NAG Fortran Library Chapter Introduction

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 large sparse 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 $\text{rank}(A) = n$. 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 $\text{rank}(A) = n$, 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.

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

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

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

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

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

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

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 $Ax \simeq b$, $m > n$, 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, $\operatorname{rank}(A) < n$, 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:

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

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 $\operatorname{rank}(A) < n$, 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

$$1.7, \quad 2.7\times10^{-1}, \quad 2.1\times10^{-2}, \quad 1.0\times10^{-3}, \quad 2.9\times10^{-5}, \quad 4.9\times10^{-7}, \quad 3.5\times10^{-9}.$$

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**:

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

   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**:

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

   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**:

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

   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: please 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 $Ax_i = b_i$, for $i = 1, 2, \ldots, p$, 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 Chapters F01, F03 or F07 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 routine F04QAF which uses an iterative method for sparse systems of equations does not fit easily into this categorisation, but is classified as a general purpose routine 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 using additional precision arithmetic, 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 may not be prohibitive since the same matrix and the same LU decomposition is used in every linear equation solution.

A number of the Black Box routines in this chapter return estimates of the condition number and the forward error, along with the solution of the equations. But for those routines that do no return a condition estimate 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 also be used in conjunction with most of the linear equation solving routines in Chapter F11: 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

The majority of the routines for solving linear least-squares problems are to be found in Chapter F08.

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 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.

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

Problems with linear **equality constraints** can be solved by F08ZBF (DGGGLM) (for real data) or by F08ZPF (ZGGGLM) (for complex data), provided that the problems are of full rank. Problems with linear **inequality constraints** can be solved by E04NCF/E04NCA in Chapter E04.

General Gauss–Markov linear model problems, as formulated in Section 2.5, can be solved by F08ZAF (DGGLSE) (for real data) or by F08ZNF (ZGGLSE) (for complex data).

3.4 Sparse Matrix Routines

Routines specifically for sparse matrices are appropriate 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.

