NAG Toolbox Chapter Introduction

F01 – Matrix Operations, Including Inversion

Contents

1 Scope of the Chapter

This chapter provides facilities for four types of problem:

(i) Matrix Inversion

(ii) Matrix Factorizations

(iii) Matrix Arithmetic and Manipulation

(iv) Matrix Functions

See Section 2.1, Section 2.2, Section 2.3 and Section 2.4 where these problems are discussed.

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.,

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

(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 $\mathrm{rank}(A) = r \leq k = \min(m,n)$, 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 $\mathrm{rank}(A) = r$ 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 Arithmetic and Manipulation

The intention of functions in this section (sub-chapters F01C, F01V and F01Z) is to cater for some of the commonly occurring operations in matrix manipulation, i.e., transposing a matrix or adding part of one matrix to another, and for conversion between different storage formats,such as conversion between rectangular band matrix storage and packed band matrix storage. For vector or matrix-vector or matrix-matrix operations refer to Chapter F16.

2.4 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

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

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

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.

Functions are given for calculating the approximate inverse, that is solving the linear equations just once, and also for obtaining the accurate inverse by successive iterative corrections of this first approximation. 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 inverse. In practice the storage requirements for the 'corrected' inverse functions are about double those of the 'approximate' inverse functions, though the extra computer time is not prohibitive since the same matrix and the same LU decomposition is used in every linear equation solution.

Despite the extra work of the 'corrected' inverse functions they are superior to the 'approximate' inverse functions. A correction provides a means of estimating the number of accurate figures in the inverse or the number of 'meaningful' figures relating to the degree of uncertainty in the coefficients of the 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 F04 and 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_lapack_dgeqrf (f08ae) and nag_matop_real_gen_rq (f01qj) return QR and RQ factorizations of A respectively and nag_lapack_dgeqp3 (f08bf) returns the QR factorization with column interchanges. The corresponding complex functions are nag_lapack_zgeqrf (f08as), nag_matop_complex_gen_rq (f01rj) and nag_lapack_zgeqp3 (f08bt) respectively. Functions are also provided to form the orthogonal matrices and transform by the orthogonal matrices following

the use of the above functions. nag_matop_real_trapez_rq (f01qg) and nag_matop_complex_trapez_rq (f01rg) form the RQ factorization of an upper trapezoidal matrix for the real and complex cases respectively.

nag_matop_real_gen_pseudinv (f01bl) uses the QR factorization as described in Section 2.1(ii)(a) and is the only function that explicitly returns a pseudo-inverse. If $m \geq n$, then the function will calculate the pseudo-inverse A^+ of the matrix A. If $m < n$, then the n by m matrix A^{T} should be used. The function will calculate the pseudo-inverse $Z = (A^{\mathrm{T}})^+ = (A^+)^{\mathrm{T}}$ of A^{T} and the required pseudo-inverse will be Z^{T}. The function also attempts to calculate the rank, r, of the matrix given a tolerance to decide when elements can be regarded as zero. However, should this function fail due to an incorrect determination of the rank, the singular value decomposition method (described below) should be used.

nag_lapack_dgesvd (f08kb) and nag_lapack_zgesvd (f08kp) 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). In the real case with $m \geq n$, nag_lapack_dgeqrf (f08ae) followed by nag_eigen_real_triang_svd (f02wu) provide details of the QR factorization or the singular value decomposition depending on whether or not A is of full rank and for some problems provides an attractive alternative to nag_lapack_dgesvd (f08kb). 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 Section 3 or Section 4 in the F02 Chapter Introduction or Section 3 or Section 4 in the F04 Chapter Introduction.

nag_matop_real_gen_sparse_lu (f01br) and nag_matop_real_gen_sparse_lu_reuse (f01bs) are provided for factorizing general real sparse matrices. A more recent algorithm for the same problem is available through nag_sparse_direct_real_gen_lu (f11me). For factorizing real symmetric positive definite sparse matrices, see nag_sparse_real_symm_precon_ichol (f11ja). 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 Arithmetic and Manipulation

The functions in the F01C section are designed for the general handling of m by n matrices. Emphasis has been placed on flexibility in the argument specifications and on avoiding, where possible, the use of internally declared arrays. They are therefore suited for use with large matrices of variable row and column dimensions. Functions are included for the addition and subtraction of sub-matrices of larger matrices, as well as the standard manipulations of full matrices. Those functions involving matrix multiplication may use additional-precision arithmetic for the accumulation of inner products. See also .

