F02FJF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

To find eigenvalues and eigenvectors of a real sparse symmetric or generalized symmetric eigenvalue problem.

2 Specification

```
      SUBROUTINE F02FJF(N, M, K, NOITS, TOL, DOT, IMAGE, MONIT, NOVECS,
     1                  X, NRX, D, WORK, LWORK, RWORK, LRWORK, IWORK,
     2                  LIWORK, IFAIL)
      INTEGER           N, M, K, NOITS, NOVECS, NRX, LWORK, LRWORK,
     1                  IWORK(LIWORK), LIWORK, IFAIL
      real              TOL, DOT, X(NRX,K), D(K), WORK(LWORK),
     1                  RWORK(LRWORK)
      EXTERNAL          DOT, IMAGE, MONIT
```

3 Description

F02FJF finds the m eigenvalues of largest absolute value and the corresponding eigenvectors for the real eigenvalue problem

$$Cx = \lambda x \tag{1}$$

where C is an n by n matrix such that

$$BC = C^T B \tag{2}$$

for a given positive-definite matrix B. C is said to be B-symmetric. Different specifications of C allow for the solution of a variety of eigenvalue problems. For example, when

$$C = A \text{ and } B = I \text{ where } A = A^T$$

the routine finds the m eigenvalues of largest absolute magnitude for the standard symmetric eigenvalue problem

$$Ax = \lambda x. \tag{3}$$

The routine is intended for the case where A is sparse.

As a second example, when

$$C = B^{-1}A$$

where

$$A = A^T$$

the routine finds the m eigenvalues of largest absolute magnitude for the generalized symmetric eigenvalue problem

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

The routine is intended for the case where A and B are sparse.

The routine does not require C explicitly, but C is specified via a user-supplied routine IMAGE which, given an n element vector z, computes the image w given by

$$w = Cz.$$

For instance, in the above example, where $C = B^{-1}A$, routine IMAGE will need to solve the positive-definite system of equations $Bw = Az$ for w.

To find the m eigenvalues of smallest absolute magnitude of (3) we can choose $C = A^{-1}$ and hence find the reciprocals of the required eigenvalues, so that IMAGE will need to solve $Aw = z$ for w, and correspondingly for (4) we can choose $C = A^{-1}B$ and solve $Aw = Bz$ for w.

A table of examples of choice of IMAGE is given in Table 1. It should be remembered that the routine also returns the corresponding eigenvectors and that B is positive-definite. Throughout A is assumed to be symmetric and, where necessary, non-singularity is also assumed.

Eigenvalues Required	Problem		
	$Ax = \lambda x$ $(B = I)$	$Ax = \lambda Bx$	$ABx = \lambda x$
Largest	Compute $w = Az$	Solve $Bw = Az$	Compute $w = ABz$
Smallest (Find $1/\lambda$)	Solve $Aw = z$	Solve $Aw = Bz$	Solve $Av = z$, $Bw = v$
Furthest from σ (Find $\lambda - \sigma$)	Compute $w = (A-\sigma I)z$	Solve $Bw = (A - \sigma B)z$	Compute $w = (AB - \sigma I)z$
Closest to σ (Find $1/(\lambda - \sigma)$)	Solve $(A - \sigma I)w = z$	Solve $(A - \sigma B)w = Bz$	Solve $(AB - \sigma I)w = z$

Table 1
The Requirement of IMAGE for Various Problems.

The matrix B also need not be supplied explicitly, but is specified via a user-supplied routine DOT which, given n element vectors z and w, computes the generalized dot product $w^T Bz$.

F02FJF is based upon routine SIMITZ (see Nikolai [1]), which is itself a derivative of the Algol procedure ritzit (see Rutishauser [4]), and uses the method of simultaneous (subspace) iteration. (See Parlett [2] for description, analysis and advice on the use of the method.)

