NAG 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 or banded. It may also be rectangular, in which case a least squares solution is obtained.

Much of the functionality of this chapter has been superseded by functions from Chapters f07 and f08 (LAPACK routines) as those chapters have grown and have included driver and expert driver functions.

For a general introduction to sparse systems of equations, see the f11 Chapter Introduction, which provides functions for large sparse systems.

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, for $i = 1, 2, \ldots, p$, 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 $Ax_i = b_i$, for $i = 1, 2, \ldots, p$. Some functions 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.

2.1 Unique Solution of $Ax = b$

Most functions 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

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

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

If A is real symmetric and positive definite, $U = L^{\mathrm{T}}$, while if A is complex Hermitian and positive definite, $U = L^{\mathrm{H}}$; 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.

It must be emphasized 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 less than full precision.

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

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

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.

Another 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 $Ax \simeq b$, $m > n$, $\text{rank}(A) = n$

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

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

The solution is obtained in two steps.

(a) 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 nonsingular 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.

(b) 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 $\hat{x}$.

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

$$A = QDP^{\mathrm{T}}$$

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

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

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 emphasized 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 functions 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.

2.7 Estimating the 1-norm of a Matrix

The 1-norm of a matrix A is defined to be:

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

Typically it is useful to calculate the condition number of a matrix with respect to the solution of linear equations, or inversion. The higher the condition number the less accuracy might be expected from a numerical computation. A condition number for the solution of linear equations is $\|A\|.\left\|A^{-1}\right\|$. Since this might be a relatively expensive computation it often suffices to estimate the norm of each matrix.

3 Recommendations on Choice and Use of Available Functions

See also Section 3 in the f07 Chapter Introduction for recommendations on the choice of available functions from that chapter.

3.1 Black Box and General Purpose Functions

Most of the functions in this chapter are categorised either as Black Box functions or general purpose functions.

Black Box functions 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 functions to use and are suitable when all the right-hand sides are known in advance and do not occupy too much storage.

General purpose functions, in general, require a previous call to a function in Chapters f01 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 functions simply call a factorization function and then a general purpose function to solve the equations.

3.2 Systems of Linear Equations

Most of the functions in this chapter solve linear equations $Ax = b$ when A is n by n and a unique solution is expected (see Section 2.1). 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). For some of the combinations see Chapter f07. nag_real_cholesky_skyline_solve (f04mcc) (which needs to be preceded by a call to nag_real_cholesky_skyline (f01mcc)) can be used for the solution of variable band-width (skyline) positive definite systems.

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

A number of the Black Box functions in this chapter return estimates of the condition number and the forward error, along with the solution of the equations. But for those functions that do not return a condition estimate two functions are provided – nag_linsys_real_gen_norm_rcomm (f04ydc) for real matrices, nag_linsys_complex_gen_norm_rcomm (f04zdc) for complex matrices – which can return a cheap but reliable estimate of $\left\|A^{-1}\right\|$, and hence an estimate of the **condition number** $\kappa(A)$ (see Section 2.1). These functions can also be used in conjunction with most of the linear equation solving functions in Chapter f11: further advice is given in the function documents.

Other functions 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 functions for solving linear least squares problems are to be found in Chapter f08.

Functions for solving linear least squares problems using the QR factorization or the SVD can be found in Chapters f01, f02 and f08. When $m \ge n$ and a unique solution is expected, the QR factorization can

be used, otherwise the QR factorization with pivoting, or the SVD should be used. For $m \gg n$, the SVD is not significantly more expensive than the QR factorization. See Chapter f08 for further discussion.

Problems with linear **equality constraints** can be solved by nag_dgglse (f08zac) (for real data) or by nag_zgglse (f08znc) (for complex data), provided that the problems are of full rank. Problems with linear **inequality constraints** can be solved by nag_opt_lin_lsq (e04ncc) in Chapter e04.