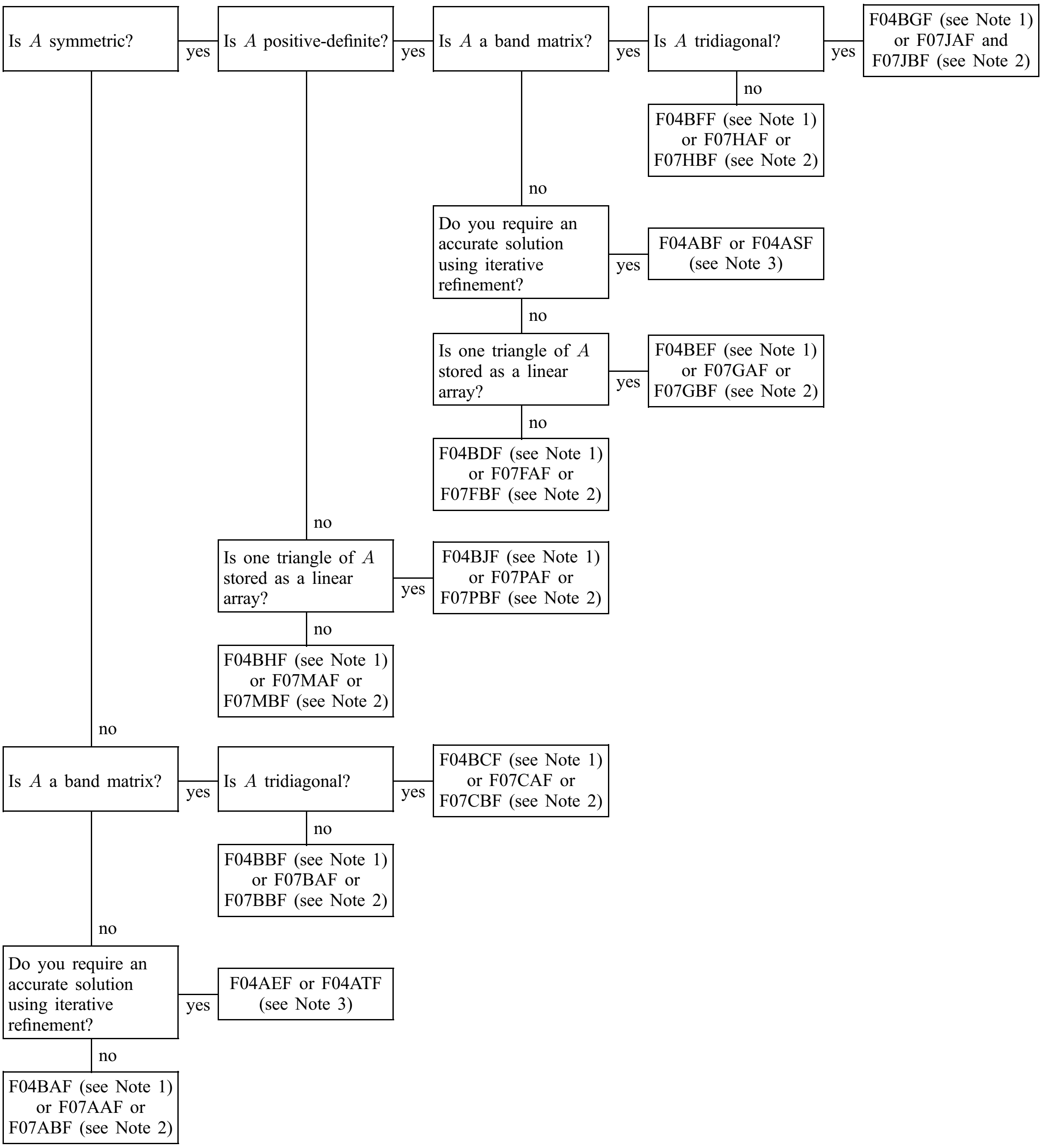
Chapter F11 contains routines for both the direct and iterative solution of real sparse linear systems. There are two routines in Chapter F04 for solving sparse linear equations (F04AXF and F04QAF). F04AXF utilizes a factorization of the matrix A obtained from F01BRF or F01BSF, while F04QAF uses an iterative technique and requires a user-supplied function to compute matrix-vector products Ac and A^Tc for any given vector c.

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).