The routine performs simultaneous iteration on $k > m$ vectors. Initial estimates to $p \leq k$ eigenvectors, corresponding to the p eigenvalues of C of largest absolute value, may be supplied by the user to F02FJF. When possible k should be chosen so that the kth eigenvalue is not too close to the m required eigenvalues, but if k is initially chosen too small then F02FJF may be re-entered, supplying approximations to the k eigenvectors found so far and with k then increased.

At each major iteration F02FJF solves an r by r $(r \leq k)$ eigenvalue sub-problem in order to obtain an approximation to the eigenvalues for which convergence has not yet occurred. This approximation is refined by Chebyshev acceleration.

4 References

[1] Nikolai P J (1979) Algorithm 538: Eigenvectors and eigenvalues of real generalized symmetric matrices by simultaneous iteration *ACM Trans. Math. Software* **5** 118–125

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

[3] Rutishauser H (1969) Computational aspects of F L Bauer's simultaneous iteration method *Numer. Math.* **13** 4–13

[4] Rutishauser H (1970) Simultaneous iteration method for symmetric matrices *Numer. Math.* **16** 205–223

5 Parameters

1: N — INTEGER *Input*

On entry: n, the order of the matrix C.

Constraint: $N \geq 1$.

2: M — INTEGER *Input/Output*

On entry: m, the number of eigenvalues required.

Constraint: M $\geq$ 1.

On exit: m', the number of eigenvalues actually found. It is equal to m if IFAIL = 0 on exit, and is less than m if IFAIL = 2, 3 or 4. See Section 6 and Section 8 for further information.

3: K — INTEGER *Input*

On entry: the number of simultaneous iteration vectors to be used. Too small a value of K may inhibit convergence, while a larger value of K incurs additional storage and additional work per iteration.

Suggested value: K = M + 4 will often be a reasonable choice in the absence of better information.

Constraint: M < K $\leq$ N.

4: NOITS — INTEGER *Input/Output*

On entry: the maximum number of major iterations (eigenvalue sub-problems) to be performed. If NOITS $\leq$ 0, then the value 100 is used in place of NOITS.

On exit: the number of iterations actually performed.

5: TOL — ***real*** *Input*

On entry: a relative tolerance to be used in accepting eigenvalues and eigenvectors. If the eigenvalues are required to about t significant figures, then TOL should be set to about 10^{-t}. d_i is accepted as an eigenvalue as soon as two successive approximations to d_i differ by less than $(|\tilde{d}_i|\times\mathrm{TOL})/10$, where $\tilde{d}_i$ is the latest approximation to d_i. Once an eigenvalue has been accepted, then an eigenvector is accepted as soon as $(d_i f_i)/(d_i - d_k) < \mathrm{TOL}$, where f_i is the normalised residual of the current approximation to the eigenvector (see Section 8 for further information). The values of the f_i and d_i can be printed from routine MONIT. If TOL is supplied outside the range $(\epsilon, 1.0)$, where ϵ is the ***machine precision***, then the value ϵ is used in place of TOL.

6: DOT — ***real*** FUNCTION, supplied by the user. *External Procedure*

DOT must return the value $w^T Bz$ for given vectors w and z. For the standard eigenvalue problem, where $B = I$, DOT must return the dot product $w^T z$.

Its specification is:

```
real FUNCTION DOT(IFLAG, N, Z, W, RWORK, LRWORK, IWORK, LIWORK)
INTEGER          IFLAG, N, LRWORK, IWORK(LIWORK), LIWORK
real             Z(N), W(N), RWORK(LRWORK)
```

1: IFLAG — INTEGER *Input/Output*

On entry: IFLAG is always non-negative.

On exit: IFLAG may be used as a flag to indicate a failure in the computation of $w^{\mathrm{T}}Bz$. If IFLAG is negative on exit from DOT, then F02FJF will exit immediately with IFAIL set to IFLAG. Note that in this case DOT must still be assigned a value.

2: N — INTEGER *Input*

On entry: the number of elements in the vectors z and w and the order of the matrix B.

3: Z(N) — ***real*** array *Input*

On entry: the vector z for which $w^T Bz$ is required.

