Chapter F02

Eigenvalues and Eigenvectors

Contents

1 Scope of the Chapter

This chapter provides routines for various types of matrix eigenvalue problem:

– standard eigenvalue problems (finding eigenvalues and eigenvectors of a square matrix A)

– singular value problems (finding singular values and singular vectors of a rectangular matrix A)

– generalized eigenvalue problems (finding eigenvalues and eigenvectors of a matrix pencil $A - \lambda B$)

Routines are provided for both real and complex data.

Additional routines for these problems can be found in Chapter F08 which contains software derived from LAPACK (see Anderson *et al.* [1]). However, you should read the introduction to this chapter, F02, before turning to F08, especially if you are a new user.

Chapter F02 contains **Black Box** routines that enable many problems to be solved by a call to a single routine, and the decision trees in Section 4 direct you to the most appropriate routines in Chapter F02. These Black Box routines call routines in Chapter F07 and Chapter F08 wherever possible to perform the computations, and there are pointers in Section 4 to the relevant decision trees in Chapter F08.

2 Background to the Problems

Here we describe the different types of problem which can be tackled by the routines in this chapter, and give a brief outline of the methods used to solve them. If you have one specific type of problem to solve, you need only read the relevant subsection and then turn to Section 3. Consult a standard textbook for a more thorough discussion, for example Golub and Van Loan [2] or Parlett [3].

In each subsection, we first describe the problem in terms of real matrices. The changes needed to adapt the discussion to complex matrices are usually simple and obvious: a matrix transpose such as Q^T must be replaced by its conjugate transpose Q^H; symmetric matrices must be replaced by Hermitian matrices, and orthogonal matrices by unitary matrices. Any additional changes are noted at the end of the subsection.

2.1 Standard Eigenvalue Problems

Let A be a square matrix of order n. The *standard eigenvalue problem* is to find eigenvalues, λ, and corresponding eigenvectors, $x \neq 0$, such that

$$Ax = \lambda x. \tag{1}$$

(The phrase 'eigenvalue problem' is sometimes abbreviated to *eigenproblem*.)

2.1.1 Standard symmetric eigenvalue problems

If A is real symmetric, the eigenvalue problem has many desirable features, and it is advisable to take advantage of symmetry whenever possible.

The eigenvalues λ are all real, and the eigenvectors can be chosen to be mutually orthogonal. That is, we can write

$$Az_i = \lambda_i z_i \quad \text{for} \quad i = 1, 2, \ldots, n$$

or equivalently:

$$AZ = Z\Lambda \tag{2}$$

where Λ is a real diagonal matrix whose diagonal elements λ_i are the eigenvalues, and Z is a real orthogonal matrix whose columns z_i are the eigenvectors. This implies that $z_i^T z_j = 0$ if $i \neq j$, and $\|z_i\|_2 = 1$.

Equation (2) can be rewritten

$$A = Z\Lambda Z^T. \tag{3}$$

This is known as the *eigen-decomposition* or *spectral factorization* of A.

Eigenvalues of a real symmetric matrix are well-conditioned, that is, they are not unduly sensitive to perturbations in the original matrix A. The sensitivity of an eigenvector depends on how small the gap is

between its eigenvalue and any other eigenvalue: the smaller the gap, the more sensitive the eigenvector. More details on the accuracy of computed eigenvalues and eigenvectors are given in the routine documents, and in the F08 Chapter Introduction.

For dense or band matrices, the computation of eigenvalues and eigenvectors proceeds in the following stages:

(1) A is reduced to a symmetric tridiagonal matrix T by an orthogonal similarity transformation: $A = QTQ^T$, where Q is orthogonal. (A *tridiagonal* matrix is zero except for the main diagonal and the first subdiagonal and superdiagonal on either side.) T has the same eigenvalues as A and is easier to handle.

(2) Eigenvalues and eigenvectors of T are computed as required. If all eigenvalues (and optionally eigenvectors) are required, they are computed by the QR algorithm, which effectively factorizes T as $T = S\Lambda S^T$, where S is orthogonal. If only selected eigenvalues are required, they are computed by bisection, and if selected eigenvectors are required, they are computed by inverse iteration. If s is an eigenvector of T, then Qs is an eigenvector of A.

All the above remarks also apply – with the obvious changes – to the case when A is a complex Hermitian matrix. The eigenvectors are complex, but the eigenvalues are all real, and so is the tridiagonal matrix T.

