NAG Library Routine Document

F02FKF

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.

Note: *this routine uses* **optional parameters** *to define choices in the problem specification. If you wish to use* default *settings for all of the optional parameters, you need only read Sections 1 to 10 of this document. If, however, you wish to reset some or all of the settings this must be done by calling the option setting routine F12FDF from the user-supplied subroutine OPTION. Please refer to Section 11 for a detailed description of the specification of the optional parameters.*

1 Purpose

F02FKF computes selected eigenvalues and eigenvectors of a real sparse symmetric matrix.

2 Specification

```
SUBROUTINE F02FKF (N, NNZ, A, IROW, ICOL, NEV, NCV, SIGMA, MONIT,          &
                   OPTION, NCONV, W, V, LDV, RESID, IUSER, RUSER, IFAIL)

INTEGER            N, NNZ, IROW(NNZ), ICOL(NNZ), NEV, NCV, NCONV, LDV,     &
                   IUSER(*), IFAIL
REAL (KIND=nag_wp) A(NNZ), SIGMA, W(NCV), V(LDV,*), RESID(NEV),            &
                   RUSER(*)
EXTERNAL           MONIT, OPTION
```

3 Description

F02FKF computes selected eigenvalues and the corresponding right eigenvectors of a real sparse symmetric matrix A:

$$Av_i = \lambda_i v_i.$$

A specified number, n_{ev}, of eigenvalues λ_i, or the shifted inverses $\mu_i = 1/(\lambda_i - \sigma)$, may be selected either by largest or smallest modulus, largest or smallest value, or, largest and smallest values (both ends). Convergence is generally faster when selecting larger eigenvalues, smaller eigenvalues can always be selected by choosing a zero inverse shift ($\sigma = 0.0$). When eigenvalues closest to a given value are required then the shifted inverses of largest magnitude should be selected with shift equal to the required value.

The sparse matrix A is stored in symmetric coordinate storage (SCS) format. See Section 2.1.2 in the F11 Chapter Introduction.

F02FKF uses an implicitly restarted Arnoldi (Lanczos) iterative method to converge approximations to a set of required eigenvalues and corresponding eigenvectors. Further algorithmic information is given in Section 9 while a fuller discussion is provided in the F12 Chapter Introduction. If shifts are to be performed then operations using shifted inverse matrices are performed using a direct sparse solver.

4 References

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

HSL (2011) A collection of Fortran codes for large-scale scientific computation http://www.hsl.rl.ac.uk/

Lehoucq R B, Sorensen D C and Yang C (1998) *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* SIAM, Philidelphia

5 Arguments

1: N – INTEGER *Input*

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

Constraint: $\mathrm{N} > 0$.

2: NNZ – INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which F02FKF is called.The number of nonzero elements in the lower triangular part of the matrix A.

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

3: A(NNZ) – REAL (KIND=nag_wp) array *Input*

On entry: the array of nonzero elements of the lower triangular part of the n by n symmetric matrix A.

4: IROW(NNZ) – INTEGER array *Input*
5: ICOL(NNZ) – INTEGER array *Input*

On entry: the row and column indices of the elements supplied in array A.

If $\mathrm{IROW}(k) = i$ and $\mathrm{ICOL}(k) = j$ then A_{ij} is stored in $\mathrm{A}(k)$. IROW does not need to be ordered, an internal call to F11ZBF forces the correct ordering.

Constraint:

IROW and ICOL must satisfy these constraints:$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.

6: NEV – INTEGER *Input*

On entry: the number of eigenvalues to be computed.

Constraint: $0 < \mathrm{NEV} < \mathrm{N} - 1$.

7: NCV – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which F02FKF is called. The number of Arnoldi basis vectors to use during the computation.

At present there is no *a priori* analysis to guide the selection of NCV relative to NEV. However, it is recommended that $\mathrm{NCV} \geq 2 \times \mathrm{NEV} + 1$. If many problems of the same type are to be solved, you should experiment with increasing NCV while keeping NEV fixed for a given test problem. This will usually decrease the required number of matrix-vector operations but it also increases the work and storage required to maintain the orthogonal basis vectors. The optimal 'cross-over' with respect to computation time is problem dependent and must be determined empirically.

Constraint: $\mathrm{NEV} < \mathrm{NCV} \leq \mathrm{N}$.

8: SIGMA – REAL (KIND=nag_wp) *Input*

On entry: if the **Shifted Inverse** mode has been selected then SIGMA contains the real shift used; otherwise SIGMA is not referenced. This mode can be selected by setting the appropriate options in the user-supplied subroutine OPTION.