4: W(N) — ***real*** array *Input*

On entry: the vector w for which $w^T Bz$ is required.

5: RWORK(LRWORK) — ***real*** array *User Workspace*
6: LRWORK — INTEGER *Input*
7: IWORK(LIWORK) — INTEGER array *User Workspace*
8: LIWORK — INTEGER *Input*

DOT is called from F02FJF with the parameters RWORK, LRWORK, IWORK and LIWORK as supplied to F02FJF. The user is free to use the arrays RWORK and IWORK to supply information to DOT and to IMAGE as an alternative to using COMMON.

DOT must be declared as EXTERNAL in the (sub)program from which F02FJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: IMAGE — SUBROUTINE, supplied by the user. *External Procedure*

IMAGE must return the vector $w = Cz$ for a given vector z.

Its specification is:

```
SUBROUTINE IMAGE(IFLAG, N, Z, W, RWORK, LRWORK, IWORK, LIWORK)
INTEGER         IFLAG, N, LRWORK, IWORK(LIWORK), LIWORK
real            Z(N), W(N), RWORK(LRWORK)
```

1: IFLAG — INTEGER *Input/Output*

On entry: IFLAG is always non-negative.

On exit: IFLAG may be used as a flag to indicate a failure in the computation of w. If IFLAG is negative on exit from IMAGE, then F02FJF will exit immediately with IFAIL set to IFLAG.

2: N — INTEGER *Input*

On entry: n, the number of elements in the vectors w and z, and the order of the matrix C.

3: Z(N) — ***real*** array *Input*

On entry: the vector z for which Cz is required.

4: W(N) — ***real*** array *Output*

On exit: the vector $w = Cz$.

5: RWORK(LRWORK) — ***real*** array *User Workspace*
6: LRWORK — INTEGER *Input*
7: IWORK(LIWORK) — INTEGER array *User Workspace*
8: LIWORK — INTEGER *Input*

IMAGE is called from F02FJF with the parameters RWORK, LRWORK, IWORK and LIWORK as supplied to F02FJF. The user is free to use the arrays RWORK and IWORK to supply information to IMAGE and DOT as an alternative to using COMMON.

IMAGE must be declared as EXTERNAL in the (sub)program from which F02FJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: MONIT — SUBROUTINE, supplied by the user. *External Procedure*

MONIT is used to monitor the progress of F02FJF. MONIT may be the dummy subroutine F02FJZ if no monitoring is actually required. (F02FJZ is included in the NAG Fortran Library and so need not be supplied by the user. The routine name F02FJZ may be implementation dependent: see the Users' Note for your implementation for details.) MONIT is called after the solution of each eigenvalue sub-problem and also just prior to return from F02FJF. The parameters ISTATE and NEXTIT allow selective printing by MONIT.

Its specification is:

```
SUBROUTINE MONIT(ISTATE, NEXTIT, NEVALS, NEVECS, K, F, D)
INTEGER          ISTATE, NEXTIT, NEVALS, NEVECS, K
real             F(K), D(K)
```

1: ISTATE — INTEGER *Input*

On entry: ISTATE specifies the state of F02FJF and will have values as follows:

ISTATE = 0

No eigenvalue or eigenvector has just been accepted.

ISTATE = 1

One or more eigenvalues have been accepted since the last call to MONIT.

ISTATE = 2

One or more eigenvectors have been accepted since the last call to MONIT.

ISTATE = 3

One or more eigenvalues and eigenvectors have been accepted since the last call to MONIT.

ISTATE = 4

Return from F02FJF is about to occur.

2: NEXTIT — INTEGER *Input*

On entry: the number of the next iteration.

3: NEVALS — INTEGER *Input*

On entry: the number of eigenvalues accepted so far.

4: NEVECS — INTEGER *Input*

On entry: the number of eigenvectors accepted so far.

5: K — INTEGER *Input*

On entry: k, the number of simultaneous iteration vectors.