If A is large and sparse, the methods just described would be very wasteful in both storage and computing time, and therefore an alternative algorithm, known as *subspace iteration*, is provided (for real problems only) to find a (usually small) subset of the eigenvalues and their corresponding eigenvectors.

2.1.2 Standard nonsymmetric eigenvalue problems

A real nonsymmetric matrix A may have complex eigenvalues, occurring as complex conjugate pairs. If x is an eigenvector corresponding to a complex eigenvalue λ, then the complex conjugate vector $\bar{x}$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}$. Note that the vector x defined in equation (1) is sometimes called a *right eigenvector*; a *left eigenvector* y is defined by:

$$y^H A = \lambda y^H \quad \text{or} \quad A^T y = \bar{\lambda} y.$$

Routines in this chapter only compute right eigenvectors (the usual requirement), but routines in Chapter F08 can compute left or right eigenvectors or both.

The eigenvalue problem can be solved via the *Schur factorization* of A, defined as

$$A = ZTZ^T,$$

where Z is an orthogonal matrix and T is a real upper *quasi-triangular* matrix, with the same eigenvalues as A. T is called the *Schur form* of A. If all the eigenvalues of A are real, then T is upper triangular, and its diagonal elements are the eigenvalues of A. If A has complex conjugate pairs of eigenvalues, then T has 2 by 2 diagonal blocks, whose eigenvalues are the complex conjugate pairs of eigenvalues of A. (The structure of T is simpler if the matrices are complex – see below.)

For example, the following matrix is in quasi-triangular form

$$\begin{pmatrix} 1 & * & * & * \\ 0 & 2 & -1 & * \\ 0 & 1 & 2 & * \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

and has eigenvalues 1, $2 \pm i$, and 3. (The elements indicated by '$*$' may take any values.)

The columns of Z are called the *Schur vectors*. For each k ($1 \leq k \leq n$), the first k columns of Z form an orthonormal basis for the invariant subspace corresponding to the first k eigenvalues on the diagonal of T. (An *invariant subspace* (for A) is a subspace S such that for any vector v in S, Av is also in S.) Because this basis is orthonormal, it is preferable in many applications to compute Schur vectors rather than eigenvectors. It is possible to order the Schur factorization so that any desired set of k eigenvalues occupy the k leading positions on the diagonal of T, and routines for this purpose are provided in Chapter F08.

Note that if A is symmetric, the Schur vectors are the same as the eigenvectors, but if A is nonsymmetric, they are distinct, and the Schur vectors, being orthonormal, are often more satisfactory to work with in numerical computation.

Eigenvalues and eigenvectors of an nonsymmetric matrix may be ill-conditioned, that is, sensitive to perturbations in A. Chapter F08 contains routines which compute or estimate the condition numbers of eigenvalues and eigenvectors, and the Introduction to that Chapter gives more details about the error analysis of nonsymmetric eigenproblems. The accuracy with which eigenvalues and eigenvectors can be obtained is often improved by *balancing* a matrix. This is discussed further in Section 3.4 below.

Computation of eigenvalues, eigenvectors or the Schur factorization proceeds in the following stages:

(1) A is reduced to an upper Hessenberg matrix H by an orthogonal similarity transformation: $A = QHQ^T$, where Q is orthogonal. (An *upper Hessenberg* matrix is zero below the first subdiagonal.) H has the same eigenvalues as A, and is easier to handle.

(2) The upper Hessenberg matrix H is reduced to Schur form T by the QR algorithm, giving the Schur factorization $H = STS^T$. The eigenvalues of A are obtained from the diagonal blocks of T. The matrix Z of Schur vectors (if required) is computed as $Z = QS$.

(3) After the eigenvalues have been found, eigenvectors may be computed, if required, in two different ways. Eigenvectors of H can be computed by inverse iteration, and then pre-multiplied by Q to give eigenvectors of A; this approach is usually preferred if only a few eigenvectors are required. Alternatively, eigenvectors of T can be computed by back-substitution, and pre-multiplied by Z to give eigenvectors of A.

All the above remarks also apply – with the obvious changes – to the case when A is a complex matrix. The eigenvalues are in general complex, so there is no need for special treatment of complex conjugate pairs, and the Schur form T is simply a complex upper triangular matrix.