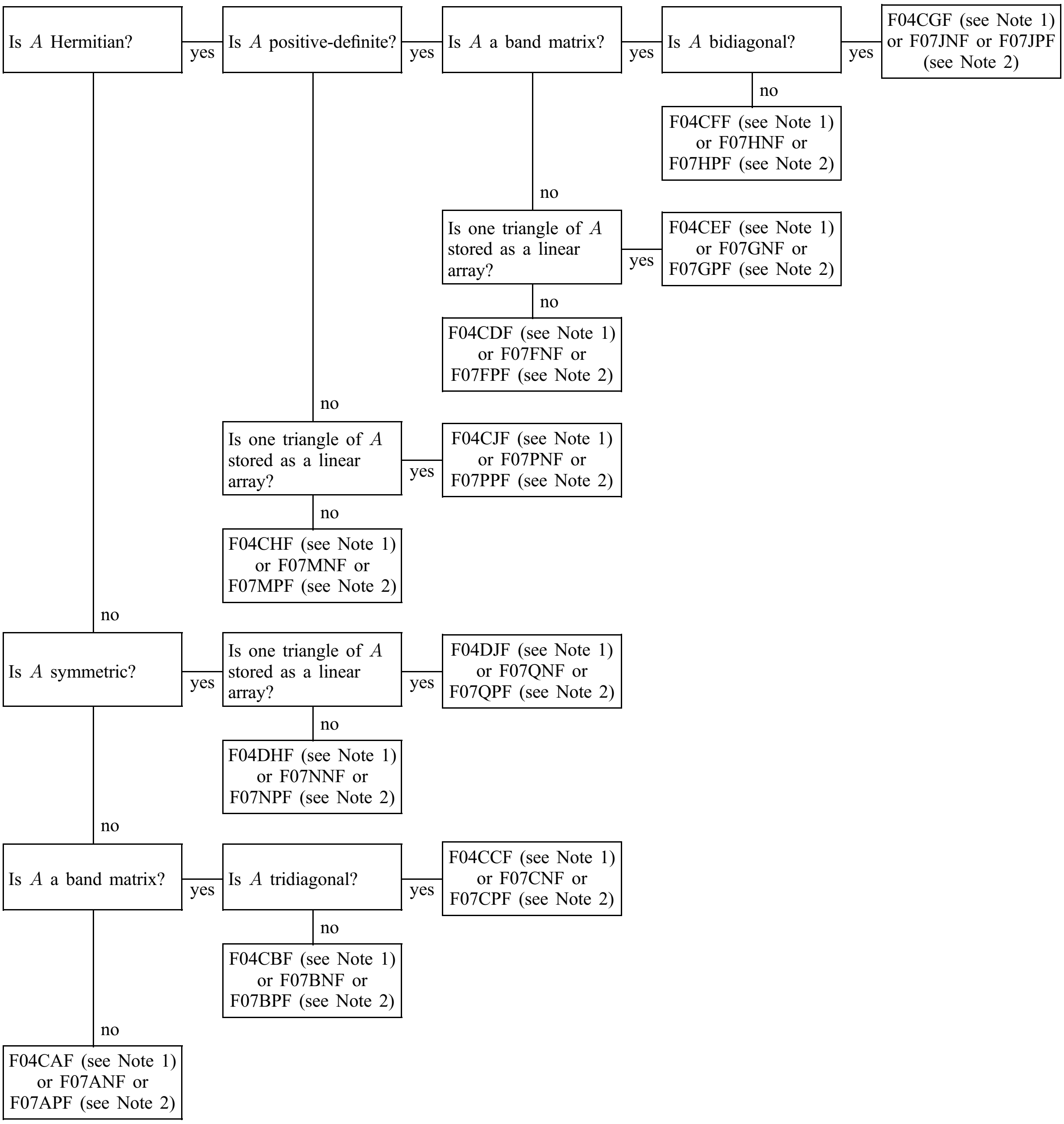
4 Decision Trees

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.