6: F(K) — ***real*** array *Input*

On entry: a vector of error quantities measuring the state of convergence of the simultaneous iteration vectors. See the parameter TOL of F02FJF above and Section 8 for further details. Each element of F is initially set to the value 4.0 and an element remains at 4.0 until the corresponding vector is tested.

7: D(K) — ***real*** array *Input*

On entry: D(i) contains the latest approximation to the absolute value of the ith eigenvalue of C.

MONIT must be declared as EXTERNAL in the (sub)program from which F02FJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

9: NOVECS — INTEGER *Input*

On entry: the number of approximate vectors that are being supplied in X. If NOVECS is outside the range (0,K), then the value 0 is used in place of NOVECS.

10: X(NRX,K) — ***real*** array *Input/Output*

On entry: if $0 <$ NOVECS $\leq$ K, the first NOVECS columns of X must contain approximations to the eigenvectors corresponding to the NOVECS eigenvalues of largest absolute value of C.

Supplying approximate eigenvectors can be useful when reasonable approximations are known, or when the routine is being restarted with a larger value of K. Otherwise it is not necessary to supply approximate vectors, as simultaneous iteration vectors will be generated randomly by the routine.

On exit: if IFAIL = 0, 2, 3 or 4, the first m' columns contain the eigenvectors corresponding to the eigenvalues returned in the first m' elements of D (see below); and the next $k - m' - 1$ columns contain approximations to the eigenvectors corresponding to the approximate eigenvalues returned in the next $k - m' - 1$ elements of D. Here m' is the value returned in M (see above), the number of eigenvalues actually found. The kth column is used as workspace.

11: NRX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F02FJF is called.

Constraint: NRX $\geq$ N.

12: D(K) — ***real*** array *Output*

On exit: if IFAIL = 0, 2, 3 or 4, the first m' elements contain the first m' eigenvalues in decreasing order of magnitude; and the next $k-m'-1$ elements contain approximations to the next $k - m' - 1$ eigenvalues. Here m' is the value returned in M (see above), the number of eigenvalues actually found. D(k) contains the value e where $(-e, e)$ is the latest interval over which Chebyshev acceleration is performed.

13: WORK(LWORK) — ***real*** array *Workspace*

14: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F02FJF is called.

Constraint: LWORK $\geq 3 \times$ K + max(K×K,2×N).

15: RWORK(LRWORK) — ***real*** array *User Workspace*

RWORK is not used by F02FJF, but is passed directly to routines DOT and IMAGE and may be used to supply information to these routines.

16: LRWORK — INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which F02FJF is called.

Constraint: LRWORK $\geq$ 1.

17: IWORK(LIWORK) — INTEGER array *User Workspace*

IWORK is not used by F02FJF, but is passed directly to routines DOT and IMAGE and may be used to supply information to these routines.

18: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F02FJF is called.

Constraint: LIWORK $\geq$ 1.

19: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq$ 0 on exit, users are recommended to set IFAIL to -1 before entry. **It is then essential to test the value of IFAIL on exit**. To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

6 Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from F02FJF because the user has set IFLAG negative in DOT or IMAGE. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL = 1

On entry, N < 1,
or M < 1,
or M ≥ K,
or K > N,
or NRX < N,
or LWORK < 3 × K + max(K×K,2×N),
or LRWORK < 1,
or LIWORK < 1.

IFAIL = 2

Not all the requested eigenvalues and vectors have been obtained. Approximations to the rth eigenvalue are oscillating rapidly indicating that severe cancellation is occurring in the rth eigenvector and so M is returned as $(r - 1)$. A restart with a larger value of K may permit convergence.

IFAIL = 3

Not all the requested eigenvalues and vectors have been obtained. The rate of convergence of the remaining eigenvectors suggests that more than NOITS iterations would be required and so the input value of M has been reduced. A restart with a larger value of K may permit convergence.

IFAIL = 4

Not all the requested eigenvalues and vectors have been obtained. NOITS iterations have been performed. A restart, possibly with a larger value of K, may permit convergence.

