NAG Library Chapter Introduction

f01 – Matrix Operations, Including Inversion

Contents

1 Scope of the Chapter

This chapter provides facilities for three types of problem:

(i) Matrix Inversion

(ii) Matrix Factorizations

(iii) Matrix Functions

These problems are discussed separately in Section 2.1, Section 2.2 and Section 2.3.

2 Background to the Problems

2.1 Matrix Inversion

(i) Nonsingular square matrices of order n.

If A, a square matrix of order n, is nonsingular (has rank n), then its inverse X exists and satisfies the equations $AX = XA = I$ (the identity or unit matrix).

It is worth noting that if $AX - I = R$, so that R is the 'residual' matrix, then a bound on the relative error is given by $\|R\|$, i.e.,

$$\frac{\|X - A^{-1}\|}{\|A^{-1}\|} \leq \|R\|.$$

(ii) General real rectangular matrices.

A real matrix A has no inverse if it is square (n by n) and singular (has rank $< n$), or if it is of shape (m by n) with $m \neq n$, but there is a **Generalized** or **Pseudo-inverse** A^+ which satisfies the equations

$$AA^+A = A, \quad A^+AA^+ = A^+, \quad (AA^+)^{\mathrm{T}} = AA^+, \quad (A^+A)^{\mathrm{T}} = A^+A$$

(which of course are also satisfied by the inverse X of A if A is square and nonsingular).

(a) if $m \geq n$ and $\mathrm{rank}(A) = n$ then A can be factorized using a ***QR*** **factorization**, given by

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where Q is an m by m orthogonal matrix and R is an n by n, nonsingular, upper triangular matrix. The pseudo-inverse of A is then given by

$$A^+ = R^{-1}\tilde{Q}^{\mathrm{T}},$$

where $\tilde{Q}$ consists of the first n columns of Q.

(b) if $m \leq n$ and $\mathrm{rank}(A) = m$ then A can be factorized using an ***RQ*** **factorization**, given by

$$A = (R \quad 0)Q^{\mathrm{T}}$$

where Q is an n by n orthogonal matrix and R is an m by m, nonsingular, upper triangular matrix. The pseudo-inverse of A is then given by

$$A^+ = \tilde{Q}R^{-1},$$

where $\tilde{Q}$ consists of the first m columns of Q.

(c) if $m \geq n$ and $\mathrm{rank}(A) = r \leq n$ then A can be factorized using a QR factorization, with column interchanges, as

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}P^{\mathrm{T}},$$

where Q is an m by m orthogonal matrix, R is an r by n upper trapezoidal matrix and P is an n by n permutation matrix. The pseudo-inverse of A is then given by

$$A^+ = PR^{\mathrm{T}}\left(RR^{\mathrm{T}}\right)^{-1}\tilde{Q}^{\mathrm{T}},$$

where $\tilde{Q}$ consists of the first r columns of Q.

(d) if $\operatorname{rank}(A) = r \leq k = \min(m,n)$, then A can be factorized as the **singular value decomposition**

$$A = U\Sigma V^{\mathrm{T}},$$

where U is an m by m orthogonal matrix, V is an n by n orthogonal matrix and Σ is an m by n diagonal matrix with non-negative diagonal elements σ. The first k columns of U and V are the **left-** and **right-hand singular vectors** of A respectively and the k diagonal elements of Σ are the **singular values** of A. Σ may be chosen so that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$$

and in this case if $\operatorname{rank}(A) = r$ then

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \quad \sigma_{r+1} = \cdots = \sigma_k = 0.$$

If $\tilde{U}$ and $\tilde{V}$ consist of the first r columns of U and V respectively and $\tilde{\Sigma}$ is an r by r diagonal matrix with diagonal elements $\sigma_1, \sigma_2, \ldots, \sigma_r$ then A is given by

$$A = \tilde{U}\tilde{\Sigma}\tilde{V}^{\mathrm{T}}$$

and the pseudo-inverse of A is given by

$$A^+ = \tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^{\mathrm{T}}.$$

Notice that

$$A^{\mathrm{T}}A = V\left(\Sigma^{\mathrm{T}}\Sigma\right)V^{\mathrm{T}}$$

which is the classical eigenvalue (spectral) factorization of $A^{\mathrm{T}}A$.

(e) if A is complex then the above relationships are still true if we use 'unitary' in place of 'orthogonal' and conjugate transpose in place of transpose. For example, the singular value decomposition of A is

$$A = U\Sigma V^{\mathrm{H}},$$

where U and V are unitary, V^{H} the conjugate transpose of V and Σ is as in (d) above.

2.2 Matrix Factorizations

The functions in this section perform matrix factorizations which are required for the solution of systems of linear equations with various special structures. A few functions which perform associated computations are also included.

Other functions for matrix factorizations are to be found in Chapters f07, f08 and f11.

This section also contains a few functions associated with eigenvalue problems (see Chapter f02). (Historical note: this section used to contain many more such functions, but they have now been superseded by functions in Chapter f08.)

2.3 Matrix Functions

Given a square matrix A, the matrix function $f(A)$ is a matrix with the same dimensions as A which provides a generalization of the scalar function f.

If A has a full set of eigenvectors V then A can be factorized as

$$A = VDV^{-1},$$

where D is the diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A. $f(A)$ is given by

$$f(A) = Vf(D)V^{-1},$$

where $f(D)$ is the diagonal matrix whose ith diagonal element is $f(d_i)$.