2.2 The Singular Value Decomposition

The *singular value decomposition* (SVD) of a real m by n matrix A is given by

$$A = U\Sigma V^T,$$

where U and V are orthogonal and Σ is an m by n diagonal matrix with real diagonal elements, σ_i, such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(m,n)} \geq 0.$$

The σ_i are the *singular values* of A and the first $\min(m,n)$ columns of U and V are, respectively, the *left* and *right singular vectors* of A. The singular values and singular vectors satisfy

$$Av_i = \sigma_i u_i \text{ and } A^T u_i = \sigma_i v_i$$

where u_i and v_i are the ith columns of U and V respectively.

The singular value decomposition of A is closely related to the eigen-decompositions of the symmetric matrices A^TA or AA^T, because:

$$A^TAv_i = \sigma_i^2 v_i \text{ and } AA^T u_i = \sigma_i^2 u_i.$$

However, these relationships are not recommended as a means of computing singular values or vectors.

Singular values are well-conditioned, that is, they are not unduly sensitive to perturbations in A. The sensitivity of a singular vector depends on how small the gap is between its singular value and any other singular value: the smaller the gap, the more sensitive the singular vector. More details on the accuracy of computed singular values and vectors are given in the routine documents and in the the F08 Chapter Introduction.

The singular value decomposition is useful for the numerical determination of the rank of a matrix, and for solving linear least-squares problems, especially when they are rank-deficient (or nearly so). See Chapter F04.

Computation of singular values and vectors proceeds in the following stages:

(1) A is reduced to an upper bidiagonal matrix B by an orthogonal transformation $A = U_1 B V_1^T$, where U_1 and V_1 are orthogonal. (An *upper bidiagonal* matrix is zero except for the main diagonal and the first superdiagonal.) B has the same singular values as A, and is easier to handle.

(2) The SVD of the bidiagonal matrix B is computed as $B = U_2 \Sigma V_2^T$, where U_2 and V_2 are orthogonal and Σ is diagonal as described above. Then in the SVD of A, $U = U_1 U_2$ and $V = V_1 V_2$.

All the above remarks also apply – with the obvious changes – to the case when A is a complex matrix. The singular vectors are complex, but the singular values are real and non-negative, and the bidiagonal matrix B is also real.

2.3 Generalized Eigenvalue Problems

Let A and B be square matrices of order n. The *generalized eigenvalue problem* is to find eigenvalues, λ, and corresponding eigenvectors, $x \neq 0$, such that

$$Ax = \lambda Bx. \tag{4}$$

For given A and B, the set of all matrices of the form $A - \lambda B$ is called a *pencil*, and λ and x are said to be an eigenvalue and eigenvector of the pencil $A - \lambda B$.

When B is non-singular, equation (4) is mathematically equivalent to $(B^{-1}A)x = \lambda x$, and when A is non-singular, it is equivalent to $(A^{-1}B)x = (1/\lambda)x$. Thus, in theory, if one of the matrices A or B is known to be nonsingular, the problem could be reduced to a standard eigenvalue problem.

However, for this reduction to be satisfactory from the point of view of numerical stability, it is necessary not only that B (or A) should be nonsingular, but that it should be well-conditioned with respect to inversion. The nearer B is to singularity, the more unsatisfactory $B^{-1}A$ will be as a vehicle for determining the required eigenvalues. Well-determined eigenvalues of the original problem (4) may be poorly determined even by the correctly rounded version of $B^{-1}A$.

We consider first a special class of problems in which B is known to be non-singular, and then return to the general case in the following subsection.

2.3.1 Generalized symmetric-definite eigenvalue problems

If A and B are *symmetric* and B is *positive-definite*, then the generalized eigenvalue problem has desirable properties similar to those of the standard symmetric eigenvalue problem. The eigenvalues are all real, and the eigenvectors, while not orthogonal in the usual sense, satisfy the relations $z_i^T B z_j = 0$ for $i \neq j$ and can be normalized so that $z_i^T B z_i = 1$.

Note that it is not enough for A and B to be symmetric; B must also be positive-definite, which implies non-singularity. Eigenproblems with these properties are referred to as *symmetric-definite* problems.

If Λ is the diagonal matrix whose diagonal elements are the eigenvalues, and Z is the matrix whose columns are the eigenvectors, then:

$$Z^T A Z = \Lambda \text{ and } Z^T B Z = I.$$

To compute eigenvalues and eigenvectors, the problem can be reduced to a standard symmetric eigenvalue problem, using the Cholesky factorization of B as LL^T or $U^T U$ (see Chapter F07). Note, however, that this reduction does implicitly involve the inversion of B, and hence this approach should *not* be used if B is ill-conditioned with respect to inversion.