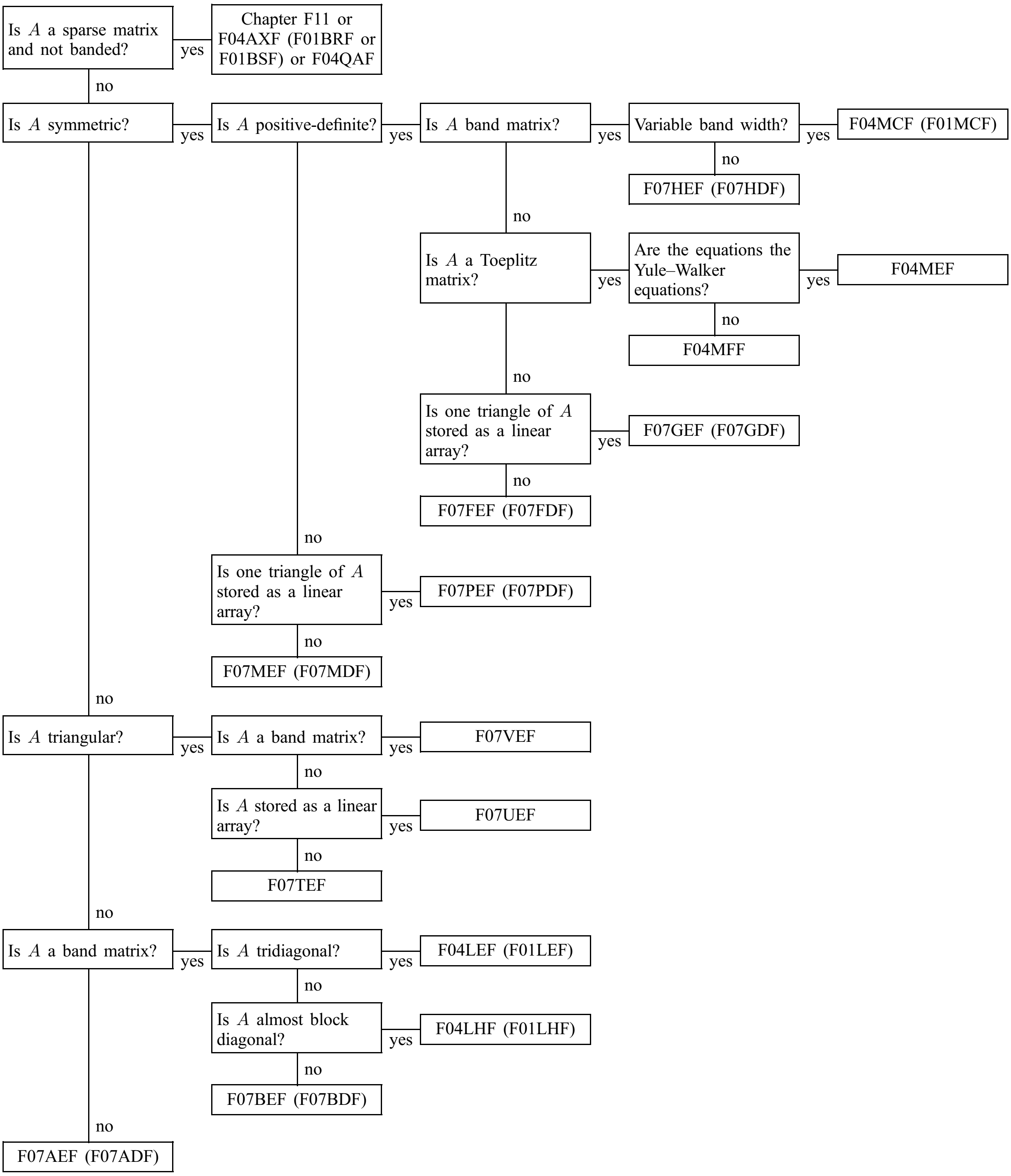
Tree 1: Black Box routines for unique solution of $Ax = b$ (Real matrix)


Is A symmetric?
yes
Is A positive-definite?
yes
Is A a band matrix?
yes
Is A tridiagonal?
yes
F04BGF (see Note 1) or F07JAF and F07JBF (see Note 2)
no
F04BFF (see Note 1) or F07HAF or F07HBF (see Note 2)
no
Do you require an accurate solution using iterative refinement?
yes
F04ABF or F04ASF (see Note 3)
no
Is one triangle of A stored as a linear array?
yes
F04BEF (see Note 1) or F07GAF or F07GBF (see Note 2)
no
F04BDF (see Note 1) or F07FAF or F07FBF (see Note 2)
no
Is one triangle of A stored as a linear array?
yes
F04BJF (see Note 1) or F07PAF or F07PBF (see Note 2)
no
F04BHF (see Note 1) or F07MAF or F07MBF (see Note 2)
no
Is A a band matrix?
yes
Is A tridiagonal?
yes
F04BCF (see Note 1) or F07CAF or F07CBF (see Note 2)
no
F04BBF (see Note 1) or F07BAF or F07BBF (see Note 2)
no
Do you require an accurate solution using iterative refinement?
yes
F04AEF or F04ATF (see Note 3)
no
F04BAF (see Note 1) or F07AAF or F07ABF (see Note 2)