In general, A may not have a full set of eigenvectors. The matrix function can then be defined via a Cauchy integral. For $A \in \mathbb{C}^{n \times n}$,

$$f(A) = \frac{1}{2\pi i}\int_{\Gamma} f(z)(zI - A)^{-1} dz,$$

where Γ is a closed contour surrounding the eigenvalues of A, and f is analytic within Γ.

Some matrix functions are defined implicitly. A matrix logarithm is a solution X to the equation

$$e^X = A.$$

In general X is not unique, but if A has no eigenvalues on the closed negative real line then a unique *principal logarithm* exists whose eigenvalues have imaginary part between π and $-\pi$. Similarly, a matrix square root is a solution X to the equation

$$X^2 = A.$$

If A has no eigenvalues on the closed negative real line then a unique *principal square root* exists with eigenvalues in the right half-plane. If A has a vanishing eigenvalue then $\log(A)$ cannot be computed. If the vanishing eigenvalue is *defective* (its algebraic multiplicity exceeds its geometric multiplicity, or equivalently it occurs in a Jordan block of size greater than 1) then the square root cannot be computed. If the vanishing eigenvalue is *semisimple* (its algebraic and geometric multiplicities are equal, or equivalently it occurs only in Jordan blocks of size 1) then a square root can be computed.

Algorithms for computing matrix functions are usually tailored to a specific function. Currently Chapter f01 contains routines for calculating the exponential, logarithm, sine, cosine, sinh, cosh, square root and general real power of both real and complex matrices. In addition there are routines to compute a general function of real symmetric and complex Hermitian matrices and a general function of general real and complex matrices.

The Fréchet derivative of a matrix function $f(A)$ in the direction of the matrix E is the linear function mapping E to $L_f(A,E)$ such that

$$f(A+E) - f(A) - L_f(A,E) = O(\|E\|).$$

The Fréchet derivative measures the first-order effect on $f(A)$ of perturbations in A. Chapter f01 contains functions for calculating the Fréchet derivative of the exponential, logarithm and real powers of both real and complex matrices.

The condition number of a matrix function is a measure of its sensitivity to perturbations in the data. The absolute condition number measures these perturbations in an absolute sense, and is defined by

$$\mathrm{cond}_{\mathrm{abs}}(f,A) := \lim_{\epsilon \to 0} \sup_{\{\|E\| \to 0\}} \frac{\|f(A+E) - f(A)\|}{\epsilon}.$$

The relative condition number, which is usually of more interest, measures these perturbations in a relative sense, and is defined by

$$\mathrm{cond}_{\mathrm{rel}}(f,A) = \mathrm{cond}_{\mathrm{abs}}(f,A)\frac{\|A\|}{\|f(A)\|}.$$

The absolute and relative condition numbers can be expressed in terms of the norm of the Fréchet derivative by

$$\mathrm{cond}_{\mathrm{abs}}(f,A) = \max_{E \neq 0} \frac{\|L(A,E)\|}{\|E\|},$$

$$\mathrm{cond}_{\mathrm{rel}}(f,A) = \frac{\|A\|}{\|f(A)\|}\max_{E \neq 0} \frac{\|L(A,E)\|}{\|E\|}.$$

Chapter f01 contains routines for calculating the condition number of the matrix exponential, logarithm, sine, cosine, sinh, cosh, square root and general real power of both real and complex matrices. It also contains routines for estimating the condition number of a general function of a real or complex matrix.

3 Recommendations on Choice and Use of Available Functions

3.1 Matrix Inversion

Note: before using any function for matrix inversion, consider carefully whether it is really needed.

Although the solution of a set of linear equations $Ax = b$ can be written as $x = A^{-1}b$, the solution should **never** be computed by first inverting A and then computing $A^{-1}b$; the functions in Chapters f04 or f07 should **always** be used to solve such sets of equations directly; they are faster in execution, and numerically more stable and accurate. Similar remarks apply to the solution of least squares problems which again should be solved by using the functions in Chapters f04 and f08 rather than by computing a pseudo-inverse.

(a) Nonsingular square matrices of order n

This chapter describes techniques for inverting a general real matrix A and matrices which are positive definite (have all eigenvalues positive) and are either real and symmetric or complex and Hermitian. It is wasteful and uneconomical not to use the appropriate function when a matrix is known to have one of these special forms. A general function must be used when the matrix is not known to be positive definite. In most functions the inverse is computed by solving the linear equations $Ax_i = e_i$, for $i = 1, 2, \ldots, n$, where e_i is the ith column of the identity matrix.

The residual matrix $R = AX - I$, where X is a computed inverse of A, conveys useful information. Firstly $\|R\|$ is a bound on the relative error in X and secondly $\|R\| < \frac{1}{2}$ guarantees the convergence of the iterative process in the 'corrected' inverse functions.

The decision trees for inversion show which functions in Chapter f07 should be used for the inversion of other special types of matrices not treated in the chapter.

(b) General real rectangular matrices

For real matrices nag_dgeqrf (f08aec) returns the QR factorization of the matrix and nag_dgeqp3 (f08bfc) returns the QR factorization with column interchanges. The corresponding complex functions are nag_zgeqrf (f08asc) and nag_zgeqp3 (f08btc) respectively. Functions are also provided to form the orthogonal matrices and transform by the orthogonal matrices following the use of the above functions.

nag_dgesvd (f08kbc) and nag_zgesvd (f08kpc) compute the singular value decomposition as described in Section 2 for real and complex matrices respectively. If A has rank $r \leq k = \min(m, n)$ then the $k - r$ smallest singular values will be negligible and the pseudo-inverse of A can be obtained as $A^{+} = V\Sigma^{-1}U^{\mathrm{T}}$ as described in Section 2. If the rank of A is not known in advance it can be estimated from the singular values (see Section 2.4 in the f04 Chapter Introduction). For large sparse matrices, leading terms in the singular value decomposition can be computed using functions from Chapter f12.