IFAIL = 5

This error is very unlikely to occur, but indicates that convergence of the eigenvalue sub-problem has not taken place. Restarting with a different set of approximate vectors may allow convergence. If this error occurs the user should check carefully that F02FJF is being called correctly.

7 Accuracy

Eigenvalues and eigenvectors will normally be computed to the accuracy requested by the parameter TOL, but eigenvectors corresponding to small or to close eigenvalues may not always be computed to the accuracy requested by the parameter TOL. Use of the routine MONIT to monitor acceptance of eigenvalues and eigenvectors is recommended.

8 Further Comments

The time taken by the routine will be principally determined by the time taken to solve the eigenvalue sub-problem and the time taken by the routines DOT and IMAGE. The time taken to solve an eigenvalue sub-problem is approximately proportional to nk^2. It is important to be aware that several calls to DOT and IMAGE may occur on each major iteration.

As can be seen from Table 1, many applications of F02FJF will require routine IMAGE to solve a system of linear equations. For example, to find the smallest eigenvalues of $Ax = \lambda Bx$, IMAGE needs to solve equations of the form $Aw = Bz$ for w and routines from Chapter F01 and Chapter F04 of the NAG Fortran Library will frequently be useful in this context. In particular, if A is a positive-definite variable band matrix, F04MCF may be used after A has been factorized by F01MCF. Thus factorization need be performed only once prior to calling F02FJF. An illustration of this type of use is given in the example program.

An approximation $\tilde{d}_h$, to the ith eigenvalue, is accepted as soon as $\tilde{d}_h$ and the previous approximation differ by less than $|\tilde{d}_h|\times\mathrm{TOL}/10$. Eigenvectors are accepted in groups corresponding to clusters of eigenvalues that are equal, or nearly equal, in absolute value and that have already been accepted. If d_r is the last eigenvalue in such a group and we define the residual r_j as

$$r_j = Cx_j - y_r$$

where y_r is the projection of Cx_j, with respect to B, onto the space spanned by $x_1, x_2, \ldots, x_r$ and x_j is the current approximation to the jth eigenvector, then the value f_i returned in MONIT is given by

$$f_i = \max \|r_j\|_B / \|Cx_j\|_B \quad \|x\|_B^2 = x^T B x$$

and each vector in the group is accepted as an eigenvector if

$$(|d_r| f_r)/(|d_r| - e) < \mathrm{TOL}$$

where e is the current approximation to $|\tilde{d}_k|$. The values of the f_i are systematically increased if the convergence criteria appear to be too strict. See Rutishauser [4] for further details.

The algorithm implemented by F02FJF differs slightly from SIMITZ (Nikolai [1]) in that the eigenvalue sub-problem is solved using the singular value decomposition of the upper triangular matrix R of the Gram–Schmidt factorization of Cx_r, rather than forming $R^T R$.

9 Example

To find the four eigenvalues of smallest absolute value and corresponding eigenvectors for the generalized symmetric eigenvalue problem $Ax = \lambda Bx$, where A and B are the 16 by 16 matrices

$$A = -\frac{1}{4}\begin{pmatrix}
-4 & 1 & & & 1 & & & & & & & & & & & \\
1 & -4 & 1 & & & 1 & & & & & & & & & & \\
 & 1 & -4 & 1 & & & 1 & & & & & & & & & \\
 & & 1 & -4 & 1 & & & 1 & & & & & & & & \\
1 & & & 1 & -4 & 1 & & & 1 & & & & & & & \\
 & 1 & & & 1 & -4 & 1 & & & 1 & & & & & & \\
 & & 1 & & & 1 & -4 & 1 & & & 1 & & & & & \\
 & & & 1 & & & 1 & -4 & 1 & & & 1 & & & & \\
 & & & & 1 & & & 1 & -4 & 1 & & & 1 & & & \\
 & & & & & 1 & & & 1 & -4 & 1 & & & 1 & & \\
 & & & & & & 1 & & & 1 & -4 & 1 & & & 1 & \\
 & & & & & & & 1 & & & 1 & -4 & 1 & & & 1 \\
 & & & & & & & & 1 & & & 1 & -4 & 1 & & \\
 & & & & & & & & & 1 & & & 1 & -4 & 1 & \\
 & & & & & & & & & & 1 & & & 1 & -4 & 1 \\
 & & & & & & & & & & & 1 & & & 1 & -4
\end{pmatrix}$$