The functions in the F01V (LAPACK) and F01Z section are designed to allow conversion between full storage format and one of the packed storage schemes required by some of the functions in Chapters F02, F04, F07 and F08.

3.3.1 NAG Names and LAPACK Names

Functions with NAG name beginning F01V may be called either by their NAG names or by their LAPACK names. When using the NAG Library, the double precision form of the LAPACK name must be used (beginning with D- or Z-).

References to Chapter F01 functions in the manual normally include the LAPACK double precision names, for example, nag_matop_dtrttf (f01ve).

The LAPACK function names follow a simple scheme (which is similar to that used for the BLAS in). Most names have the structure XYYTZZ, where the components have the following meanings:

–the initial letter, X, indicates the data type (real or complex) and precision:

S – real, single precision (in Fortran, 4 byte length REAL)

D – real, double precision (in Fortran, 8 byte length REAL)

C – complex, single precision (in Fortran, 8 byte length COMPLEX)

Z – complex, double precision (in Fortran, 16 byte length COMPLEX)

–the fourth letter, T, indicates that the function is performing a storage scheme transformation (conversion)

–the letters YY indicate the original storage scheme used to store a triangular part of the matrix A, while the letters ZZ indicate the target storage scheme of the conversion (YY cannot equal ZZ since this would do nothing):

TF – Rectangular Full Packed Format (RFP)

TP – Packed Format

TR – Full Format

3.4 Matrix Functions

nag_matop_real_gen_matrix_exp (f01ec) and nag_matop_complex_gen_matrix_exp (f01fc) 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 (f01jg) and nag_matop_complex_gen_matrix_cond_exp (f01kg) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_exp (f01jh) and nag_matop_complex_gen_matrix_frcht_exp (f01kh) should be used.

nag_matop_real_symm_matrix_exp (f01ed) and nag_matop_complex_herm_matrix_exp (f01fd) 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_matop_real_symm_matrix_exp (f01ed), or nag_matop_complex_herm_matrix_exp (f01fd) be used as they are more efficient and, in general, more accurate than nag_matop_real_gen_matrix_exp (f01ec) and nag_matop_complex_gen_matrix_exp (f01fc).

nag_matop_real_gen_matrix_log (f01ej) and nag_matop_complex_gen_matrix_log (f01fj) 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 (f01jj) and nag_matop_complex_gen_matrix_cond_log (f01kj) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_log (f01jk) and nag_matop_complex_gen_matrix_frcht_log (f01kk) should be used.

nag_matop_real_gen_matrix_fun_std (f01ek) and nag_matop_complex_gen_matrix_fun_std (f01fk) 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_matop_real_gen_matrix_exp (f01ec) or nag_matop_complex_gen_matrix_exp (f01fc) be used as they are, in general, more accurate than nag_matop_real_gen_matrix_fun_std (f01ek) and nag_matop_complex_gen_matrix_fun_std (f01fk). If estimates of the condition number of the matrix function are required then nag_matop_real_gen_matrix_cond_std (f01ja) and nag_matop_complex_gen_matrix_cond_std (f01ka) should be used.

nag_matop_real_gen_matrix_fun_num (f01el) and nag_matop_real_gen_matrix_fun_usd (f01em) compute the matrix function, $f(A)$, of a real square matrix. nag_matop_complex_gen_matrix_fun_num (f01fl) and nag_matop_complex_gen_matrix_fun_usd (f01fm) 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 (f01el) and nag_matop_complex_gen_matrix_fun_num (f01fl) use numerical differentiation to obtain the derivatives of f. nag_matop_real_gen_matrix_fun_usd (f01em) and nag_matop_complex_gen_matrix_fun_usd (f01fm) 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 (f01jb) and nag_matop_complex_gen_matrix_cond_num (f01kb) 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 (f01jc) and nag_matop_complex_gen_matrix_cond_usd (f01kc) 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 (f01ef) and nag_matop_complex_herm_matrix_fun (f01ff) are used respectively as they are more efficient and, in general, more accurate than nag_matop_real_gen_matrix_fun_num (f01el), nag_matop_real_gen_matrix_fun_usd (f01em), nag_matop_complex_gen_matrix_fun_num (f01fl) and nag_matop_complex_gen_matrix_fun_usd (f01fm).