3.2 Matrix Factorizations

Each of these functions serves a special purpose required for the solution of sets of simultaneous linear equations or the eigenvalue problem. For further details you should consult Sections 3 or 4 in the f02 Chapter Introduction or Sections 3 or 4 in the f04 Chapter Introduction.

nag_sparse_nsym_fac (f11dac) is provided for factorizing general real sparse matrices. A more recent algorithm for the same problem is available through nag_superlu_lu_factorize (f11mec). For factorizing real symmetric positive definite sparse matrices, see nag_sparse_sym_chol_fac (f11jac). These functions should be used only when A is **not** banded and when the total number of nonzero elements is less than 10% of the total number of elements. In all other cases either the band functions or the general functions should be used.

3.3 Matrix Functions

nag_real_gen_matrix_exp (f01ecc) and nag_matop_complex_gen_matrix_exp (f01fcc) compute the matrix exponential, e^A, of a real and complex square matrix A respectively. If estimates of the condition number of the matrix exponential are required then nag_matop_real_gen_matrix_cond_exp (f01jgc) and nag_matop_complex_gen_matrix_cond_exp (f01kgc) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_exp (f01jhc) and nag_matop_complex_gen_matrix_frcht_exp (f01khc) should be used.

nag_real_symm_matrix_exp (f01edc) and nag_matop_complex_herm_matrix_exp (f01fdc) compute the matrix exponential, e^A, of a real symmetric and complex Hermitian matrix respectively. If the matrix is real symmetric, or complex Hermitian then it is recommended that nag_real_symm_matrix_exp (f01edc), or nag_matop_complex_herm_matrix_exp (f01fdc) be used as they are more efficient and, in general, more accurate than nag_real_gen_matrix_exp (f01ecc) and nag_matop_complex_gen_matrix_exp (f01fcc).

nag_matop_real_gen_matrix_log (f01ejc) and nag_matop_complex_gen_matrix_log (f01fjc) compute the principal matrix logarithm, $\log(A)$, of a real and complex square matrix A respectively. If estimates of the condition number of the matrix logarithm are required then nag_matop_real_gen_matrix_cond_log (f01jjc) and nag_matop_complex_gen_matrix_cond_log (f01kjc) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_log (f01jkc) and nag_matop_complex_gen_matrix_frcht_log (f01kkc) should be used.

nag_matop_real_gen_matrix_fun_std (f01ekc) and nag_matop_complex_gen_matrix_fun_std (f01fkc) compute the matrix exponential, sine, cosine, sinh or cosh of a real and complex square matrix A respectively. If the matrix exponential is required then it is recommended that nag_real_gen_matrix_exp (f01ecc) or nag_matop_complex_gen_matrix_exp (f01fcc) be used as they are, in general, more accurate than nag_matop_real_gen_matrix_fun_std (f01ekc) and nag_matop_complex_gen_matrix_fun_std (f01fkc). If estimates of the condition number of the matrix function are required then nag_matop_real_gen_matrix_cond_std (f01jac) and nag_matop_complex_gen_matrix_cond_std (f01kac) should be used.

nag_matop_real_gen_matrix_fun_num (f01elc) and nag_matop_real_gen_matrix_fun_usd (f01emc) compute the matrix function, $f(A)$, of a real square matrix. nag_matop_complex_gen_matrix_fun_num (f01flc) and nag_matop_complex_gen_matrix_fun_usd (f01fmc) compute the matrix function of a complex square matrix. The derivatives of f are required for these computations. nag_matop_real_gen_matrix_fun_num (f01elc) and nag_matop_complex_gen_matrix_fun_num (f01flc) use numerical differentiation to obtain the derivatives of f. nag_matop_real_gen_matrix_fun_usd (f01emc) and nag_matop_complex_gen_matrix_fun_usd (f01fmc) use derivatives you have supplied. If estimates of the condition number are required but you are not supplying derivatives then nag_matop_real_gen_matrix_cond_num (f01jbc) and nag_matop_complex_gen_matrix_cond_num (f01kbc) should be used. If estimates of the condition number of the matrix function are required and you are supplying derivatives of f, then nag_matop_real_gen_matrix_cond_usd (f01jcc) and nag_matop_complex_gen_matrix_cond_usd (f01kcc) should be used.

If the matrix A is real symmetric or complex Hermitian then it is recommended that to compute the matrix function, $f(A)$, nag_matop_real_symm_matrix_fun (f01efc) and nag_matop_complex_herm_matrix_fun (f01ffc) are used respectively as they are more efficient and, in general, more accurate than nag_matop_real_gen_matrix_fun_num (f01elc), nag_matop_real_gen_matrix_fun_usd (f01emc), nag_matop_complex_gen_matrix_fun_num (f01flc) and nag_matop_complex_gen_matrix_fun_usd (f01fmc).

nag_matop_real_gen_matrix_actexp (f01gac) and nag_matop_complex_gen_matrix_actexp (f01hac) compute the matrix function $e^{tA}B$ for explicitly stored dense real and complex matrices A and B respectively while nag_matop_real_gen_matrix_actexp_rcomm (f01gbc) and nag_matop_complex_gen_matrix_actexp_rcomm (f01hbc) compute the same using reverse communication. In the latter case, control is returned to you. You should calculate any required matrix-matrix products and then call the function again.

nag_matop_real_gen_matrix_sqrt (f01enc) and nag_matop_complex_gen_matrix_sqrt (f01fnc) compute the principal square root $A^{1/2}$ of a real and complex square matrix A respectively. If A is complex and upper triangular then nag_matop_complex_tri_matrix_sqrt (f01fpc) should be used. If A is real and upper quasi-triangular then nag_matop_real_tri_matrix_sqrt (f01epc) should be used. If estimates of the condition number of the matrix square root are required then nag_matop_real_gen_matrix_cond_sqrt (f01jdc) and nag_matop_complex_gen_matrix_cond_sqrt (f01kdc) should be used.