$$B = -\frac{1}{2}\begin{pmatrix}
-2 & 1 & & & & & & & & & & & & & & \\
1 & -2 & 1 & & & & & & & & & & & & & \\
 & 1 & -2 & 1 & & & & & & & & & & & & \\
 & & 1 & -2 & 1 & & & & & & & & & & & \\
 & & & 1 & -2 & 1 & & & & & & & & & & \\
 & & & & 1 & -2 & 1 & & & & & & & & & \\
 & & & & & 1 & -2 & 1 & & & & & & & & \\
 & & & & & & 1 & -2 & 1 & & & & & & & \\
 & & & & & & & 1 & -2 & 1 & & & & & & \\
 & & & & & & & & 1 & -2 & 1 & & & & & \\
 & & & & & & & & & 1 & -2 & 1 & & & & \\
 & & & & & & & & & & 1 & -2 & 1 & & & \\
 & & & & & & & & & & & 1 & -2 & 1 & & \\
 & & & & & & & & & & & & 1 & -2 & 1 & \\
 & & & & & & & & & & & & & 1 & -2 & 1 \\
 & & & & & & & & & & & & & & 1 & -2
\end{pmatrix}$$

TOL is taken as 0.0001 and 6 iteration vectors are used. F11JAF is used to factorize the matrix A, prior to calling F02FJF, and F11JCF is used within IMAGE to solve the equations $Aw = Bz$ for w. Details of the factorization of A are passed from F11JAF to F11JCF by means of the COMMON block BLOCK1.

Output from MONIT occurs each time ISTATE is non-zero. Note that the required eigenvalues are the reciprocals of the eigenvalues returned by F02FJF.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F02FJF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NMAX, LA, LRWORK, KMAX, LWORK, LIWORK, NRX
      PARAMETER        (NMAX=16,LA=10*NMAX,LRWORK=1,KMAX=6,
     +                 LWORK=5*KMAX+2*NMAX,LIWORK=2*LA+7*NMAX+1,
     +                 NRX=NMAX)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Scalars in Common ..
      INTEGER          NNZ
*     .. Arrays in Common ..
      real             A(LA)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1)
*     .. Local Scalars ..
      real             DSCALE, DTOL, TOL
      INTEGER          I, IFAIL, J, K, L, LFILL, M, N, NNZC, NOITS,
     +                 NOVECS, NPIVM
      CHARACTER        MIC, PSTRAT
*     .. Local Arrays ..
      real             D(NMAX), RWORK(LRWORK), WORK(LWORK), X(NRX,KMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Functions ..
      real             DOT
      EXTERNAL         DOT
*     .. External Subroutines ..
      EXTERNAL         F02FJF, F02FJZ, F11JAF, IMAGE
*     .. Common blocks ..
      COMMON           /BLOCK1/A, IROW, ICOL, IPIV, ISTR, NNZ
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F02FJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M, K, TOL
      WRITE (NOUT,*)
      IF (N.LT.5 .OR. N.GT.16) THEN
         WRITE (NOUT,99999) 'N is out of range.  N =', N
      ELSE IF (M.LT.1 .OR. M.GE.K .OR. K.GT.KMAX) THEN
         WRITE (NOUT,99999) 'M or K out of range.  M =', M, '   K =', K
      ELSE
*
*        Set up the sparse symmetric coefficient matrix A.
*
         L = 0
         DO 20 I = 1, N
            IF (I.GE.5) THEN
               L = L + 1
               A(L) = -0.25e0
               IROW(L) = I
               ICOL(L) = I - 4
            END IF
            IF (I.GE.2) THEN
               L = L + 1
               A(L) = -0.25e0
               IROW(L) = I
               ICOL(L) = I - 1
            END IF
            L = L + 1
            A(L) = 1.0e0
            IROW(L) = I
            ICOL(L) = I
   20    CONTINUE
         NNZ = L
*
*        Call F11JAF to find an incomplete Cholesky factorisation of A.
*
         LFILL = 2
         DTOL = 0.0e0
         MIC = 'Modified'
         DSCALE = 0.0e0
         PSTRAT = 'Markowitz'
         IFAIL = 1
*
         CALL F11JAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999) 'F11JAF fails. IFAIL =', IFAIL
         ELSE
*
*           Call F02FJF to find eigenvalues and eigenvectors.
            IFAIL = 1
*           * To obtain monitoring information from the supplied
*           subroutine MONIT, replace the name F02FJZ by MONIT in
*           the next statement, and declare MONIT as external *
*
            NOITS = 1000
```