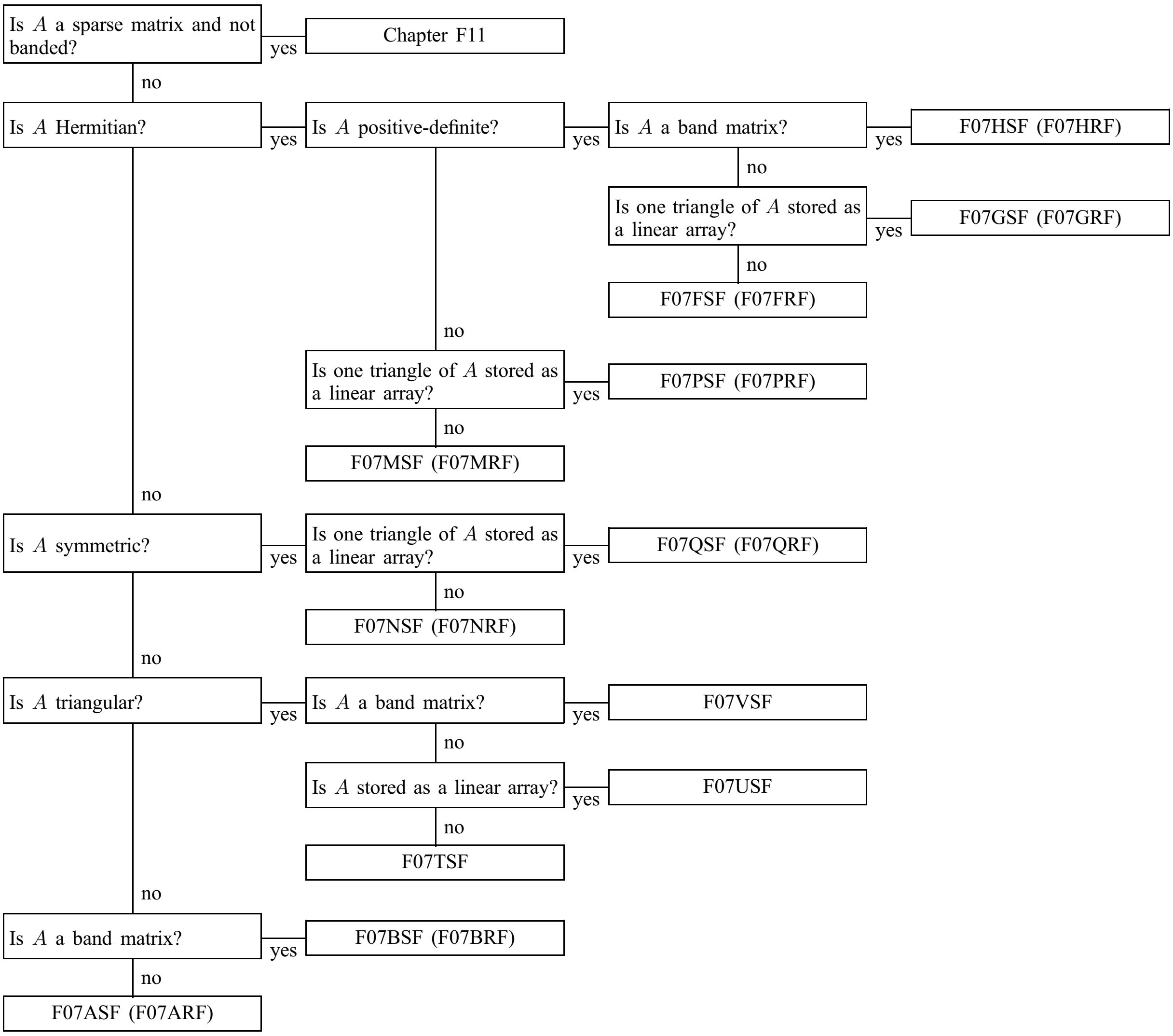
Tree 2: Black Box routines for unique solution of $Ax = b$ (Complex matrix)