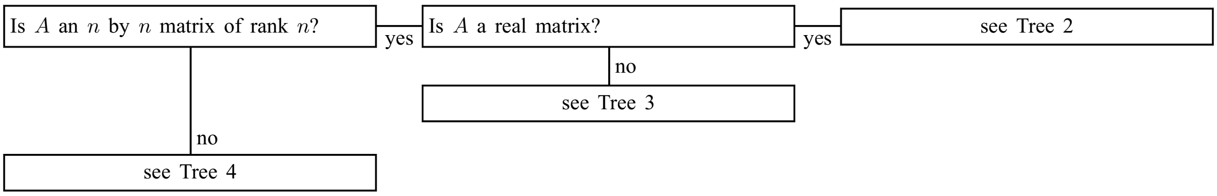
nag_matop_real_gen_matrix_pow (f01eqc) and nag_matop_complex_gen_matrix_pow (f01fqc) compute the matrix power A^p, where $p \in \mathbb{R}$, of real and complex matrices respectively. If estimates of the condition number of the matrix power are required then nag_matop_real_gen_matrix_cond_pow (f01jec) and nag_matop_complex_gen_matrix_cond_pow (f01kec) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_pow (f01jfc) and nag_matop_complex_gen_matrix_frcht_pow (f01kfc) should be used.

4 Decision Trees

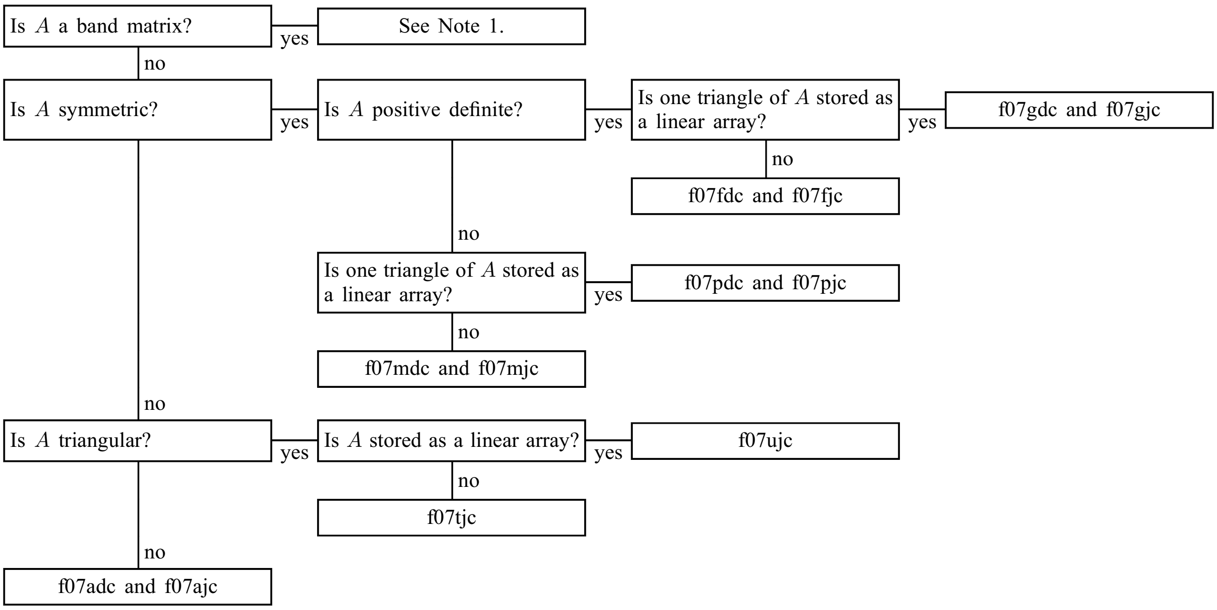
The decision trees show the functions in this chapter and in Chapter f07 and Chapter f08 that should be used for inverting matrices of various types. They also show which function should be used to calculate various matrix functions.

