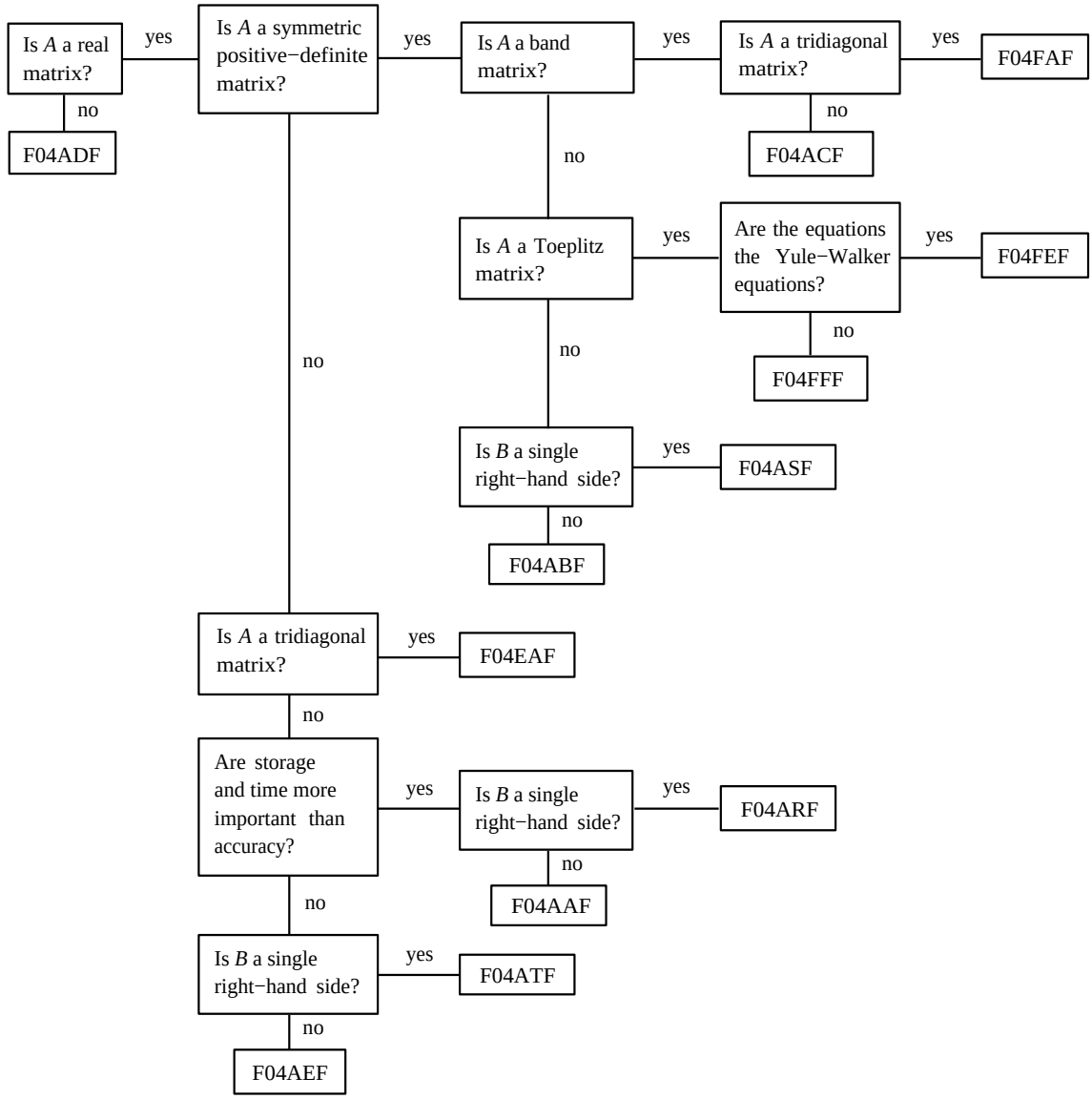
F04QAF solves sparse least-squares problems by an iterative technique, and also allows the solution of damped (regularised) least-squares problems (see the routine document for details).