9: MONIT – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONIT is used to monitor the progress of F02FKF. MONIT may be the dummy subroutine F02FKZ if no monitoring is actually required. (F02FKZ is included in the NAG Library.) MONIT is called after the solution of each eigenvalue sub-problem and also just prior to return from F02FKF.

The specification of MONIT is:

```
SUBROUTINE MONIT (NCV, NITER, NCONV, W, RZEST, ISTAT, IUSER,        &
                  RUSER)
INTEGER            NCV, NITER, NCONV, ISTAT, IUSER(*)
REAL (KIND=nag_wp) W(NCV), RZEST(NCV), RUSER(*)
```

1: NCV – INTEGER *Input*

On entry: the dimension of the arrays W and RZEST. The number of Arnoldi basis vectors used during the computation.

2: NITER – INTEGER *Input*

On entry: the number of the current Arnoldi iteration.

3: NCONV – INTEGER *Input*

On entry: the number of converged eigenvalues so far.

4: W(NCV) – REAL (KIND=nag_wp) array *Input*

On entry: the first NCONV elements of W contain the converged approximate eigenvalues.

5: RZEST(NCV) – REAL (KIND=nag_wp) array *Input*

On entry: the first NCONV elements of RZEST contain the Ritz estimates (error bounds) on the converged approximate eigenvalues.

6: ISTAT – INTEGER *Input/Output*

On entry: set to zero.

On exit: if set to a nonzero value F02FKF returns immediately with IFAIL $= 9$.

7: IUSER($*$) – INTEGER array *User Workspace*
8: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

MONIT is called with the arguments IUSER and RUSER as supplied to F02FKF. You should use the arrays IUSER and RUSER to supply information to MONIT.

MONIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which F02FKF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: OPTION – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

You can supply non-default options to the Arnoldi eigensolver by repeated calls to F12FDF from within OPTION. (Please note that it is only necessary to call F12FDF; no call to F12FAF is required from within OPTION.) For example, you can set the mode to **Shifted Inverse**, you can increase the **Iteration Limit** beyond its default and you can print varying levels of detail on the iterative process using **Print Level**.

If only the default options (including that the eigenvalues of largest magnitude are sought) are to be used then OPTION may be the dummy subroutine F02EKY (F02EKY is included in the NAG Library). See Section 10 for an example of using OPTION to set some non-default options.

The specification of OPTION is:

```
SUBROUTINE OPTION (ICOMM, COMM, ISTAT, IUSER, RUSER)
INTEGER            ICOMM(*), ISTAT, IUSER(*)
REAL (KIND=nag_wp) COMM(*), RUSER(*)
```

1: ICOMM($*$) – INTEGER array *Communication Array*

On entry: contains details of the default option set. This array must be passed as argument ICOMM in any call to F12FDF.

On exit: contains data on the current options set which may be altered from the default set via calls to F12FDF.

2: COMM($*$) – REAL (KIND=nag_wp) array *Communication Array*

On entry: contains details of the default option set. This array must be passed as argument COMM in any call to F12FDF.

On exit: contains data on the current options set which may be altered from the default set via calls to F12FDF.

3: ISTAT – INTEGER *Input/Output*

On entry: set to zero.

On exit: if set to a nonzero value F02FKF returns immediately with IFAIL $= 10$.

4: IUSER($*$) – INTEGER array *User Workspace*
5: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

OPTION is called with the arguments IUSER and RUSER as supplied to F02FKF. You should use the arrays IUSER and RUSER to supply information to OPTION.

OPTION must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which F02FKF is called.

11: NCONV – INTEGER *Output*

On exit: the number of converged approximations to the selected eigenvalues. On successful exit, this will normally be NEV.

12: W(NCV) – REAL (KIND=nag_wp) array *Output*

On exit: the first NCONV elements contain the converged approximations to the selected eigenvalues.

13: V(LDV, $*$) – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array V must be at least NCV.

On exit: contains the eigenvectors associated with the eigenvalue λ_i, for $i = 1, 2, \ldots,$NCONV (stored in W). For eigenvalue, λ_j, the corresponding eigenvector is stored in V(i, j), for $i = 1, 2, \ldots, n$.

14: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F02FKF is called.

Constraint: LDV $\geq$ N.

15: RESID(NEV) – REAL (KIND=nag_wp) array *Output*

On exit: the residual $\|Aw_i - \lambda_i w_i\|_2$ for the estimates to the eigenpair λ_i and w_i is returned in RESID(i), for $i = 1, 2, \ldots,$NCONV.