Is A Hermitian?
yes
Is A positive-definite?
yes
Is A a band matrix?
yes
Is A bidiagonal?
yes
F04CGF (see Note 1) or F07JNF or F07JPF (see Note 2)
no
F04CFF (see Note 1) or F07HNF or F07HPF (see Note 2)
no
Is one triangle of A stored as a linear array?
yes
F04CEF (see Note 1) or F07GNF or F07GPF (see Note 2)
no
F04CDF (see Note 1) or F07FNF or F07FPF (see Note 2)
no
Is one triangle of A stored as a linear array?
yes
F04CJF (see Note 1) or F07PNF or F07PPF (see Note 2)
no
F04CHF (see Note 1) or F07MNF or F07MPF (see Note 2)
no
Is A symmetric?
yes
Is one triangle of A stored as a linear array?
yes
F04DJF (see Note 1) or F07QNF or F07QPF (see Note 2)
no
F04DHF (see Note 1) or F07NNF or F07NPF (see Note 2)
no
Is A a band matrix?
yes
Is A tridiagonal?
yes
F04CCF (see Note 1) or F07CNF or F07CPF (see Note 2)
no
F04CBF (see Note 1) or F07BNF or F07BPF (see Note 2)
no
F04CAF (see Note 1) or F07ANF or F07APF (see Note 2)

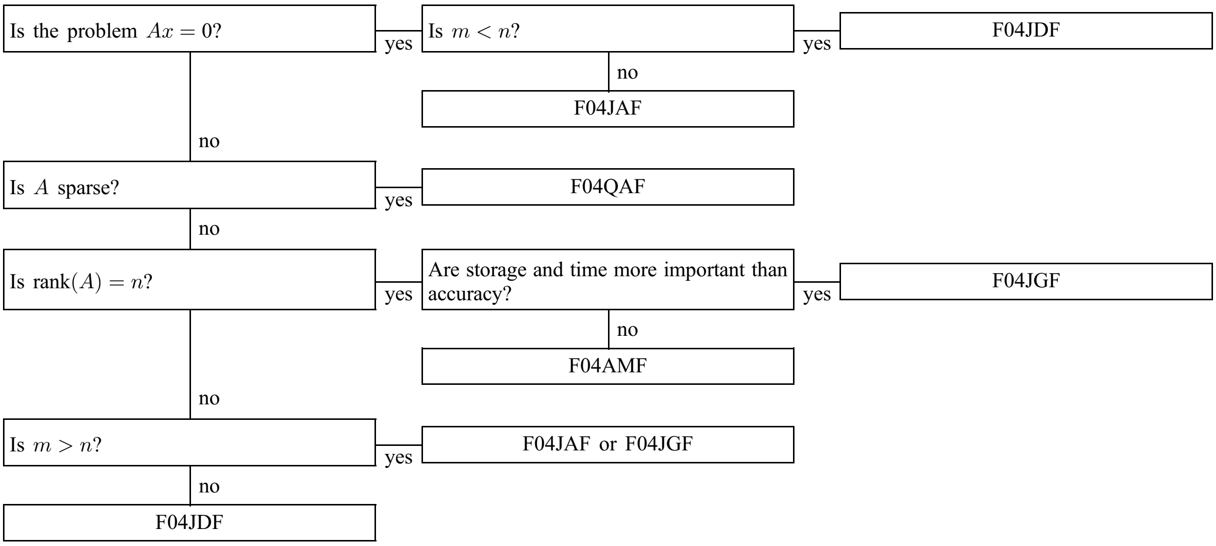
Tree 3: General purpose routines for unique solution of $Ax = b$ (Real matrix)



Tree 4: General purpose routines for unique solution of $Ax = b$ (Complex matrix)



Tree 5: Least-squares and homogeneous equations (without constraints)



Note 1: also returns an estimate of the condition number and the forward error.

Note 2: also returns an estimate of the condition number, the forward error and the backward error. Requires additional workspace.

Note 3: for a single right-hand side only.