4 Decision Trees

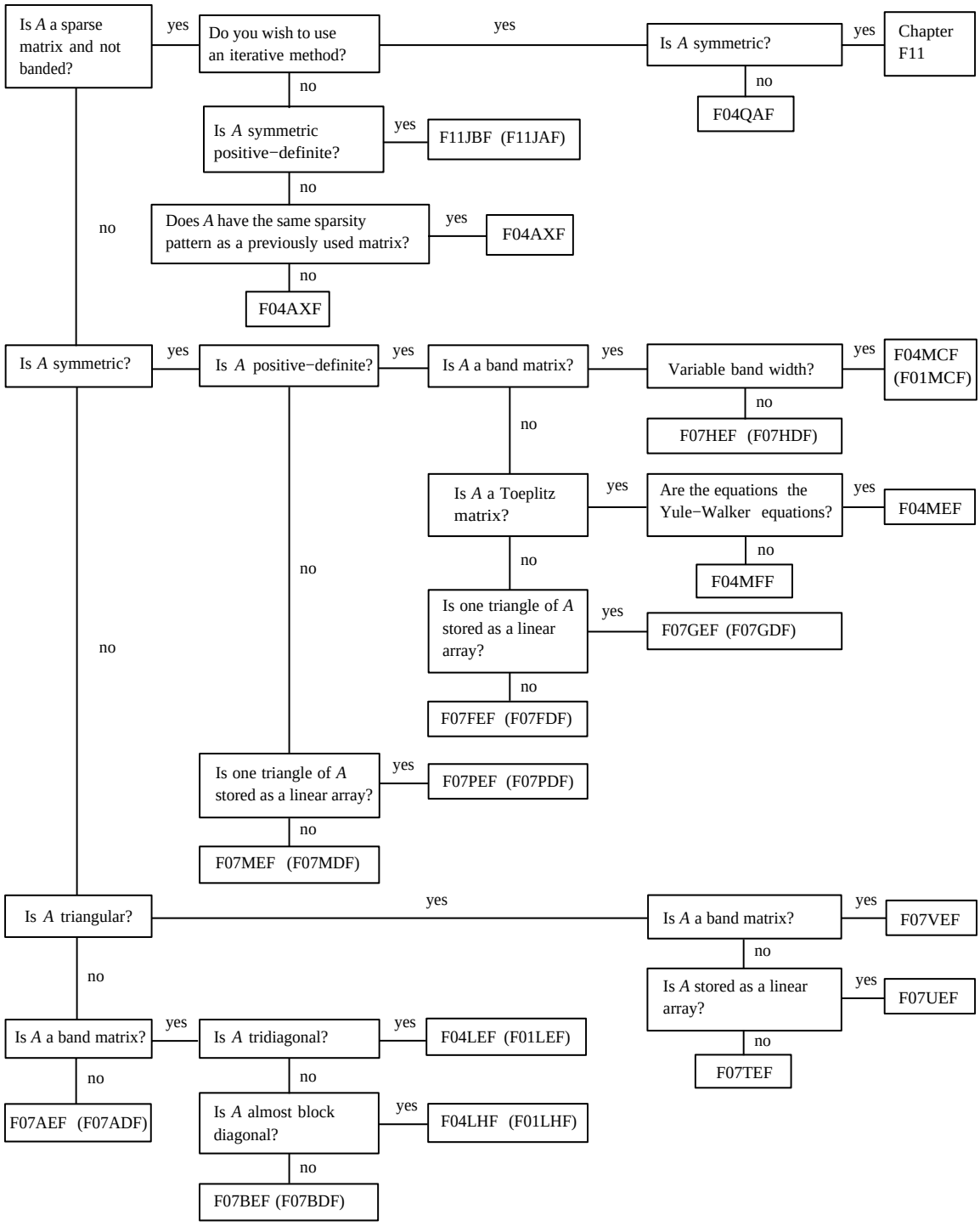
If at any stage the answer to a question is 'Don't know' this should be read as 'No'.