16: IUSER($*$) – INTEGER array *User Workspace*

IUSER is not used by F02FKF, but is passed directly to MONIT and OPTION and should be used to pass information to these routines.

17: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

RUSER is not used by F02FKF, but is passed directly to MONIT and OPTION and should be used to pass information to these routines.

18: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N $= \langle value \rangle$.
Constraint: N > 0.

IFAIL $= 2$

On entry, NNZ $= \langle value \rangle$.
Constraint: NNZ > 0.

On entry, NNZ $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: NNZ $\leq$ N $\times$ (N $+ 1)/2$.

IFAIL $= 4$

On entry, for $i = \langle value \rangle$, IROW($i$) $= \langle value \rangle$.
Constraint: $1 \leq$ IROW(i) $\leq$ N.

IFAIL $= 5$

On entry, for $i = \langle value \rangle$, ICOL($i$) $= \langle value \rangle$, IROW(i) $= \langle value \rangle$.
Constraint: $1 \leq$ ICOL(i) $\leq$ IROW(i).

IFAIL $= 6$

On entry, NEV $= \langle value \rangle$.
Constraint: NEV > 0.

On entry, NEV $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: NEV $<$ (N $- 1$).

IFAIL $= 7$

On entry, NCV $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: NCV $\leq$ N.

On entry, NCV $= \langle value \rangle$ and NEV $= \langle value \rangle$.
Constraint: NCV $>$ NEV.

IFAIL $= 8$

On entry, the matrix $(A - \sigma I)$ is numerically singular and could not be inverted. Try perturbing the value of σ.

IFAIL $= 9$

User requested termination in MONIT, ISTAT $= \langle value \rangle$.

IFAIL $= 10$

User requested termination in OPTION, ISTAT $= \langle value \rangle$.

IFAIL $= 14$

On entry, LDV $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDV $\geq$ N.

IFAIL $= 20$

The maximum number of iterations, through the optional parameter **Iteration Limit**, has been set to a non-positive value.

IFAIL $= 21$

The option **Both Ends** has been set but only 1 eigenvalue is requested.

IFAIL $= 22$

The maximum number of iterations has been reached.
The maximum number of iterations $= \langle value \rangle$.
The number of converged eigenvalues $= \langle value \rangle$.
See the routine document for further details.

IFAIL $= 30$

A serious error, code $\langle value \rangle$, has occurred in an internal call to $\langle value \rangle$. Check all subroutine calls and array sizes. If the call is correct then please contact NAG for assistance.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The relative accuracy of a Ritz value (eigenvalue approximation), λ, is considered acceptable if its Ritz estimate $\leq$ **Tolerance** $\times\ \lambda$. The default value for **Tolerance** is the ***machine precision*** given by X02AJF. The Ritz estimates are available via the MONIT subroutine at each iteration in the Arnoldi process, or can be printed by setting option **Print Level** to a positive value.

8 Parallelism and Performance

F02FKF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F02FKF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

F02FKF calls routines based on the ARPACK suite in Chapter F12. These routines use an implicitly restarted Lanczos iterative method to converge to approximations to a set of required eigenvalues (see the F12 Chapter Introduction).

In the default **Regular** mode, only matrix-vector multiplications are performed using the sparse matrix A during the Lanczos process; F11XEF can be used to perform this task. Each iteration is therefore cheap computationally, relative to the alternative, **Shifted Inverse**, mode described below. It is most efficient (i.e., the total number of iterations required is small) when the eigenvalues of largest magnitude are sought and these are distinct.

Although there is an option for returning the smallest eigenvalues using this mode (see **Smallest Magnitude** option), the number of iterations required for convergence will be far greater or the method may not converge at all. However, where convergence is achieved, **Regular** mode may still prove to be the most efficient since no inversions are required. Where smallest eigenvalues are sought and **Regular** mode is not suitable, or eigenvalues close to a given real value are sought, the **Shifted Inverse** mode should be used.

If the **Shifted Inverse** mode is used (via a call to F12FDF in OPTION) then the matrix $A - \sigma I$ is used in linear system solves by the Lanczos process. This is first factorized internally using a direct sparse LDL^{T} factorization under the assumption that the matrix is indefinite. If the factorization determines that the matrix is numerically singular then the routine exits with an error. In this situation it is normally sufficient to perturb σ by a small amount and call F02FKF again. After successful factorization, subsequent solves are performed by backsubstitution using the sparse factorization.