nag_matop_real_gen_matrix_actexp (f01ga) and nag_matop_complex_gen_matrix_actexp (f01ha) 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 (f01gb) and nag_matop_complex_gen_matrix_actexp_rcomm (f01hb) 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. See Section [Direct and Reverse Communication functions] in Calling NAG Routines From MATLAB for further information.

nag_matop_real_gen_matrix_sqrt (f01en) and nag_matop_complex_gen_matrix_sqrt (f01fn) 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 (f01fp) should be used. If A is real and upper quasi-triangular then nag_matop_real_tri_matrix_sqrt (f01ep) should be used. If estimates of the condition number of the matrix square root are required then nag_matop_real_gen_matrix_cond_sqrt (f01jd) and nag_matop_complex_gen_matrix_cond_sqrt (f01kd) should be used.

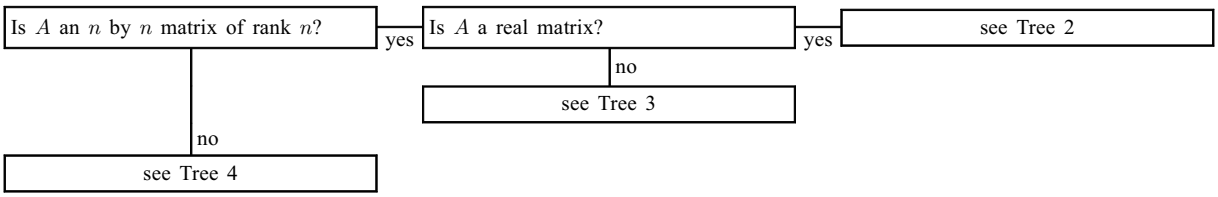
nag_matop_real_gen_matrix_pow (f01eq) and nag_matop_complex_gen_matrix_pow (f01fq) compute the matrix power A^p, where $p \in \mathbb{R}$, of real and complex matrices respectively. If estimates of the condition number of the matrix power are required then nag_matop_real_gen_matrix_cond_pow (f01je) and nag_matop_complex_gen_matrix_cond_pow (f01ke) should be used. If Fréchet derivatives are required then nag_matop_real_gen_matrix_frcht_pow (f01jf) and nag_matop_complex_gen_matrix_frcht_pow (f01kf) should be used.