(i) **Matrix Inversion:**

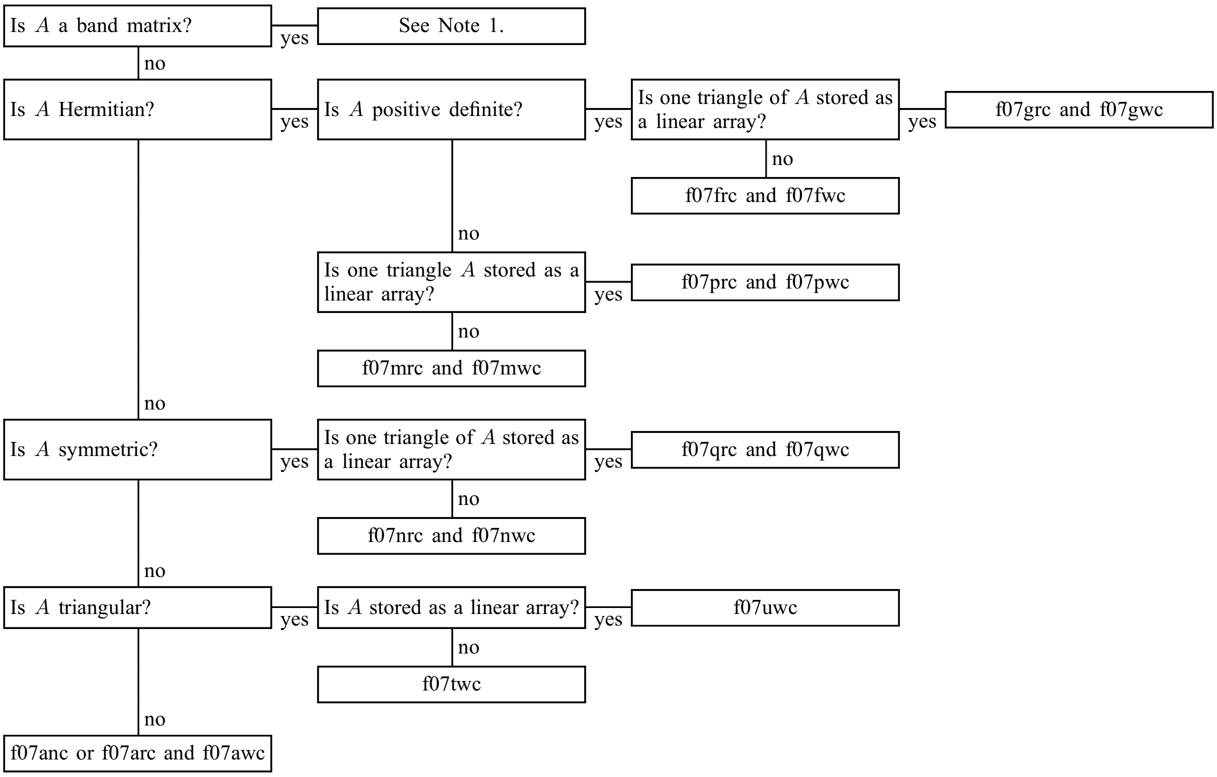
Tree 1



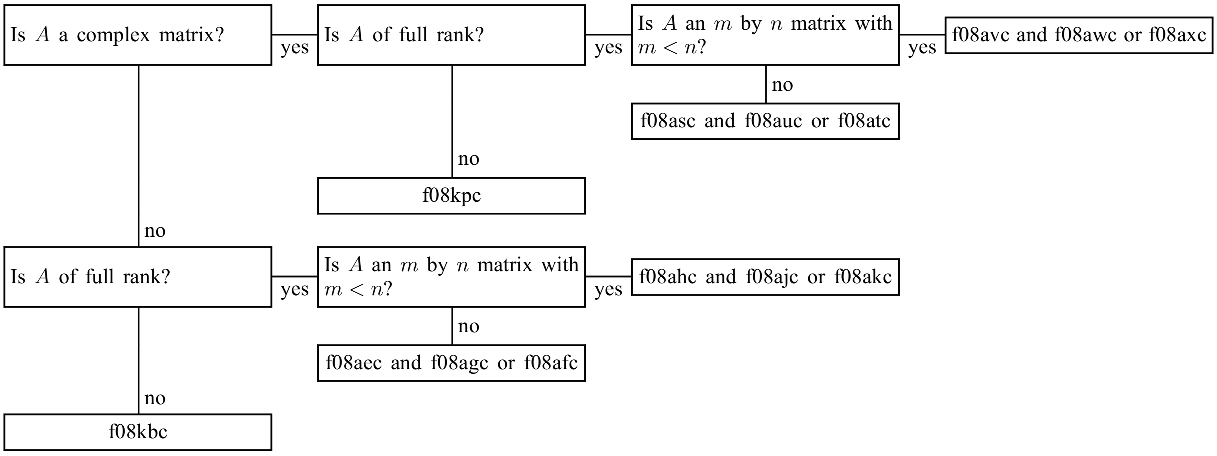
Tree 2: Inverse of a real *n* by *n* matrix of full rank



Tree 3: Inverse of a complex *n* by *n* matrix of full rank



Tree 4: Pseudo-inverses


Is A a complex matrix?
yes
Is A of full rank?
yes
Is A an m by n matrix with $m < n$?
yes
f08avc and f08awc or f08axc
no
f08asc and f08auc or f08atc
no
f08kpc
no
Is A of full rank?
yes
Is A an m by n matrix with $m < n$?
yes
f08ahc and f08ajc or f08akc
no
f08aec and f08agc or f08afc
no
f08kbc