Finally, F02FKF transforms the eigenvectors. Each eigenvector w is normalized so that $\|w\|_2 = 1$.

The monitoring routine MONIT provides some basic information on the convergence of the Lanczos iterations. Much greater levels of detail on the Lanczos process are available via option **Print Level**. If this is set to a positive value then information will be printed, by default, to standard output. The destination of monitoring information can be changed using the **Monitoring** option.

9.1 Additional Licensor

The direct sparse LDL^{T} factorization performed in **Shifted Inverse** mode is performed by an implementation of HSL_MA97 (see HSL (2011)).

10 Example

This example solves $Ax = \lambda x$ in **Shifted Inverse** mode, where A is obtained from the standard central difference discretization of the one-dimensional Laplacian operator $\frac{\partial^2 u}{\partial x^2}$ on $[0, 1]$, with zero Dirichlet boundary conditions.

10.1 Program Text

```
!   F02FKF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module f02fkfe_mod

!     F02FKF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: mymonit, myoption
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6
    Contains
      Subroutine myoption(icomm,comm,istat,iuser,ruser)

!       .. Use Statements ..
        Use nag_library, Only: f12fdf
!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (Inout)        :: istat
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: comm(*), ruser(*)
        Integer, Intent (Inout)        :: icomm(*), iuser(*)
!       .. Local Scalars ..
        Integer                        :: ifail1
        Character (25)                 :: rec
!       .. Intrinsic Procedures ..
        Intrinsic                      :: max
!       .. Executable Statements ..
        Continue

        istat = 0

        If (iuser(1)>0) Then
          Write (rec,99999) 'Print Level=', iuser(1)
          ifail1 = 1
          Call f12fdf(rec,icomm,comm,ifail1)
          istat = max(istat,ifail1)
        End If
        If (iuser(2)>100) Then
          Write (rec,99999) 'Iteration Limit=', iuser(2)
          ifail1 = 1
          Call f12fdf(rec,icomm,comm,ifail1)
          istat = max(istat,ifail1)
        End If
        If (iuser(3)>0) Then
          ifail1 = 1
          Call f12fdf('Shifted Inverse',icomm,comm,ifail1)
          istat = max(istat,ifail1)
        End If
99999   Format (A,I5)
      End Subroutine myoption

      Subroutine mymonit(ncv,niter,nconv,w,rzest,istat,iuser,ruser)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (Inout)        :: istat
        Integer, Intent (In)           :: nconv, ncv, niter
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: rzest(ncv), w(ncv)
        Integer, Intent (Inout)        :: iuser(*)
```

```
!       .. Local Scalars ..
        Integer                        :: i
!       .. Executable Statements ..
        Continue

        If (iuser(4)>0) Then
          If (niter==1 .And. iuser(3)>0) Then
            Write (nout,99999) ' Arnoldi basis vectors used:', ncv
            Write (nout,*)                                                 &
              ' The following Ritz values (mu) are related to the'
            Write (nout,*)                                                 &
              ' true eigenvalues (lambda) by lambda = sigma + 1/mu'
          End If
          Write (nout,*)
          Write (nout,99999) ' Iteration number ', niter
          Write (nout,99998) ' Ritz values converged so far (', nconv,    &
            ') and their Ritz estimates:'
          Do i = 1, nconv
            Write (nout,99997) i, w(i), rzest(i)
          End Do
          Write (nout,*) ' Next (unconverged) Ritz value:'
          Write (nout,99996) nconv + 1, w(nconv+1)
        End If
        istat = 0
99999   Format (1X,A,I4)
99998   Format (1X,A,I4,A)
99997   Format (1X,1X,I4,1X,E13.5,1X,E13.5)
99996   Format (1X,1X,I4,1X,E13.5)
      End Subroutine mymonit
    End Module f02fkfe_mod
    Program f02fkfe

!     Example problem for F02FKF.

!     .. Use Statements ..
      Use nag_library, Only: f02fkf, nag_wp, x04abf, x04caf
      Use f02fkfe_mod, Only: mymonit, myoption, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Integer, Parameter               :: iset = 1
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: h2, sigma
      Integer                          :: i, ifail, imon, j, k, ldv, lo,      &
                                          maxit, mode, n, nconv, ncv, nev,    &
                                          nnz, nx, outchn, prtlvl
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), d_print(:,:), resid(:),       &
                                          v(:,:), w(:)
      Real (Kind=nag_wp)               :: ruser(1)
      Integer, Allocatable             :: icol(:), irow(:)
      Integer                          :: iuser(4)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: real
!     .. Executable Statements ..
      Write (nout,*) 'F02FKF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

      Read (nin,*) nx
      Read (nin,*) nev
      Read (nin,*) ncv
      Read (nin,*) sigma

!     Construct the matrix A in sparse form and store in A.
!     The main diagonal of A is full and there are two subdiagonals of A:
!     the first and the nx-th.
```

```
      n = nx*nx
      nnz = 3*n - 2*nx
      Allocate (a(nnz),irow(nnz),icol(nnz))

!     Zero out A.

      a(1:nnz) = 0.0_nag_wp

!     Main diagonal of A.
      h2 = one/real((nx+1)*(nx+1),kind=nag_wp)
      a(1:n) = 4.0_nag_wp/h2
      Do i = 1, n
        irow(i) = i
        icol(i) = i
      End Do

!     First subdiagonal of A.
      k = n
      Do i = 1, nx
        lo = (i-1)*nx
        Do j = lo + 1, lo + nx - 1
          k = k + 1
          irow(k) = j + 1
          icol(k) = j
          a(k) = -one/h2
        End Do
      End Do

!     nx-th subdiagonal
      Do i = 1, nx - 1
        lo = (i-1)*nx
        Do j = lo + 1, lo + nx
          k = k + 1
          irow(k) = j + nx
          icol(k) = j
          a(k) = -one/h2
        End Do
      End Do

!     Set some options via iuser array and routine argument OPTION.
!     iuser(1) = print level, iuser(2) = iteration limit,
!     iuser(3)>0 means shifted-invert mode
!     iuser(4)>0 means print monitoring info

      Read (nin,*) prtlvl
      Read (nin,*) maxit
      Read (nin,*) mode
      Read (nin,*) imon
      ruser(1) = one
      iuser(1) = prtlvl
      iuser(2) = maxit
      iuser(3) = mode
      iuser(4) = imon

!     Find eigenvalues of largest magnitude and the corresponding
!     eigenvectors.

      ldv = n
      Allocate (w(ncv),v(ldv,ncv),resid(n))

      ifail = -1
      Call f02fkf(n,nnz,a,irow,icol,nev,ncv,sigma,mymonit,myoption,nconv,w,v,  &
        ldv,resid,iuser,ruser,ifail)

      If (ifail/=0) Then
        Go To 100
      End If

!     Print Eigenvalues and the residual norm  ||A*x - lambda*x||.

      Allocate (d_print(nconv,2))
      d_print(1:nconv,1) = w(1:nconv)
      d_print(1:nconv,2) = resid(1:nconv)

      Write (nout,*)
      Flush (nout)

      outchn = nout
      Call x04abf(iset,outchn)

      ifail = 0
      Call x04caf('G','N',nconv,2,d_print,nconv,' Ritz values and residuals',  &
        ifail)

100   Continue
    End Program f02fkfe
```