4 Decision Trees

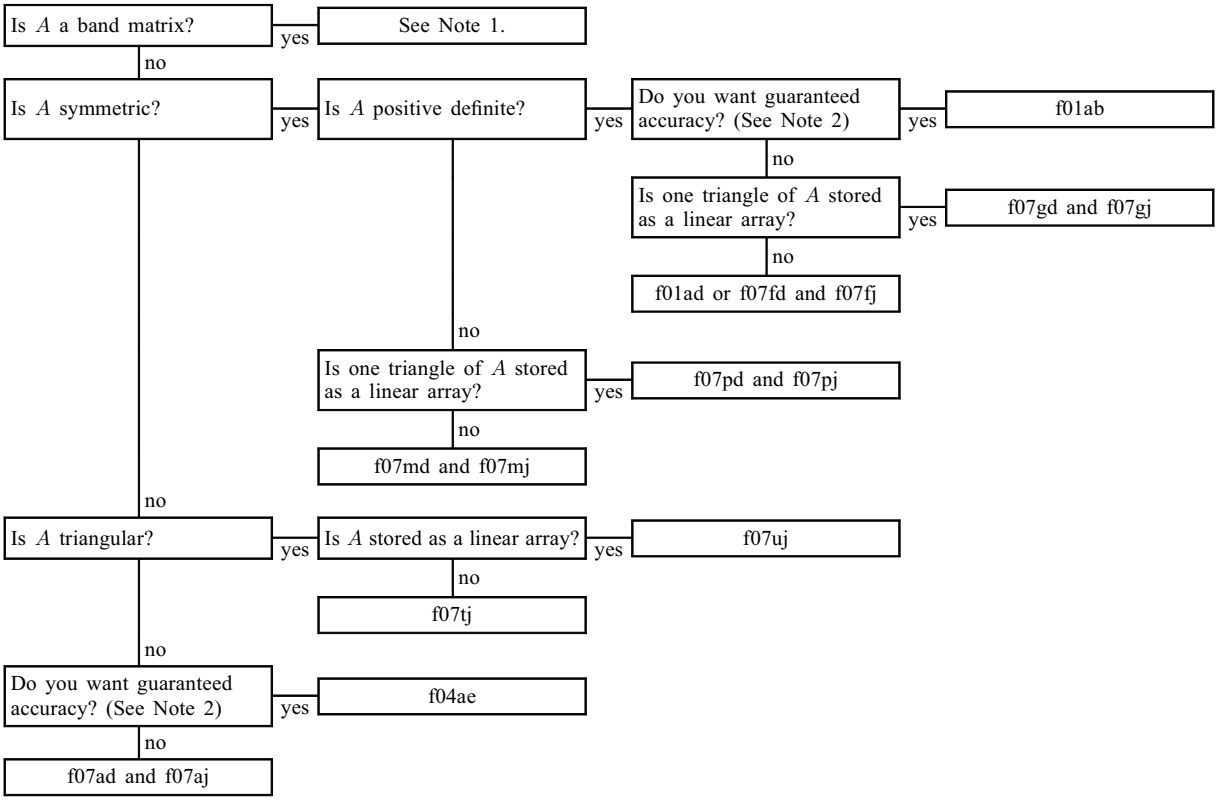
The decision trees show the functions in this chapter and in Chapter F04, Chapter F07 and Chapter F08 that should be used for inverting matrices of various types. They also show which function should be used to calculate various matrix functions.