The name of the routine (if any) that should be used to factorize the matrix A is given in brackets after the name of the routine for solving the equations.

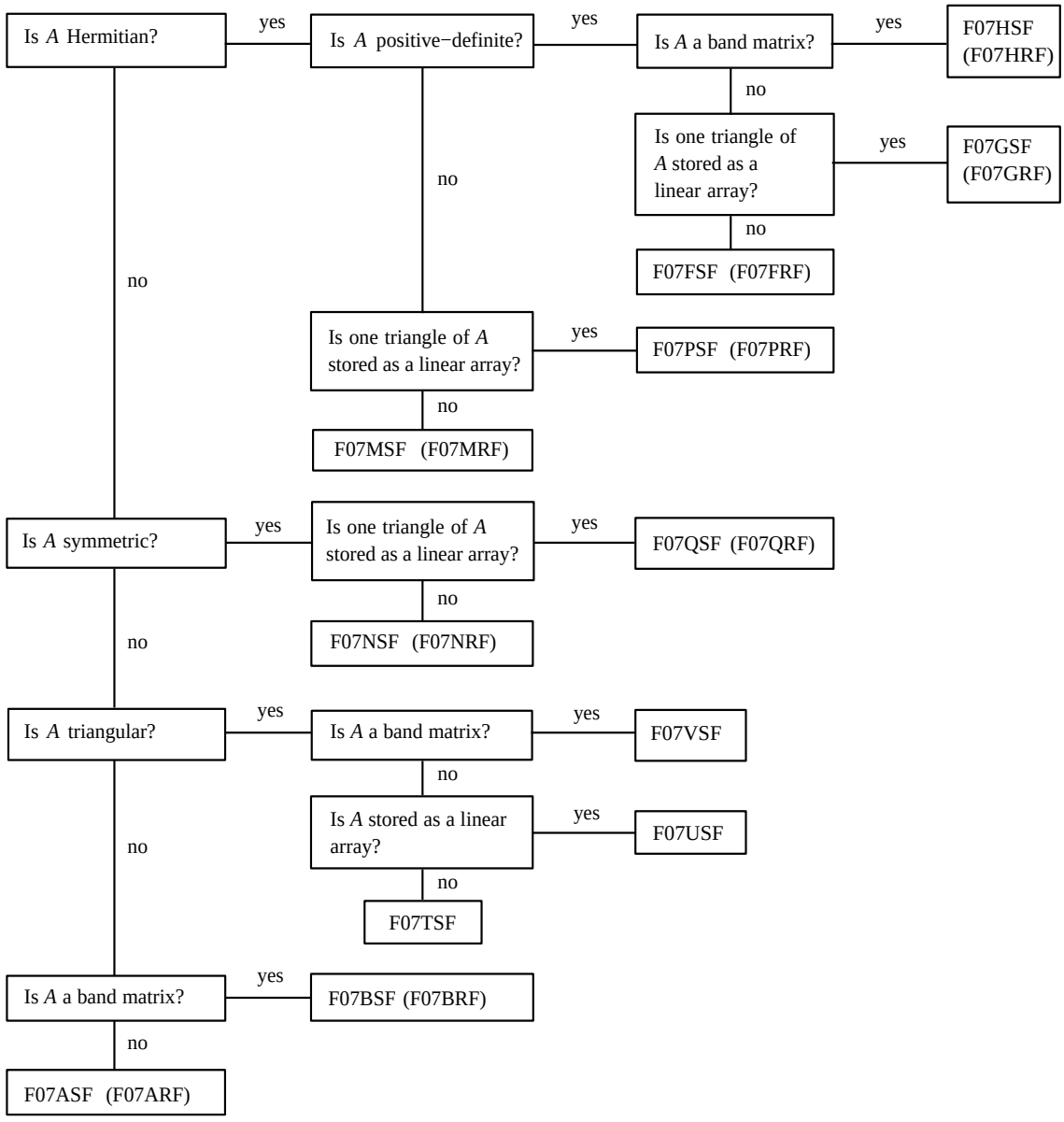
Tree 1: Black Box routines for unique solution of $Ax = b$