10.2 Program Data

```
F02FKF Example Program Data
 20       : nx, matrix order n = nx*nx
 8        : nev, number of eigenvalues requested
 20       : ncv, size of subspace
 1.0      : sigma, shift (want eigenvalues close to sigma)
 0        : print level
 500      : maximum number of itrerations
 1        : mode (0 = regular, 1 = shifted inverse)
 0        : imon (0 = no monitoring, 1 = monitoring on)
```

10.3 Program Results

```
 F02FKF Example Program Results


  Ritz values and residuals
              1           2
 1    1.9702E+01   5.6583E-13
 2    4.9036E+01   7.2456E-13
 3    4.9036E+01   7.0022E-13
 4    7.8370E+01   8.8089E-13
 5    9.7197E+01   8.7693E-13
 6    9.7197E+01   8.7672E-13
 7    1.2653E+02   6.8423E-13
 8    1.2653E+02   9.8310E-13
```

11 Optional Parameters

Internally F02FKF calls routines from the suite F12FAF, F12FBF, F12FCF, F12FDF and F12FEF. Several optional parameters for these computational routines define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of F02FKF these optional parameters are also used here and have associated *default values* that are usually appropriate. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

Optional parameters may be specified via the user-supplied subroutine OPTION in the call to F02FKF. OPTION must be coded such that one call to F12FDF is necessary to set each optional parameter. All optional parameters you do not specify are set to their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 11.1.

Advisory

Both Ends

Defaults

Iteration Limit
Largest Algebraic
Largest Magnitude
List
Monitoring
Nolist
Print Level
Regular
Regular Inverse
Shifted Inverse
Smallest Algebraic
Smallest Magnitude
Tolerance

11.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined;

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for ***machine precision*** (see X02AJF).