(i) **Matrix Inversion:**

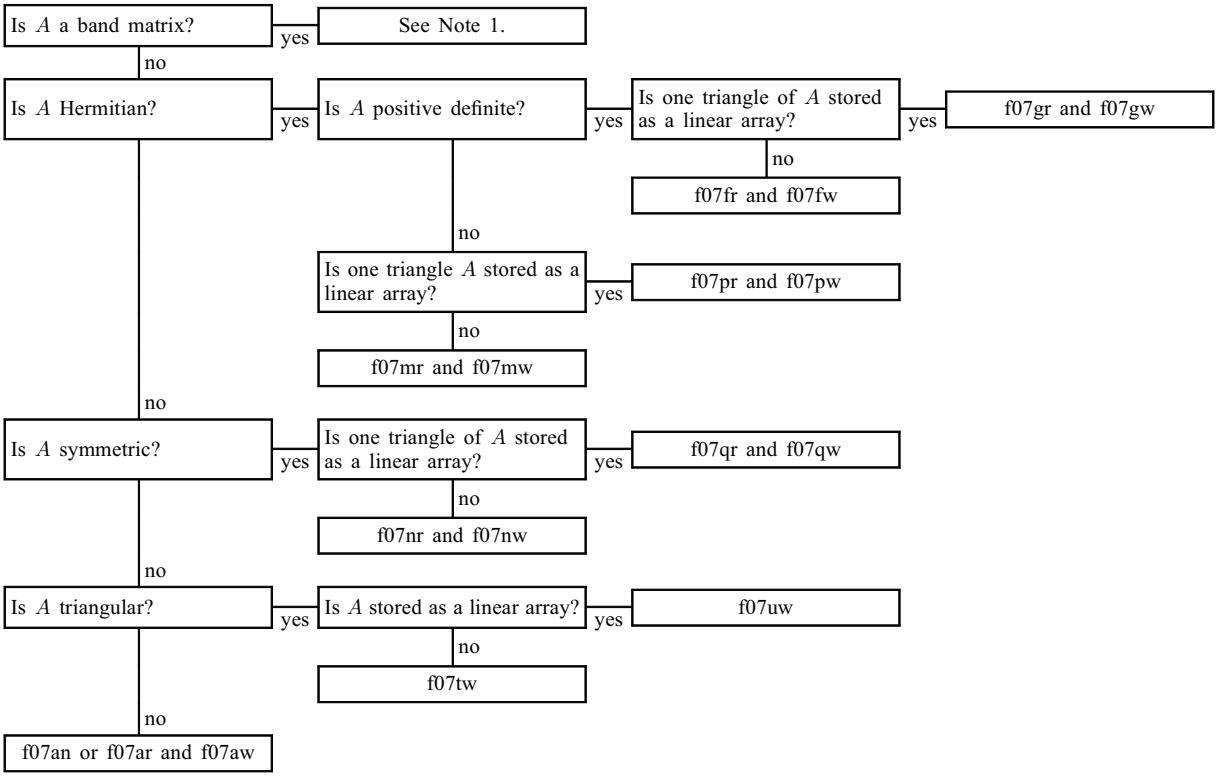
Tree 1



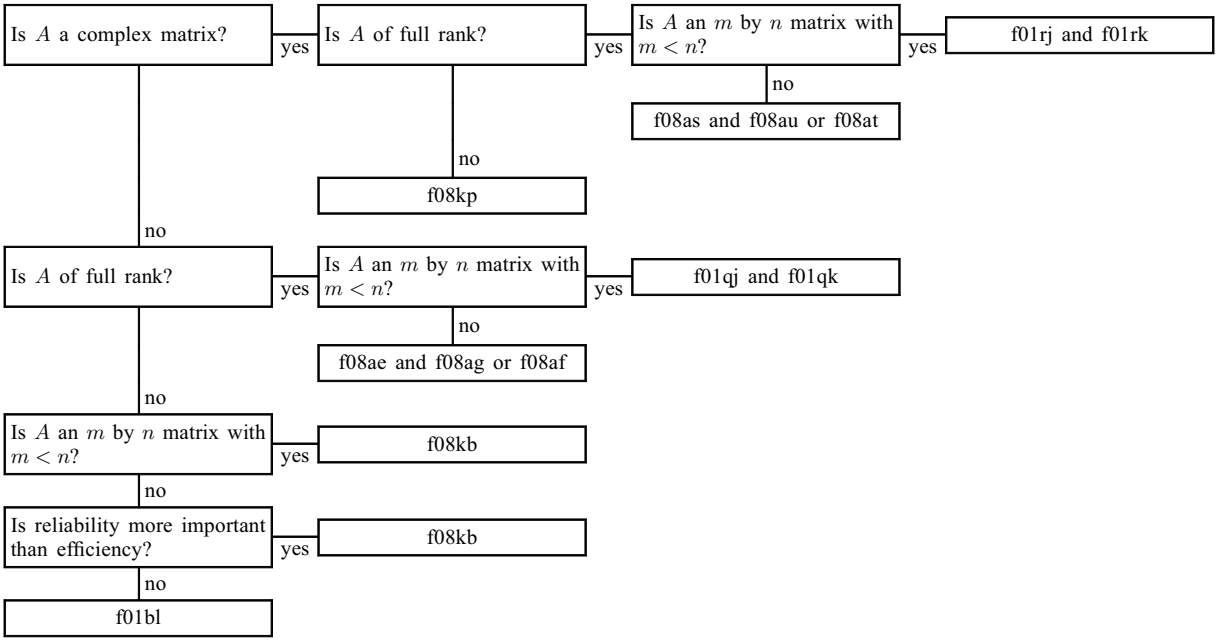
Tree 2: Inverse of a real *n* by *n* matrix of full rank



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



Tree 4: Pseudo-inverses



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.