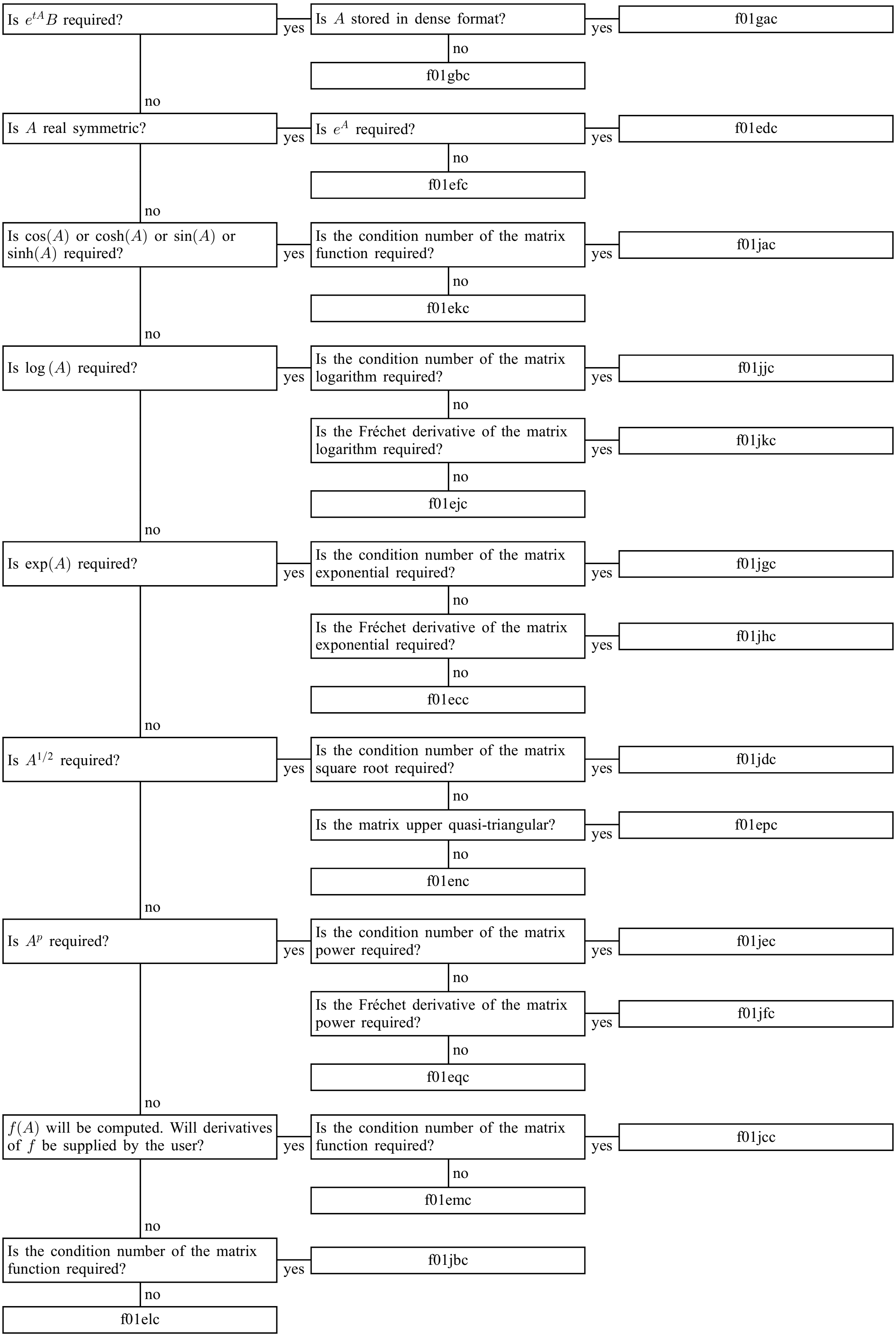
Note 1: the inverse of a band matrix A does not in general have the same shape as A, and no functions are provided specifically for finding such an inverse. The matrix must either be treated as a full matrix, or the equations $AX = B$ must be solved, where B has been initialized to the identity matrix I. In the latter case, see the decision trees in Section 4 in the f04 Chapter Introduction.

Note 2: by ‘guaranteed accuracy’ we mean that the accuracy of the inverse is improved by use of the iterative refinement technique using additional precision.

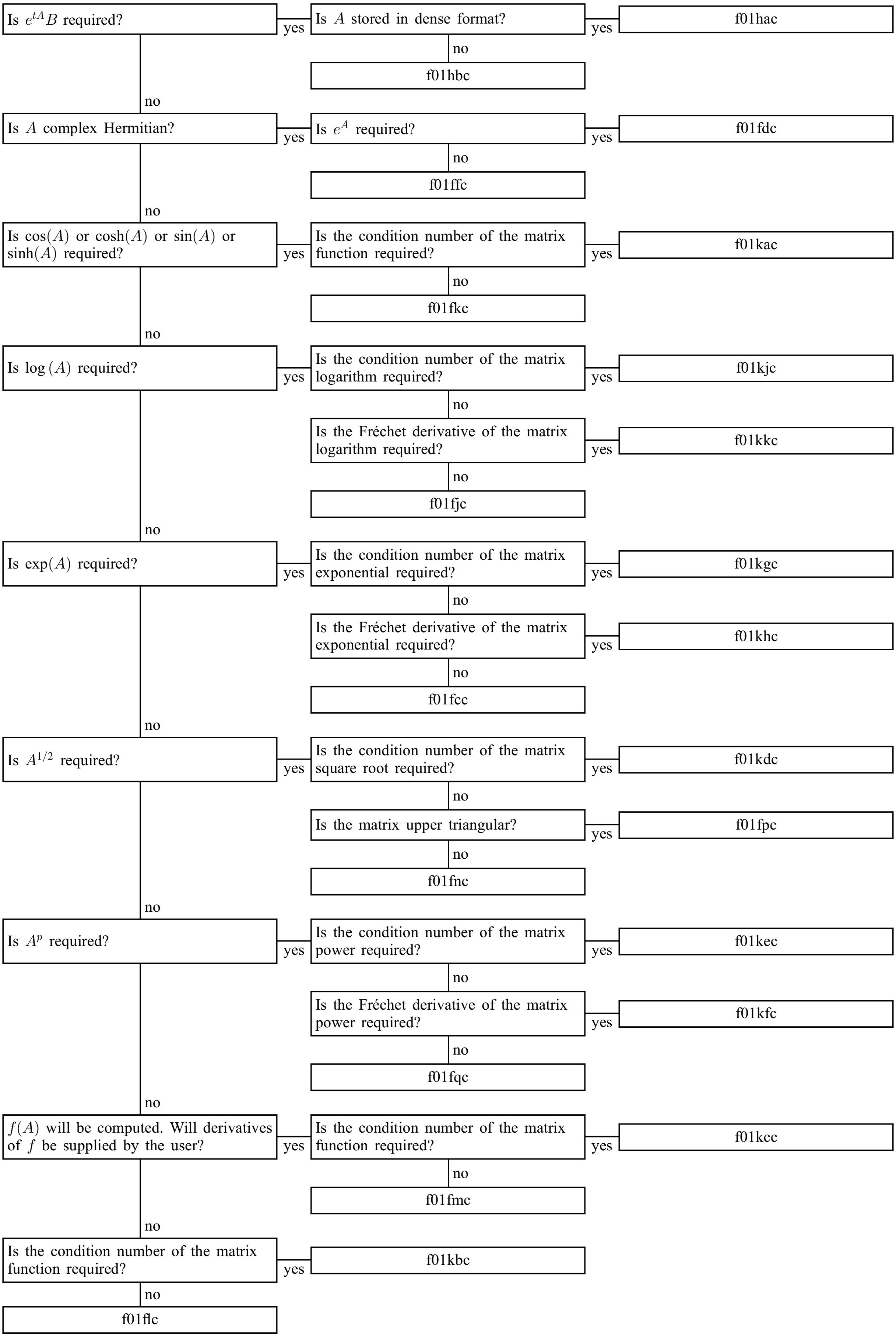
(ii) **Matrix Factorizations:** see the decision trees in Section 4 in the f02 and f04 Chapter Introductions.