General Gauss–Markov linear model problems, as formulated in Section 2.5, can be solved by nag_dggglm (f08zbc) (for real data) or by nag_zggglm (f08zpc) (for complex data).

3.4 Sparse Matrix Functions

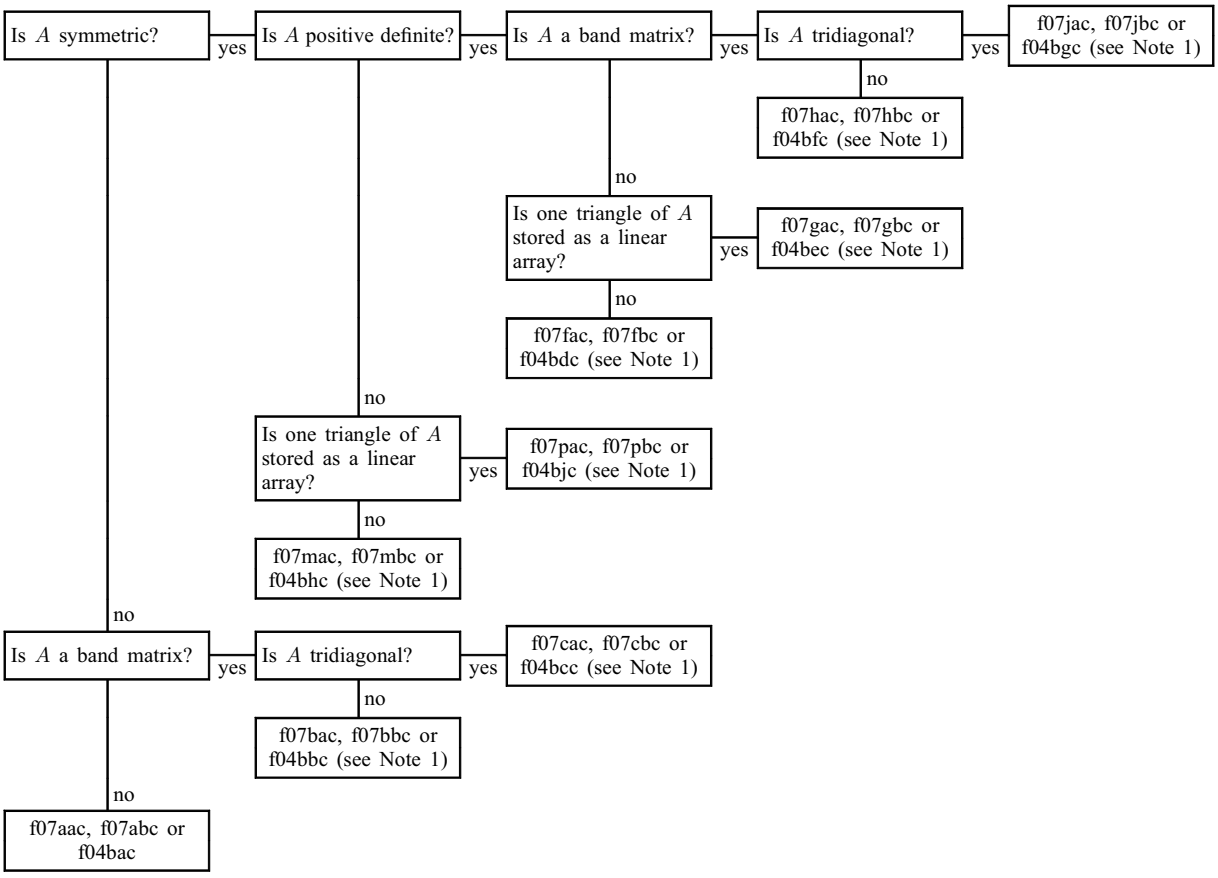
Functions specifically for sparse matrices are appropriate only when the number of nonzero elements is very small, less than, say, 10% of the n^2 elements of A, and the matrix does not have a relatively small band width.