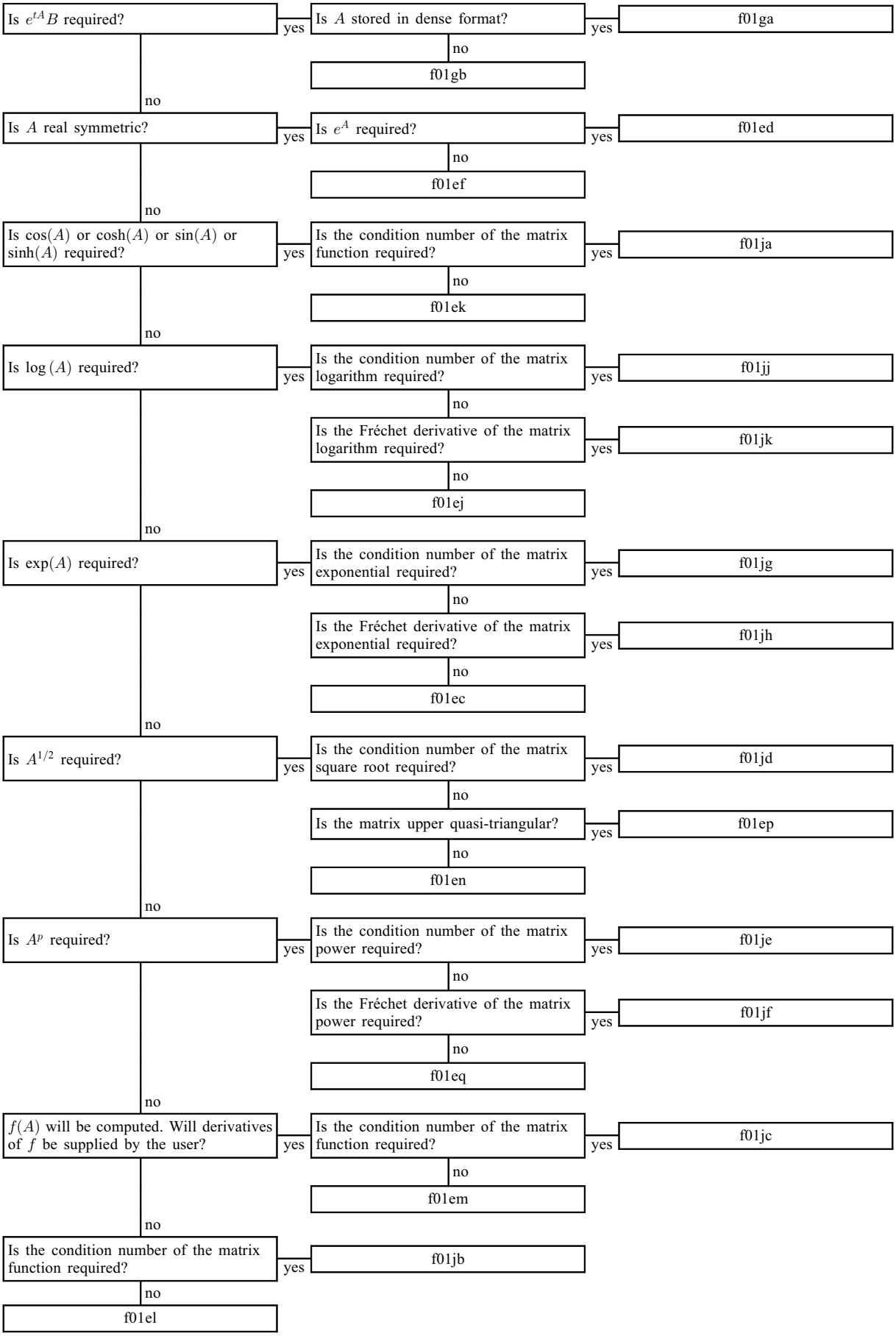
Note 2: by 'guaranteed accuracy' we mean that the accuracy of the inverse is improved by use of the iterative refinement technique using additional precision.

(ii) **Matrix Factorizations:** see the decision trees in Section 4 in the F02 and F04 Chapter Introductions.