(iii) **Matrix Functions:**

Tree 5: Matrix functions $f(A)$ of an *n* by *n* real matrix A


Is $e^{tA}B$ required?
yes
Is A stored in dense format?
yes
f01gac
no
f01gbc
no
Is A real symmetric?
yes
Is e^A required?
yes
f01edc
no
f01efc
no
Is $\cos(A)$ or $\cosh(A)$ or $\sin(A)$ or $\sinh(A)$ required?
yes
Is the condition number of the matrix function required?
yes
f01jac
no
f01ekc
no
Is $\log(A)$ required?
yes
Is the condition number of the matrix logarithm required?
yes
f01jjc
no
Is the Fréchet derivative of the matrix logarithm required?
yes
f01jkc
no
f01ejc
no
Is $\exp(A)$ required?
yes
Is the condition number of the matrix exponential required?
yes
f01jgc
no
Is the Fréchet derivative of the matrix exponential required?
yes
f01jhc
no
f01ecc
no
Is $A^{1/2}$ required?
yes
Is the condition number of the matrix square root required?
yes
f01jdc
no
Is the matrix upper quasi-triangular?
yes
f01epc
no
f01enc
no
Is A^p required?
yes
Is the condition number of the matrix power required?
yes
f01jec
no
Is the Fréchet derivative of the matrix power required?
yes
f01jfc
no
f01eqc
no
$f(A)$ will be computed. Will derivatives of f be supplied by the user?
yes
Is the condition number of the matrix function required?
yes
f01jcc
no
f01emc
no
Is the condition number of the matrix function required?
yes
f01jbc
no
f01elc

Tree 6: Matrix functions $f(A)$ of an *n* by *n* complex matrix A


Is $e^{tA}B$ required?
yes
Is A stored in dense format?
yes
f01hac
no
f01hbc
no
Is A complex Hermitian?
yes
Is e^A required?
yes
f01fdc
no
f01ffc
no
Is $\cos(A)$ or $\cosh(A)$ or $\sin(A)$ or $\sinh(A)$ required?
yes
Is the condition number of the matrix function required?
yes
f01kac
no
f01fkc
no
Is $\log(A)$ required?
yes
Is the condition number of the matrix logarithm required?
yes
f01kjc
no
Is the Fréchet derivative of the matrix logarithm required?
yes
f01kkc
no
f01fjc
no
Is $\exp(A)$ required?
yes
Is the condition number of the matrix exponential required?
yes
f01kgc
no
Is the Fréchet derivative of the matrix exponential required?
yes
f01khc
no
f01fcc
no
Is $A^{1/2}$ required?
yes
Is the condition number of the matrix square root required?
yes
f01kdc
no
Is the matrix upper triangular?
yes
f01fpc
no
f01fnc
no
Is A^p required?
yes
Is the condition number of the matrix power required?
yes
f01kec
no
Is the Fréchet derivative of the matrix power required?
yes
f01kfc
no
f01fqc
no
$f(A)$ will be computed. Will derivatives of f be supplied by the user?
yes
Is the condition number of the matrix function required?
yes
f01kcc
no
f01fmc
no
Is the condition number of the matrix function required?
yes
f01kbc
no
f01flc

5 Functionality Index

Action of the matrix exponential on a complex matrix ..... nag_matop_complex_gen_matrix_actexp (f01hac)

Action of the matrix exponential on a complex matrix (reverse communication) ..... nag_matop_complex_gen_matrix_actexp_rcomm (f01hbc)

Action of the matrix exponential on a real matrix .... nag_matop_real_gen_matrix_actexp (f01gac)

Action of the matrix exponential on a real matrix (reverse communication) ..... nag_matop_real_gen_matrix_actexp_rcomm (f01gbc)