Chapter f11 contains functions for both the direct and iterative solution of sparse linear systems.

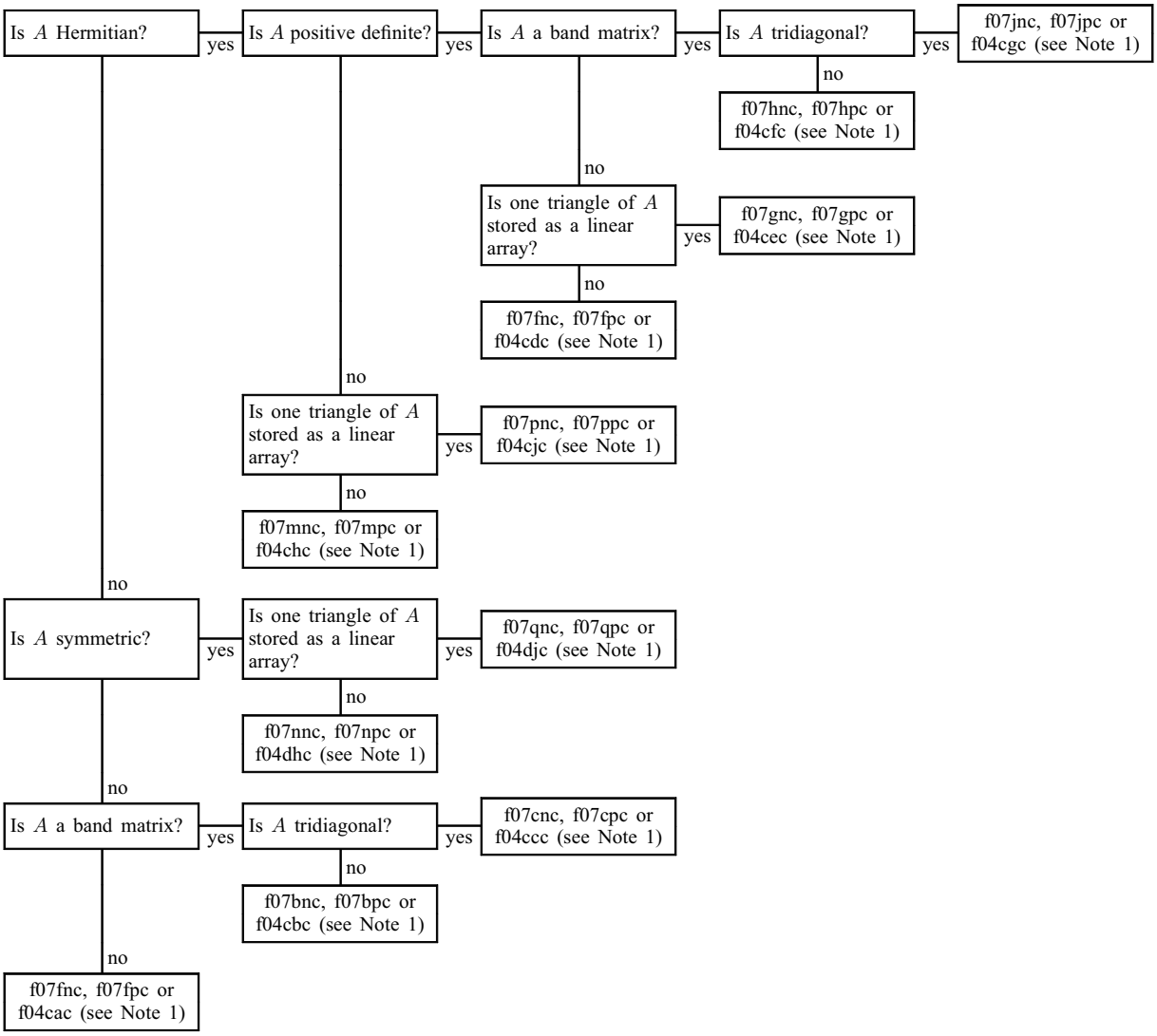
4 Decision Trees