For example, with $B = LL^T$, we have

$$Az = \lambda Bz \Leftrightarrow \quad (L^{-1}A(L^{-1})^T)(L^T z) = \lambda (L^T z).$$

Hence the eigenvalues of $Az = \lambda Bz$ are those of $Cy = \lambda y$, where C is the symmetric matrix $C = L^{-1}A(L^{-1})^T$ and $y = L^T z$. The standard symmetric eigenproblem $Cy = \lambda y$ may be solved by the methods described in Section 2.1.1. The eigenvectors z of the original problem may be recovered by computing $z = (L^T)^{-1} y$.

Most of the routines which solve this class of problems can also solve the closely related problems:

$$ABx = \lambda x \text{ or } BAx = \lambda x$$

where again A and B are symmetric and B is positive-definite. See the routine documents for details.

All the above remarks also apply – with the obvious changes – to the case when A and B are complex Hermitian matrices. Such problems are called *Hermitian-definite*. The eigenvectors are complex, but the eigenvalues are all real.

If A and B are large and sparse, reduction to an equivalent standard eigenproblem as described above would almost certainly result in a large dense matrix C, and hence would be very wasteful in both storage and computing time. The method of subspace iteration, mentioned in Section 2.1.1, can also be used for large sparse generalized symmetric-definite problems.

2.3.2 Generalized nonsymmetric eigenvalue problems

Any generalized eigenproblem which is not symmetric-definite with well-conditioned B must be handled as if it were a general nonsymmetric problem.

If B is singular, the problem has infinite eigenvalues. These are not a problem; they are equivalent to zero eigenvalues of the problem $Bx = \mu Ax$. Computationally they appear as very large values.

If A and B are both singular and have a common null-space, then $A - \lambda B$ is singular for all λ; in other words, any value λ can be regarded as an eigenvalue. Pencils with this property are called *singular*.

As with standard nonsymmetric problems, a real problem may have complex eigenvalues, occurring as complex conjugate pairs.

The generalized eigenvalue problem can be solved via the *generalized Schur factorization* of A and B:

$$A = QUZ^T, \quad B = QVZ^T$$

where Q and Z are orthogonal, V is upper triangular, and U is upper quasi-triangular (defined just as in Section 2.1.2).

If all the eigenvalues are real, then U is upper triangular; the eigenvalues are given by $\lambda_i = u_{ii}/v_{ii}$. If there are complex conjugate pairs of eigenvalues, then U has 2 by 2 diagonal blocks.

Eigenvalues and eigenvectors of a generalized nonsymmetric problem may be ill-conditioned, that is, sensitive to perturbations in A or B.

Particular care must be taken if, for some i, $u_{ii} = v_{ii} = 0$, or in practical terms if u_{ii} and v_{ii} are both small; this means that the pencil is singular, or approximately so. Not only is the particular value λ_i undetermined, but also **no reliance** can be placed on **any** of the computed eigenvalues. See also the routine documents.

Computation of eigenvalues and eigenvectors proceeds in the following stages:

(1) The pencil $A - \lambda B$ is reduced by an orthogonal transformation to a pencil $H - \lambda K$ in which H is upper Hessenberg and K is upper triangular: $A = Q_1 H Z_1^T$ and $B = Q_1 K Z_1^T$. The pencil $H - \lambda K$ has the same eigenvalues as $A - \lambda B$, and is easier to handle.

(2) The upper Hessenberg matrix H is reduced to upper quasi-triangular form, while K is maintained in upper triangular form, using the QZ algorithm. This gives the generalized Schur factorization: $H = Q_2 U Z_2$ and $K = Q_2 V Z_2$.

(3) Eigenvectors of the pencil $U - \lambda V$ are computed (if required) by backsubstitution, and pre-multiplied by $Z_1 Z_2$ to give eigenvectors of A.

All the above remarks also apply – with the obvious changes – to the case when A and B are complex matrices. The eigenvalues are in general complex, so there is no need for special treatment of complex conjugate pairs, and the matrix U in the generalized Schur factorization is simply a complex upper triangular matrix.