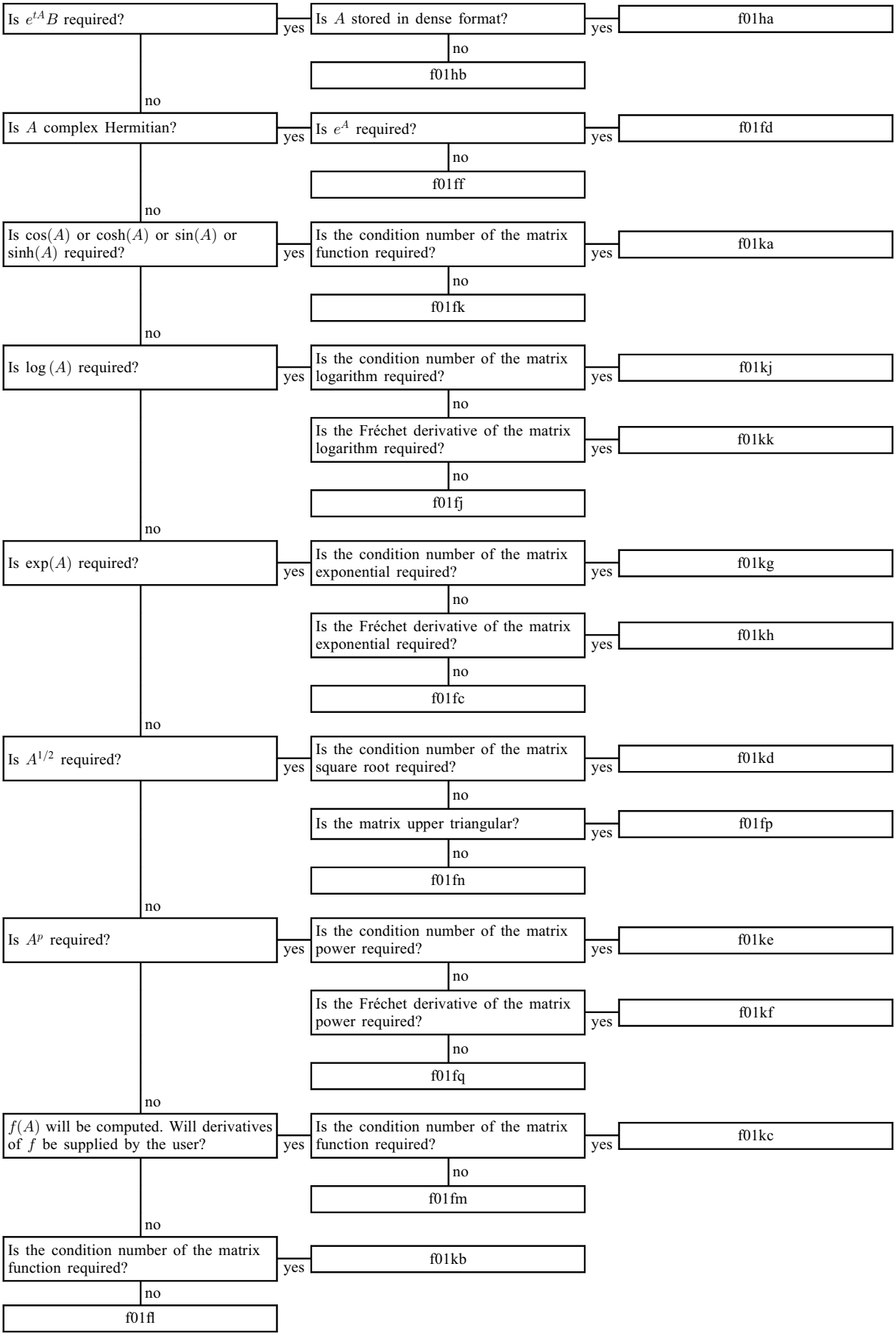
(iii) **Matrix Arithmetic and Manipulation:** not appropriate.

(iv) **Matrix Functions:**

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



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



5 Functionality Index

real sparse matrix,
factorization f01br
factorization, known sparsity pattern..... f01bs
real upper trapezoidal matrix,
RQ factorization..... f01qg
tridiagonal matrix,
LU factorization f01le

6 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