```
            NOVECS = 0
*
            CALL F02FJF(N,M,K,NOITS,TOL,DOT,IMAGE,F02FJZ,NOVECS,X,NRX,D,
     +                  WORK,LWORK,RWORK,LRWORK,IWORK,LIWORK,IFAIL)
*
            IF (IFAIL.NE.0) THEN
               WRITE (NOUT,99999) 'Warning - F02FJF returns IFAIL =',
     +           IFAIL
            END IF
            IF (IFAIL.GE.0 .AND. IFAIL.NE.1 .AND. IFAIL.LE.4 .AND. M.GE.
     +          1) THEN
               DO 40 I = 1, M
                  D(I) = 1.0e0/D(I)
   40          CONTINUE
               WRITE (NOUT,*) 'Final results'
               WRITE (NOUT,*)
               WRITE (NOUT,*) '  Eigenvalues'
               WRITE (NOUT,99998) (D(I),I=1,M)
               WRITE (NOUT,*)
               WRITE (NOUT,*) '  Eigenvectors'
               WRITE (NOUT,99998) ((X(I,J),J=1,M),I=1,N)
            END IF
         END IF
      END IF
   60 STOP
*
99999 FORMAT (1X,A,I5,A,I5)
99998 FORMAT (1X,1P,4e12.3)
      END
*
      real FUNCTION DOT(IFLAG,N,Z,W,RWORK,LRWORK,IWORK,LIWORK)
*     This function implements the dot product - transpose(W)*B*Z.
*     DOT assumes that N is at least 3.
*     .. Scalar Arguments ..
      INTEGER           IFLAG, LIWORK, LRWORK, N
*     .. Array Arguments ..
      real              RWORK(LRWORK), W(N), Z(N)
      INTEGER           IWORK(LIWORK)
*     .. Local Scalars ..
      real              S
      INTEGER           I
*     .. Executable Statements ..
      S = 0.0e0
      S = S + (Z(1)-0.5e0*Z(2))*W(1)
      S = S + (-0.5e0*Z(N-1)+Z(N))*W(N)
      DO 20 I = 2, N - 1
         S = S + (-0.5e0*Z(I-1)+Z(I)-0.5e0*Z(I+1))*W(I)
   20 CONTINUE
      DOT = S
      RETURN
      END
*
```