5 Index

Black Box routines, $Ax = b$
- Complex general band matrix F04CBF
- Complex general matrix F04CAF
- Complex general tridiagonal matrix F04CCF
- Complex Hermitian matrix
  - Packed matrix format F04CJF
  - Standard matrix format F04CHF
- Complex Hermitian positive-definite band matrix F04CFF
- Complex Hermitian positive-definite matrix
  - Packed matrix format F04CEF
  - Standard matrix format F04CDF
- Complex Hermitian positive-definite tridiagonal matrix F04CGF
- Complex symmetric matrix
  - Packed matrix format F04DJF
  - Standard matrix format F04DHF
- Real general band matrix F04BBF
- Real general matrix
  - Multiple right-hand sides
    - Iterative refinement using additional precision F04AEF
    - Standard precision F04BAF
  - Single right-hand side
    - Iterative refinement using additional precision F04ATF
- Real general tridiagonal matrix F04BCF
- Real symmetric matrix
  - Packed matrix format F04BJF
  - Standard matrix format F04BHF
- Real symmetric positive-definite band matrix F04BFF
- Real symmetric positive-definite matrix
  - Multiple right-hand sides
    - Iterative refinement using additional precision F04ABF
    - Standard precision F04BDF

Packed matrix format F04BEF

Single right-hand side

Iterative refinement using additional precision F04ASF

Real symmetric positive-definite Toeplitz matrix

General right-hand side F04FFF

Yule–Walker equations F04FEF

Real symmetric positive-definite tridiagonal matrix F04BGF

General Purpose routines, $Ax = b$,

real almost block-diagonal matrix F04LHF

real band symmetric positive-definite matrix, variable bandwidth F04MCF

real matrix F04AJF

real matrix, Iterative refinement F04AHF

real sparse matrix,

direct method F04AXF

iterative method F04QAF

real symmetric positive-definite matrix F04AGF

real symmetric positive-definite matrix, Iterative refinement F04AFF

real symmetric positive-definite Toeplitz matrix,

general right-hand side, Update solution F04MFF

Yule–Walker equations, Update solution F04MEF

real tridiagonal matrix F04LEF

Least-squares and Homogeneous Equations,

real m by n matrix,

$m \geq n$, Rank $= n$ or minimal solution F04JGF

rank $= n$, Iterative refinement F04AMF

real sparse matrix F04QAF

Service Routines,

complex matrix,

norm and condition number estimation F04ZCF

real matrix,

covariance matrix for linear least-squares problems F04YAF

norm and condition number estimation F04YCF

6 Routines Withdrawn or Scheduled for Withdrawal

Withdrawn Routine	Mark of Withdrawal	Replacement Routine(s)
F04AAF	23	F07AAF (DGESV)
F04ACF	23	F07HAF (DPBSV)
F04ADF	23	F07ANF (ZGESV)
F04AKF	17	F07ASF (ZGETRS)
F04ALF	17	F07HEF (DPBTRS)
F04ANF	18	F08AGF (DORMQR), F06PJF (DTRSV) and F06EFF (DCOPY)
F04APF	8	F04AXF
F04AQF	16	F07GEF (DPPTRS) and F07PEF (DSPTRS)
F04ARF	23	F07AAF (DGESV)
F04AUF	9	F04JGF
F04AVF	9	F07BEF (DGBTRS)
F04AWF	17	F07FSF (ZPOTRS)
F04AYF	18	F07AEF (DGETRS)
F04AZF	17	F07FEF (DPOTRS)
F04EAF	23	F07CAF (DGTSV)
F04FAF	23	F07JAF (DPTSV)
F04JAF	23	F08KAF (DGELSS)
F04JDF	23	F08KAF (DGELSS)
F04JLF	23	F08ZBF (DGGGLM)
F04JMF	23	F08ZAF (DGGLSE)
F04KLF	23	F08ZPF (ZGGGLM)

F04KMF	23	F08ZNF (ZGGLSE)
F04LDF	18	F07BEF (DGBTRS)
F04MAF	19	F11JCF
F04MBF	19	F11GDF, F11GEF, F11GFF, F11JCF or F11JEF
F04NAF	17	F07BSF (ZGBTRS) and F06SKF (ZTBSV)

7 References

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

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

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