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

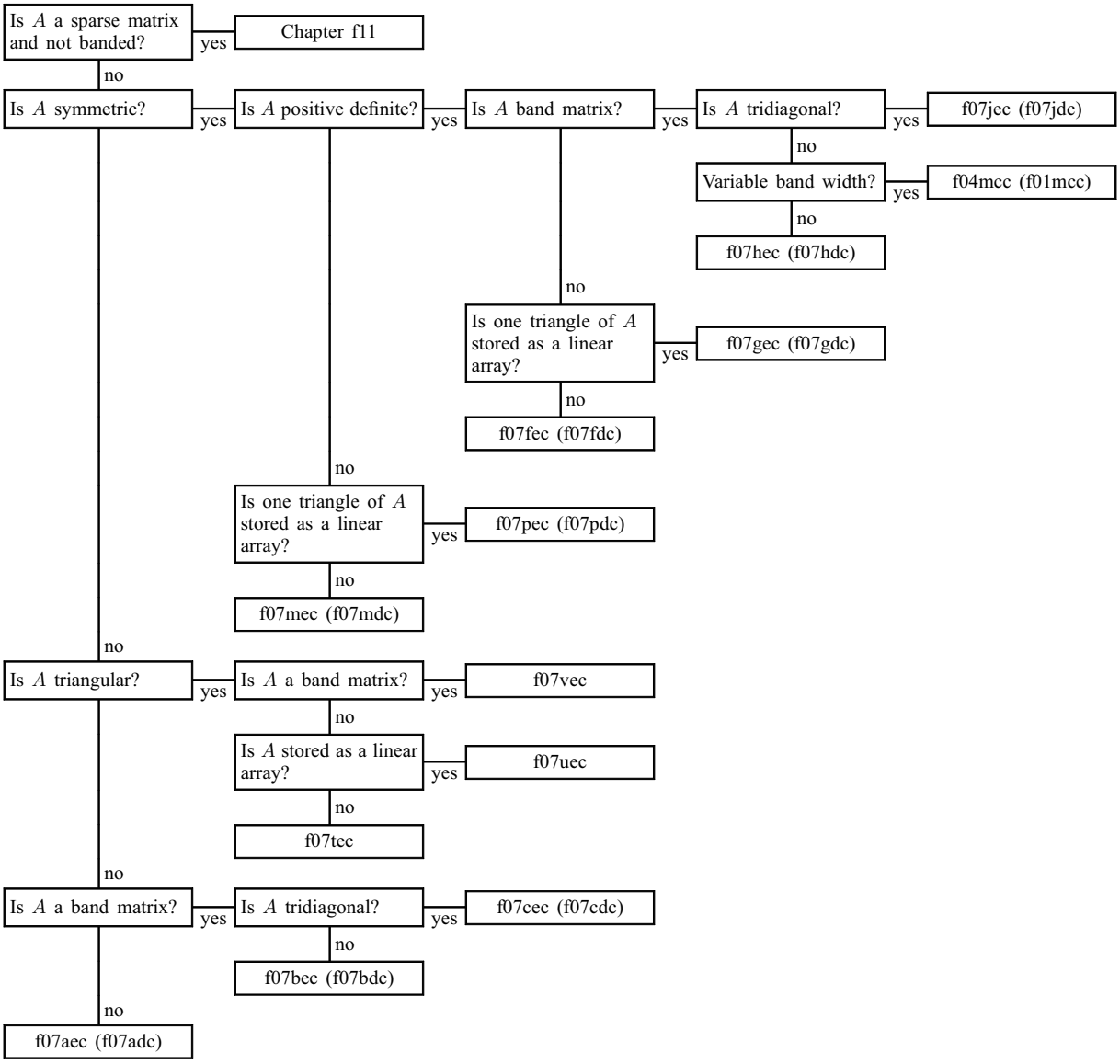
Tree 1: Black Box functions for unique solution of $Ax = b$ (Real matrix)