3 Recommendations on Choice and Use of Available Routines

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

3.1 Black Box Routines and General Purpose Routines

Routines in the NAG Library for solving eigenvalue problems fall into two categories:

Black Box Routines: These are designed to solve a standard type of problem in a single call – for example, to compute all the eigenvalues and eigenvectors of a real symmetric matrix. You are recommended to use a black box routine if there is one to meet your needs; refer to the decision tree in Section 4.1 or the index in Section 5.

General Purpose Routines: These perform the computational subtasks which make up the separate stages of the overall task, as described in Section 2 – for example, reducing a real symmetric matrix to tridiagonal form. General purpose routines are to be found, for historical reasons, some in this Chapter, a few in Chapter F01, but most in Chapter F08. If there is no black box routine that meets your needs, you will need to use one or more general purpose routines.

Here are some of the more likely reasons why you may need to do this:

Your problem is already in one of the reduced forms – for example, your symmetric matrix is already tridiagonal.

You wish to economize on storage for symmetric matrices (see Section 3.3).

You wish to find selected eigenvalues or eigenvectors of a generalized symmetric-definite eigenproblem (see also Section 3.2).

The decision trees in Section 4.2 list the combinations of general purpose routines which are needed to solve many common types of problem.

Sometimes a combination of a black box routine and one or more general purpose routines will be the most convenient way to solve your problem: the black box routine can be used to compute most of the results, and a general purpose routine can be used to perform a subsidiary computation, such as computing condition numbers of eigenvalues and eigenvectors.

3.2 Computing Selected Eigenvalues and Eigenvectors

The decision trees and the routine documents make a distinction between routines which compute *all* eigenvalues or eigenvectors, and routines which compute *selected* eigenvalues or eigenvectors; the two classes of routine use different algorithms.

It is difficult to give clear guidance on which of these two classes of routine to use in a particular case, especially with regard to computing eigenvectors. If you only wish to compute a very few eigenvectors, then a routine for selected eigenvectors will be more economical, but if you want to compute a substantial subset (an old rule of thumb suggested more than 25%), then it may be more economical to compute all of them. Conversely, if you wish to compute all the eigenvectors of a sufficiently large symmetric tridiagonal matrix, the routine for selected eigenvectors may be faster.

The choice depends on the properties of the matrix and on the computing environment; if it is critical, you should perform your own timing tests.

For nonsymmetric eigenproblems, there are no algorithms provided for computing selected eigenvalues; it is always necessary to compute all the eigenvalues, but you can then select specific eigenvectors for computation by inverse iteration.

3.3 Storage Schemes for Symmetric Matrices

Routines which handle symmetric matrices are usually designed to use either the upper or lower triangle of the matrix; it is not necessary to store the whole matrix. If either the upper or lower triangle is stored conventionally in the upper or lower triangle of a two-dimensional array, the remaining elements of the array can be used to store other useful data. However, that is not always convenient, and if it is important to economize on storage, the upper or lower triangle can be stored in a one-dimensional array

of length $n(n+1)/2$ – in other words, the storage is almost halved. This storage format is referred to as *packed storage*.

Routines designed for packed storage are usually less efficient, especially on high-performance computers, so there is a trade-off between storage and efficiency.

A *band* matrix is one whose non-zero elements are confined to a relatively small number of sub-diagonals or super-diagonals on either side of the main diagonal. Algorithms can take advantage of bandedness to reduce the amount of work and storage required.

Routines which take advantage of packed storage or bandedness are provided for both standard symmetric eigenproblems and generalized symmetric-definite eigenproblems.

3.4 Balancing for Nonsymmmetric Eigenproblems

There are two preprocessing steps which one may perform on an nonsymmetric matrix A in order to make its eigenproblem easier. Together they are referred to as *balancing*.

Permutation: This involves reordering the rows and columns to make A more nearly upper triangular (and thus closer to Schur form): $A' = PAP^T$, where P is a permutation matrix. If A has a significant number of zero elements, this preliminary permutation can reduce the amount of work required, and also improve the accuracy of the computed eigenvalues. In the extreme case, if A is permutable to upper triangular form, then no floating-point operations are needed to reduce it to Schur form.

Scaling: A diagonal matrix D is used to make the rows and columns of A' more nearly equal in norm: $A'' = DA'D^{-1}$. Scaling can make the matrix norm smaller with respect to the eigenvalues, and so possibly reduce the inaccuracy contributed by roundoff (see Chap. II/11, of [4]).

Routines are provided in Chapter F08 for performing either or both of these pre-processing steps, and also for transforming computed eigenvectors or Schur vectors back to those of the original matrix.