Keywords and character values are case and white space insensitive.

Advisory i Default $=$ the value returned by X04ABF

If the optional parameter **List** is set then optional parameter specifications are listed in a **List** ***file*** by setting the option to a file identification (unit) number associated with **Advisory** messages (see X04ABF and X04ACF).

Defaults

This special keyword may be used to reset all optional parameters to their default values.

Iteration Limit i Default $= 300$

The limit on the number of Lanczos iterations that can be performed before F12FBF exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Lanczos iterations has reached this limit then F12FBF exits with an error; F12FCF can still be called subsequently to return the number of converged eigenvalues, the converged eigenvalues and, if requested, the corresponding eigenvectors.

Largest Magnitude Default
Both Ends
Largest Algebraic
Smallest Algebraic
Smallest Magnitude

The Lanczos iterative method converges on a number of eigenvalues with given properties. The default is for F12FBF to compute the eigenvalues of largest magnitude using **Largest Magnitude**. Alternatively, eigenvalues may be chosen which have **Largest Algebraic** part, **Smallest Magnitude**, or **Smallest Algebraic** part; or eigenvalues which are from **Both Ends** of the algebraic spectrum.

Nolist Default
List

Normally each optional parameter specification is not listed as it is supplied. This behaviour can be changed using the **List** and **Nolist** options.

Monitoring i Default $= -1$

If $i > 0$, monitoring information is output to channel number i during the solution of each problem; this may be the same as the **Advisory** channel number. The type of information produced is dependent on the value of **Print Level**, see the description of the optional parameter **Print Level** for details of the information produced. Please see X04ACF to associate a file with a given channel number.

Print Level i Default $= 0$

This controls the amount of printing produced by F02FKF as follows.

$= 0$	No output except error messages. If you want to suppress all output, set **Print Level** $= 0$.
> 0	The set of selected options.
$= 2$	Problem and timing statistics on final exit from F12FBF.
≥ 5	A single line of summary output at each Lanczos iteration.
≥ 10	If **Monitoring** is set, then at each iteration, the length and additional steps of the current Lanczos factorization and the number of converged Ritz values; during re-orthogonalization, the norm of initial/restarted starting vector; on a final Lanczos iteration, the number of update iterations taken, the number of converged eigenvalues, the converged eigenvalues and their Ritz estimates.
≥ 20	Problem and timing statistics on final exit from F12FBF. If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the number of shifts being applied, the eigenvalues and estimates of the symmetric tridiagonal matrix H, the size of the Lanczos basis, the wanted Ritz values and associated Ritz estimates and the shifts applied; vector norms prior to and following re-orthogonalization.
≥ 30	If **Monitoring** > 0, **Monitoring** is set, then on final iteration, the norm of the residual; when computing the Schur form, the eigenvalues and Ritz estimates both before and after sorting; for each iteration, the norm of residual for compressed factorization and the symmetric tridiagonal matrix H; during re-orthogonalization, the initial/restarted starting vector; during the Lanczos iteration loop, a restart is flagged and the number of the residual requiring iterative refinement; while applying shifts, some indices.
≥ 40	If **Monitoring** > 0, **Monitoring** is set, then during the Lanczos iteration loop, the Lanczos vector number and norm of the current residual; while applying shifts, key measures of progress and the order of H; while computing eigenvalues of H, the last rows of the Schur and eigenvector matrices; when computing implicit shifts, the eigenvalues and Ritz estimates of H.
≥ 50	If **Monitoring** is set, then during Lanczos iteration loop: norms of key components and the active column of H, norms of residuals during iterative refinement, the final symmetric tridiagonal matrix H; while applying shifts: number of shifts, shift values, block indices, updated tridiagonal matrix H; while computing eigenvalues of H: the diagonals of H, the computed eigenvalues and Ritz estimates.

Note that setting **Print Level** ≥ 30 can result in very lengthy **Monitoring** output.

Regular Default
Regular Inverse
Shifted Inverse

These options define the computational mode which in turn defines the form of operation OP(x) to be performed.

Regular	OP $= A$
Shifted Inverse	OP $= (A - \sigma I)^{-1}$ where σ is real
Regular Inverse	OP $= A^{-1}$

Tolerance r Default $= \epsilon$

An approximate eigenvalue has deemed to have converged when the corresponding Ritz estimate is within **Tolerance** relative to the magnitude of the eigenvalue.