Matrix function,
- complex Hermitian n by n matrix,
  - matrix exponential .... nag_matop_complex_herm_matrix_exp (f01fdc)
  - matrix function .... nag_matop_complex_herm_matrix_fun (f01ffc)
- complex n by n matrix,
  - condition number for a matrix exponential ..... nag_matop_complex_gen_matrix_cond_exp (f01kgc)
  - condition number for a matrix exponential, logarithm, sine, cosine, sinh or cosh ..... nag_matop_complex_gen_matrix_cond_std (f01kac)
  - condition number for a matrix function, using numerical differentiation ..... nag_matop_complex_gen_matrix_cond_num (f01kbc)
  - condition number for a matrix function, using user-supplied derivatives ..... nag_matop_complex_gen_matrix_cond_usd (f01kcc)
  - condition number for a matrix logarithm ..... nag_matop_complex_gen_matrix_cond_log (f01kjc)
  - condition number for a matrix power .... nag_matop_complex_gen_matrix_cond_pow (f01kec)
  - condition number for the matrix square root, logarithm, sine, cosine, sinh or cosh ..... nag_matop_complex_gen_matrix_cond_sqrt (f01kdc)
  - Fréchet derivative
    - matrix exponential .... nag_matop_complex_gen_matrix_frcht_exp (f01khc)
    - matrix logarithm .... nag_matop_complex_gen_matrix_frcht_log (f01kkc)
    - matrix power .... nag_matop_complex_gen_matrix_frcht_pow (f01kfc)
  - general power
    - matrix .... nag_matop_complex_gen_matrix_pow (f01fqc)
  - matrix exponential .... nag_matop_complex_gen_matrix_exp (f01fcc)
  - matrix exponential, sine, cosine, sinh or cosh ..... nag_matop_complex_gen_matrix_fun_std (f01fkc)
  - matrix function, using numerical differentiation ..... nag_matop_complex_gen_matrix_fun_num (f01flc)
  - matrix function, using user-supplied derivatives ..... nag_matop_complex_gen_matrix_fun_usd (f01fmc)
  - matrix logarithm .... nag_matop_complex_gen_matrix_log (f01fjc)
  - matrix square root .... nag_matop_complex_gen_matrix_sqrt (f01fnc)
  - upper triangular
    - matrix square root .... nag_matop_complex_tri_matrix_sqrt (f01fpc)
- real n by n matrix,
  - condition number for a matrix exponential .... nag_matop_real_gen_matrix_cond_exp (f01jgc)
  - condition number for a matrix function, using numerical differentiation ..... nag_matop_real_gen_matrix_cond_num (f01jbc)
  - condition number for a matrix function, using user-supplied derivatives ..... nag_matop_real_gen_matrix_cond_usd (f01jcc)
  - condition number for a matrix logarithm .... nag_matop_real_gen_matrix_cond_log (f01jjc)
  - condition number for a matrix power .... nag_matop_real_gen_matrix_cond_pow (f01jec)
  - condition number for the matrix exponential, logarithm, sine, cosine, sinh or cosh ..... nag_matop_real_gen_matrix_cond_std (f01jac)
  - condition number for the matrix square root, logarithm, sine, cosine, sinh or cosh ..... nag_matop_real_gen_matrix_cond_sqrt (f01jdc)
  - Fréchet derivative
    - matrix exponential .... nag_matop_real_gen_matrix_frcht_exp (f01jhc)

matrix logarithm .... nag_matop_real_gen_matrix_frcht_log (f01jkc)

matrix power .... nag_matop_real_gen_matrix_frcht_pow (f01jfc)

general power

matrix exponential .... nag_matop_real_gen_matrix_pow (f01eqc)

matrix exponential .... nag_real_gen_matrix_exp (f01ecc)

matrix exponential, sine, cosine, sinh or cosh ...... nag_matop_real_gen_matrix_fun_std (f01ekc)

matrix function, using numerical differentiation

..... nag_matop_real_gen_matrix_fun_num (f01elc)

matrix function, using user-supplied derivatives

..... nag_matop_real_gen_matrix_fun_usd (f01emc)

matrix logarithm .... nag_matop_real_gen_matrix_log (f01ejc)

matrix square root .... nag_matop_real_gen_matrix_sqrt (f01enc)

upper quasi-triangular

matrix square root .... nag_matop_real_tri_matrix_sqrt (f01epc)

real symmetric n by n matrix,

matrix exponential .... nag_real_symm_matrix_exp (f01edc)

matrix function .... nag_matop_real_symm_matrix_fun (f01efc)

Matrix Transformations,

real band symmetric positive definite matrix,

variable bandwidth, LDL^{T} factorization .... nag_real_cholesky_skyline (f01mcc)

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_cholesky (f01bnc)	25	nag_zpotrf (f07frc)
nag_real_qr (f01qcc)	25	nag_dgeqrf (f08aec)
nag_real_apply_q (f01qdc)	25	nag_dormqr (f08agc)
nag_real_form_q (f01qec)	25	nag_dorgqr (f08afc)
nag_complex_qr (f01rcc)	25	nag_zgeqrf (f08asc)
nag_complex_apply_q (f01rdc)	25	nag_zunmqr (f08auc)
nag_complex_form_q (f01rec)	25	nag_zungqr (f08atc)

8 References

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

Higham N J (2008) *Functions of Matrices: Theory and Computation* SIAM, Philadelphia, PA, USA

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

Wilkinson J H (1977) Some recent advances in numerical linear algebra *The State of the Art in Numerical Analysis* (ed D A H Jacobs) Academic Press

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