Black box routines in this chapter which compute the Schur factorization perform only the permutation step, since diagonal scaling is not in general an orthogonal transformation. The black box routines which compute eigenvectors perform both forms of balancing.

3.5 Non-uniqueness of Eigenvectors and Singular Vectors

Eigenvectors, as defined by equations (1) or (4), are *not uniquely defined*. If x is an eigenvector, then so is kx where k is any non-zero scalar. Eigenvectors computed by different algorithms, or on different computers, may appear to disagree completely, though in fact they differ only by a scalar factor (which may be complex). These differences should not be significant in any application in which the eigenvectors will be used, but they can arouse uncertainty about the correctness of computed results.

Even if eigenvectors x are normalized so that $\|x\|_2 = 1$, this is not sufficient to fix them uniquely, since they can still be multiplied by a scalar factor k such that $|k| = 1$. To counteract this inconvenience, most of the routines in this chapter, and in Chapter F08, normalize eigenvectors (and Schur vectors) so that $\|x\|_2 = 1$ *and* the component of x with largest absolute value is real and positive. (There is still a possible indeterminacy if there are two components of equal largest absolute value – or in practice if they are very close – but this is rare.)

In symmetric problems the computed eigenvalues are sorted into ascending order, but in nonsymmetric problems the order in which the computed eigenvalues are returned is dependent on the detailed working of the algorithm and may be sensitive to rounding errors. The Schur form and Schur vectors depend on the ordering of the eigenvalues and this is another possible cause of non-uniqueness when they are computed. However, it must be stressed again that variations in the results from this cause should not be significant. (Routines in Chapter F08 can be used to transform the Schur form and Schur vectors so that the eigenvalues appear in any given order if this is important.)

In singular value problems, the left and right singular vectors u and v which correspond to a singular value σ cannot be normalized independently: if u is multiplied by a factor k such that $|k| = 1$, then v must also be multiplied by k.

Non-uniqueness also occurs among eigenvectors which correspond to a multiple eigenvalue, or among singular vectors which correspond to a multiple singular value. In practice, this is more likely to be apparent as the extreme sensitivity of eigenvectors which correspond to a cluster of close eigenvalues (or of singular vectors which correspond to a cluster of close singular values).

4 Decision Trees

4.1 Black Box routines

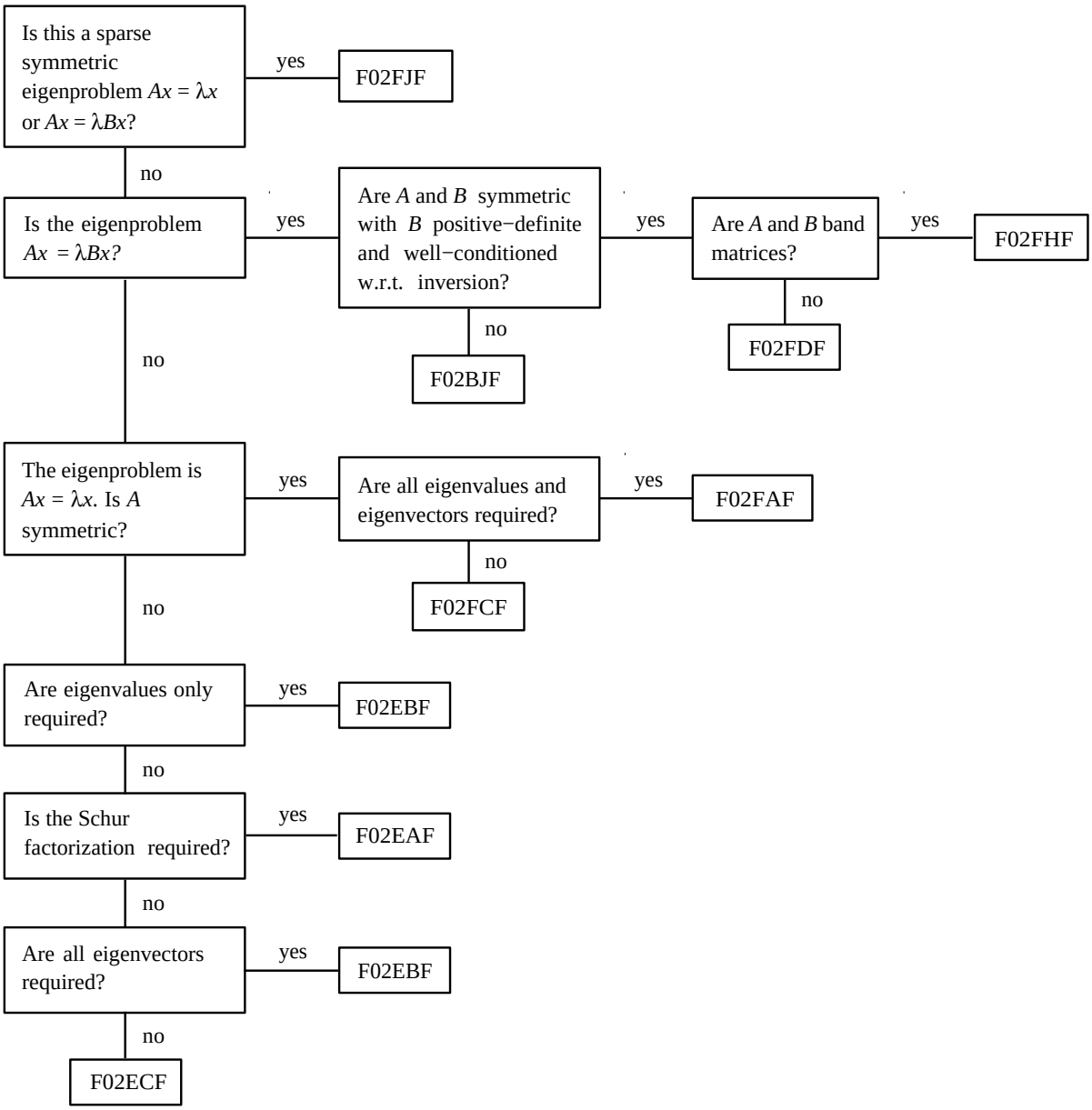
The decision tree for this section is divided into three sub-trees:

Tree 1: Eigenvalues and eigenvectors of real matrices

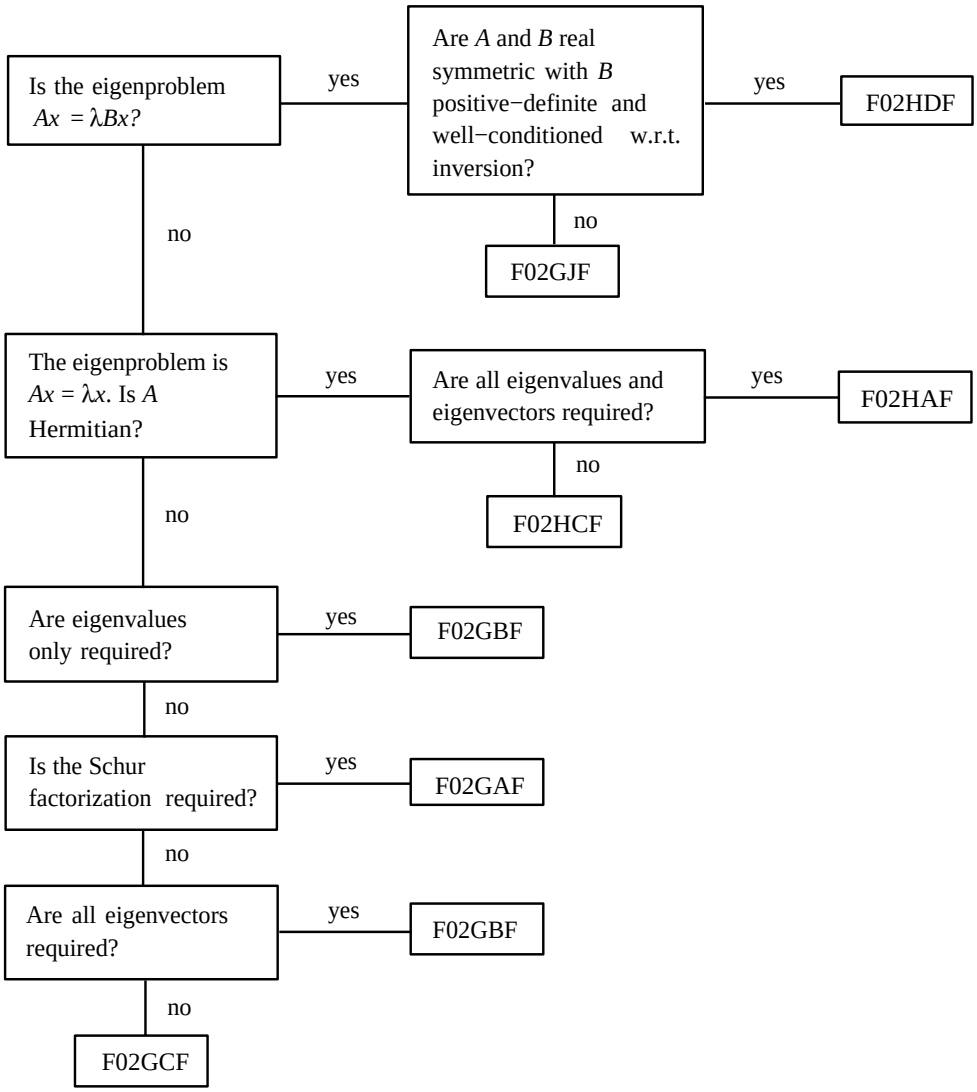
Tree 2: Eigenvalues and eigenvectors of complex matrices