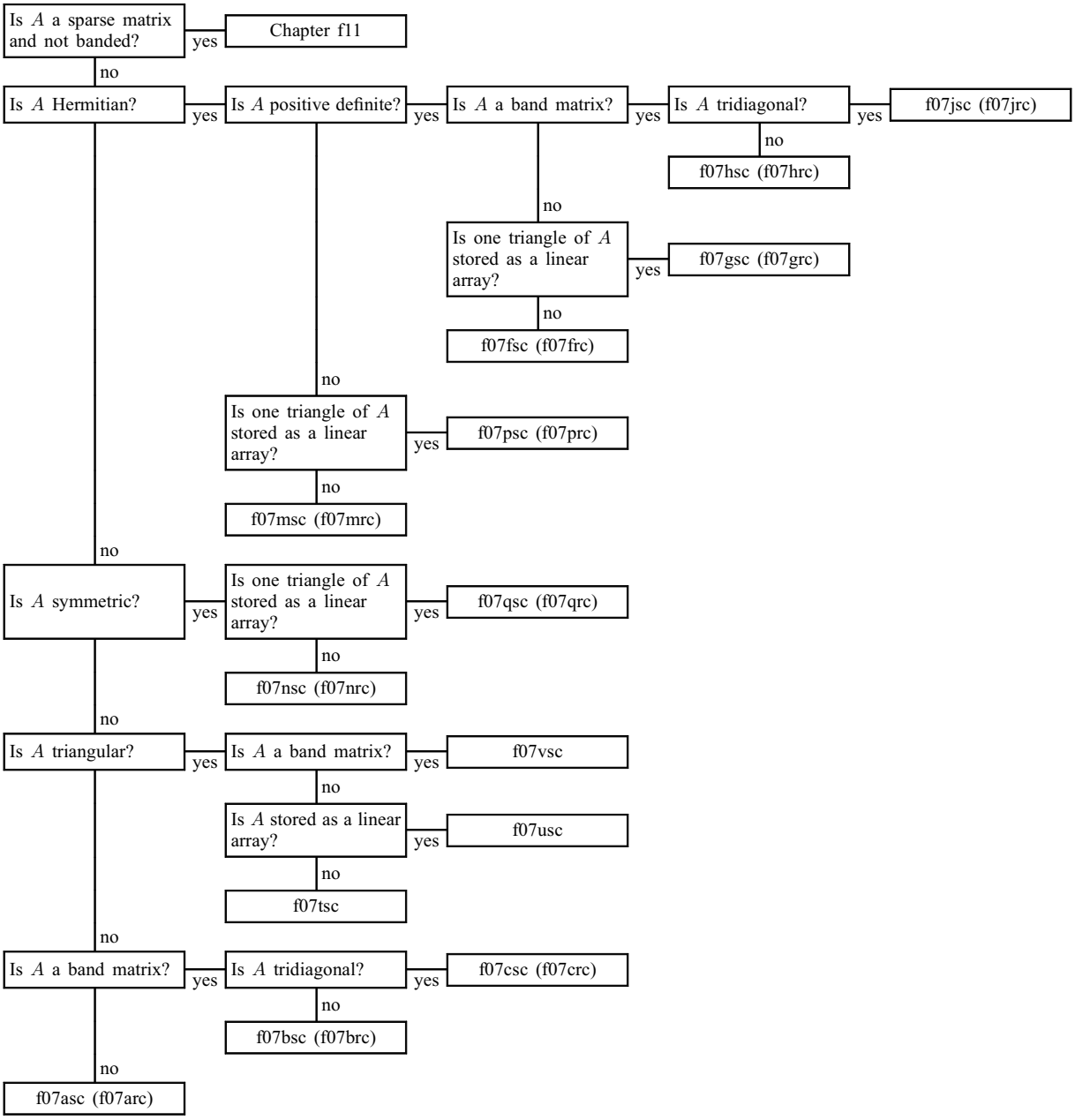
Tree 2: Black Box functions for unique solution of $Ax = b$ (Complex matrix)