```
      SUBROUTINE IMAGE(IFLAG,N,Z,W,RWORK,LRWORK,IWORK,LIWORK)
*     This routine solves  A*W = B*Z  for W.
*     The routine assumes that N is at least 3.
*     A, IROW, ICOL, IPIV, ISTR and NNZ must be as returned by routine
*     F11JAF.
*     .. Parameters ..
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=16,LA=10*NMAX,LWORK=6*NMAX+120)
*     .. Scalar Arguments ..
      INTEGER          IFLAG, LIWORK, LRWORK, N
*     .. Array Arguments ..
      real             RWORK(LRWORK), W(N), Z(N)
      INTEGER          IWORK(LIWORK)
*     .. Scalars in Common ..
      INTEGER          NNZ
*     .. Arrays in Common ..
      real             A(LA)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1)
*     .. Local Scalars ..
      real             RNORM, TOL
      INTEGER          IFAIL, ITN, J, MAXITN
      CHARACTER*2      METHOD
*     .. Local Arrays ..
      real             RHS(NMAX), WORK(LWORK)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         F11JCF
*     .. Common blocks ..
      COMMON           /BLOCK1/A, IROW, ICOL, IPIV, ISTR, NNZ
*     .. Executable Statements ..
*
*     Form B*Z in RHS and initialize W to zero.
*
      RHS(1) = Z(1) - 0.5e0*Z(2)
      W(1) = 0.0e0
      RHS(N) = -0.5e0*Z(N-1) + Z(N)
      W(N) = 0.0e0
      DO 20 J = 2, N - 1
         RHS(J) = -0.5e0*Z(J-1) + Z(J) - 0.5e0*Z(J+1)
         W(J) = 0.0e0
   20 CONTINUE
*
*     Call F11JCF to solve the equations  A*W = B*Z.
*
      METHOD = 'CG'
      TOL = X02AJF()
      MAXITN = 100
      IFAIL = 1
*
      CALL F11JCF(METHOD,N,NNZ,A,LA,IROW,ICOL,IPIV,ISTR,RHS,TOL,MAXITN,
     +            W,RNORM,ITN,WORK,LWORK,IFAIL)
*
      IF (IFAIL.GT.0) IFLAG = -IFAIL
      RETURN
      END
*
```

```
      SUBROUTINE MONIT(ISTATE,NEXTIT,NEVALS,NEVECS,K,F,D)
*     Monitoring routine for F02FJF.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalar Arguments ..
      INTEGER          ISTATE, K, NEVALS, NEVECS, NEXTIT
*     .. Array Arguments ..
      real             D(K), F(K)
*     .. Local Scalars ..
      INTEGER          I
*     .. Executable Statements ..
      IF (ISTATE.NE.0) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) '  ISTATE = ', ISTATE, ' NEXTIT = ', NEXTIT
         WRITE (NOUT,99999) '  NEVALS = ', NEVALS, ' NEVECS = ', NEVECS
         WRITE (NOUT,*) '       F           D'
         WRITE (NOUT,99998) (F(I),D(I),I=1,K)
      END IF
      RETURN
*
99999 FORMAT (1X,A,I4,A,I4)
99998 FORMAT (1X,1P,e11.3,3X,e11.3)
      END
```

9.2 Program Data

```
F02FJF Example Program Data
 16 4 6 0.0001
```

9.3 Program Results

```
 F02FJF Example Program Results

 Final results

   Eigenvalues
    5.488E-01   5.900E-01   5.994E-01   6.850E-01

   Eigenvectors
    1.189E-01  -2.153E-01   1.648E-01  -1.561E-01
   -1.378E-01   1.741E-01   1.858E-01   1.931E-01
    1.389E-01   1.626E-01  -1.763E-01  -3.005E-01
   -1.343E-01  -1.602E-01  -2.227E-01   2.058E-01
    2.012E-01  -3.217E-01   3.010E-01  -1.253E-01
   -2.235E-01   2.761E-01   2.954E-01   7.440E-02
    2.242E-01   2.692E-01  -2.899E-01  -2.312E-01
   -2.093E-01  -2.914E-01  -3.320E-01   1.018E-01
    2.093E-01  -2.914E-01   3.320E-01   1.018E-01
   -2.242E-01   2.692E-01   2.899E-01  -2.312E-01
    2.235E-01   2.761E-01  -2.954E-01   7.439E-02
   -2.012E-01  -3.217E-01  -3.010E-01  -1.253E-01
    1.343E-01  -1.602E-01   2.227E-01   2.058E-01
   -1.389E-01   1.626E-01   1.763E-01  -3.005E-01
    1.378E-01   1.741E-01  -1.858E-01   1.931E-01
   -1.189E-01  -2.153E-01  -1.648E-01  -1.561E-01
```