Tree 3: Singular values and singular vectors

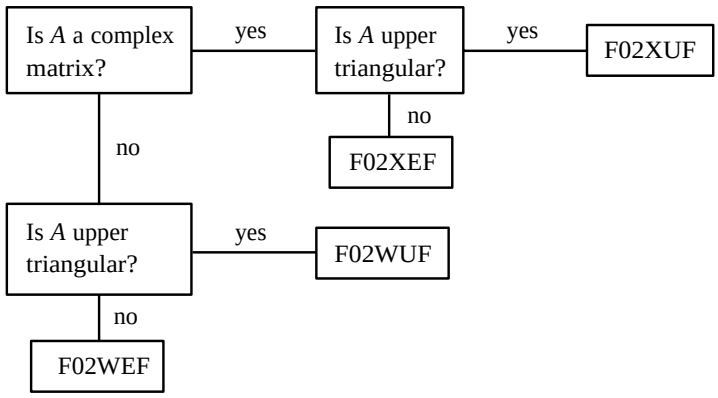
Tree 1: Eigenvalues and Eigenvectors of Real Matrices



Tree 2: Eigenvalues and Eigenvectors of Complex Matrices



Tree 3: Singular Values and Singular Vectors



4.2 General purpose routines (eigenvalues and eigenvectors)

The decision tree for this section is divided into six sub-trees. These are given in Section 4 of the F08 Chapter Introduction:

Tree 1: Real symmetric eigenproblems, $Ax = \lambda x$.

Tree 2: Real generalized symmetric-definite eigenproblems, $Ax = \lambda Bx, ABx =$ or $BAx = \lambda x$.

Tree 3: Real nonsymmetric eigenproblems, $Ax = \lambda x$.

Tree 4: Complex Hermitian eigenproblems, $Ax = \lambda x$.

Tree 5: Complex generalized Hermitian-definite eigenproblems, $Ax = \lambda Bx, ABx = \lambda x$ or $BAx = \lambda x$.

Tree 6: Complex non-Hermitian eigenproblems, $Ax = \lambda x$.

Note. There are no general purpose routines for the problem $Ax = \lambda Bx$, where A and B are real or complex but not otherwise specialised. Use the Black Box routines F02BJF or F02GJF.

As it is very unlikely that one of the routines in this section will be called on its own, the other routines required to solve a given problem are listed in the order in which they should be called.

4.3 General purpose routines (singular value decomposition)

See Section 4.2 of the F08 Chapter Introduction.

5 Index

Black Box Routines

General purpose routines (see also Chapter F08)

6 Routines Withdrawn or Scheduled for Withdrawal

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

F02AAF	F02ABF	F02ADF	F02AEF	F02AFF	F02AGF
F02AJF	F02AKF	F02AMF	F02ANF	F02APF	F02AQF
F02ARF	F02AVF	F02AWF	F02AXF	F02AYF	F02BBF
F02BCF	F02BDF	F02BEF	F02BFF	F02BKF	F02BLF
F02SWF	F02SXF	F02SYF	F02SZF	F02UWF	F02UXF
F02UYF	F02WAF	F02WBF	F02WCF		

7 References

[1] Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S and Sorensen D (1995) *LAPACK Users' Guide* (2nd Edition) SIAM, Philadelphia

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

[3] Parlett B N (1980) *The Symmetric Eigenvalue Problem* Prentice–Hall

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