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



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



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

5 Functionality Index

Black Box functions, $Ax = b$,

- complex general band matrix ..... nag_complex_band_lin_solve (f04cbc)
- complex general matrix ..... nag_complex_gen_lin_solve (f04cac)
- complex general tridiagonal matrix ..... nag_complex_tridiag_lin_solve (f04ccc)
- complex Hermitian matrix,
  - packed matrix format ..... nag_herm_packed_lin_solve (f04cjc)
  - standard matrix format ..... nag_herm_lin_solve (f04chc)
- complex Hermitian positive definite band matrix ..... nag_herm_posdef_band_lin_solve (f04cfc)
- complex Hermitian positive definite matrix,
  - packed matrix format ..... nag_herm_posdef_packed_lin_solve (f04cec)
  - standard matrix format ..... nag_herm_posdef_lin_solve (f04cdc)

complex Hermitian positive definite tridiagonal matrix
..... nag_herm_posdef_tridiag_lin_solve (f04cgc)
complex symmetric matrix,
packed matrix format ..... nag_complex_sym_packed_lin_solve (f04djc)
standard matrix format ..... nag_complex_sym_lin_solve (f04dhc)
real general band matrix ..... nag_real_band_lin_solve (f04bbc)
real general matrix,
multiple right-hand sides, standard precision ..... nag_real_gen_lin_solve (f04bac)
real general tridiagonal matrix ..... nag_real_tridiag_lin_solve (f04bcc)
real symmetric matrix,
packed matrix format ..... nag_real_sym_packed_lin_solve (f04bjc)
standard matrix format ..... nag_real_sym_lin_solve (f04bhc)
real symmetric positive definite band matrix ..... nag_real_sym_posdef_band_lin_solve (f04bfc)
real symmetric positive definite matrix,
multiple right-hand sides, standard precision ..... nag_real_sym_posdef_lin_solve (f04bdc)
packed matrix format ..... nag_real_sym_posdef_packed_lin_solve (f04bec)
real symmetric positive definite tridiagonal matrix
..... nag_real_sym_posdef_tridiag_lin_solve (f04bgc)

General Purpose functions, $Ax = b$,
real band symmetric positive definite matrix, variable bandwidth
..... nag_real_cholesky_skyline_solve (f04mcc)

Service Functions,
complex rectangular matrix,
norm and condition number estimation ..... nag_linsys_complex_gen_norm_rcomm (f04zdc)
real rectangular matrix,
norm and condition number estimation ..... nag_linsys_real_gen_norm_rcomm (f04ydc)

6 Auxiliary Functions Associated with Library Function Arguments

None.

7 Functions Withdrawn or Scheduled for Withdrawal

The following lists all those functions that have been withdrawn since Mark 23 of the Library or are scheduled for withdrawal at one of the next two marks.

Withdrawn Function	**Mark of Withdrawal**	**Replacement Function(s)**
nag_complex_lin_eqn_mult_rhs (f04adc)	25	nag_complex_gen_lin_solve (f04cac)
nag_real_cholesky_solve_mult_rhs (f04agc)	25	nag_dpotrs (f07fec)
nag_real_lu_solve_mult_rhs (f04ajc)	25	nag_dgetrs (f07aec)
nag_complex_lu_solve_mult_rhs (f04akc)	25	nag_zgetrs (f07asc)
nag_real_lin_eqn (f04arc)	25	nag_real_gen_lin_solve (f04bac)
nag_hermitian_lin_eqn_mult_rhs (f04awc)	25	nag_zpotrs (f07fsc)

8 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