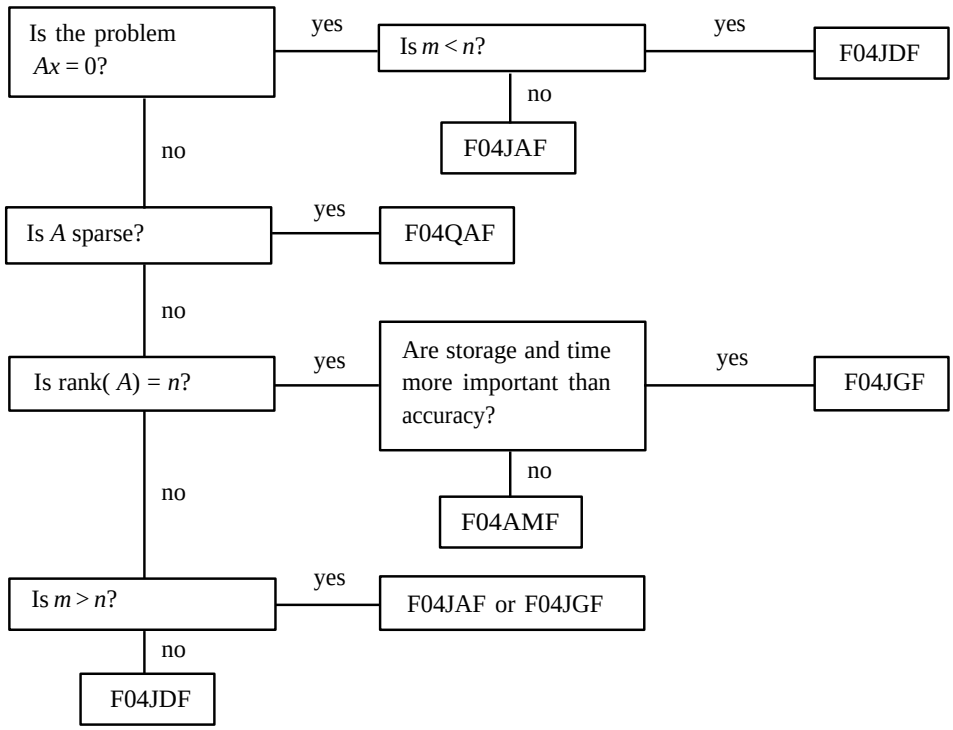
Tree 2: General Purpose routines for unique solution of $Ax = b$
(a) Real matrix



(b) Complex matrix



Tree 3: Least-squares and homogeneous equations (without constraints)



5 Index

Real Symmetric Positive-definite Toeplitz Matrix, General Right-hand Side, Update Solution,	F04MFF
Real Symmetric Positive-definite Toeplitz Matrix, Yule–Walker Equations, Update Solution,	F04MEF
Real Tridiagonal Matrix,	F04LEF

(iii) **Least-squares and Homogeneous Equations**

Complex general Gauss–Markov linear model problem	F04KLF
Complex problem with linear equality constraints	F04KMF
Real m by n Matrix, $m \geq n$, Minimal Solution,	F04JAF
Real m by n Matrix, $m \geq n$, Rank $= n$ or Minimal Solution,	F04JGF
Real m by n Matrix, $m \leq n$, Minimal Solution,	F04JDF
Real m by n Matrix, Rank $= n$, Iterative Refinement,	F04AMF
Real general Gauss–Markov linear model problem	F04JLF
Real problem with linear equality constraints	F04JMF
Real Sparse Matrix,	F04QAF

(iv) **Service Routines**

Complex Matrix, Norm and Condition Number Estimation	F04ZCF
Real Matrix, Norm and Condition Number Estimation	F04YCF

6 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have either been withdrawn or superseded. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

F04ALF	F04ANF	F04AQF	F04AWF	F04AYF	F04AZF
F04LDF	F04MAF	F04MBF	F04NAF		

7 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

[2] Lawson C L and Hanson R J (1974) *Solving Least-squares Problems* Prentice–Hall

[3] Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag