NAG Library Routine Document

F02EKF

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 F12ADF from the user-supplied subroutine OPTION. Please refer to Section 11 for a detailed description of the specification of the optional parameters*.

1 Purpose

F02EKF computes selected eigenvalues and eigenvectors of a real sparse general matrix.

2 Specification

```
SUBROUTINE F02EKF (N, NNZ, A, ICOLZP, IROWIX, NEV, NCV, SIGMA, MONIT,      &
                   OPTION, NCONV, W, V, LDV, RESID, IUSER, RUSER, IFAIL)

INTEGER                 N, NNZ, ICOLZP(N+1), IROWIX(NNZ), NEV, NCV,        &
                        NCONV, LDV, IUSER(*), IFAIL
REAL (KIND=nag_wp)      A(NNZ), SIGMA, V(LDV,*), RESID(NEV+1), RUSER(*)
COMPLEX (KIND=nag_wp) W(NCV)
EXTERNAL                MONIT, OPTION
```

3 Description

F02EKF computes selected eigenvalues and the corresponding right eigenvectors of a real sparse general matrix A:

$$Aw_i = \lambda_i w_i.$$

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 real part, or, largest or smallest imaginary part. 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 real value are required then the shifted inverses of largest magnitude should be selected with shift equal to the required real value.

Note that even though A is real, λ_i and w_i may be complex. If w_i is an eigenvector corresponding to a complex eigenvalue λ_i, then the complex conjugate vector $\bar{w}_i$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}_i$. The eigenvalues in a complex conjugate pair λ_i and $\bar{\lambda}_i$ are either both selected or both not selected.

The sparse matrix A is stored in compressed column storage (CCS) format. See Section 2.1.3 in the F11 Chapter Introduction.

F02EKF uses an implicitly restarted Arnoldi 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; further information on the solver used is provided in the F11 Chapter Introduction.

4 References

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

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: $\mathrm{N} \geq 0$.

2: NNZ – INTEGER *Input*

On entry: the dimension of the array A and The number of nonzero elements of the matrix A and, if a nonzero shifted inverse is to be applied, all diagonal elements. Each nonzero is counted once in the latter case.

Constraint: $0 \leq \mathrm{NNZ} \leq \mathrm{N}^2$.

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

On entry: the array of nonzero elements (and diagonal elements if a nonzero inverse shift is to be applied) of the n by n general matrix A.

On exit: if a nonzero shifted inverse is to be applied then the diagonal elements of A have the shift value, as supplied in SIGMA, subtracted.

4: ICOLZP(N + 1) – INTEGER array *Input*

On entry: ICOLZP(i) contains the index in A of the start of column i, for $i = 1, 2, \ldots, n$; ICOLZP(N + 1) must contain the value NNZ + 1. Thus the number of nonzero elements in column i of A is ICOLZP($i + 1$) − ICOLZP(i); when shifts are applied this includes diagonal elements irrespective of value. See Section 2.1.3 in the F11 Chapter Introduction.

5: IROWIX(NNZ) – INTEGER array *Input*

On entry: IROWIX(i) contains the row index for each entry in A. See Section 2.1.3 in the F11 Chapter Introduction.

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 F02EKF 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 CPU time is problem dependent and must be determined empirically.

Constraint: $\mathrm{NEV} + 1 < \mathrm{NCV} \leq \mathrm{N}$.

8: SIGMA – REAL (KIND=nag_wp) *Input*

On entry: if the **Shifted Inverse Real** 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 F02EKF. MONIT may be the dummy subroutine F02EKZ if no monitoring is actually required. (F02EKZ is included in the NAG Library.) MONIT is called after the solution of each eigenvalue sub-problem and also just prior to return from F02EKF.

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)      RZEST(NCV), RUSER(*)
COMPLEX (KIND=nag_wp) W(NCV)
```

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) – COMPLEX (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 F02EKF 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 F02EKF. 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 F02EKF 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 F12ADF from within OPTION. (Please note that it is only necessary to call F12ADF; no call to F12AAF is

required from within OPTION.) For example, you can set the mode to **Shifted Inverse Real**, 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 F12ADF.

On exit: contains data on the current options set which may be altered from the default set via calls to F12ADF.

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 F12ADF.

On exit: contains data on the current options set which may be altered from the default set via calls to F12ADF.

3: ISTAT – INTEGER *Input/Output*

On entry: set to zero.

On exit: if set to a nonzero value F02EKF 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 F02EKF. 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 F02EKF is called.

11: NCONV – INTEGER *Output*

On exit: the number of converged approximations to the selected eigenvalues. On successful exit, this will normally be either NEV or NEV $+ 1$ depending on the number of complex conjugate pairs of eigenvalues returned.

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

On exit: the first NCONV elements contain the converged approximations to the selected eigenvalues. Since complex conjugate pairs of eigenvalues appear together, it is possible (given an odd number of converged real eigenvalues) for F02EKF to return one more eigenvalue than requested.

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 a real eigenvalue, λ_j, the corresponding eigenvector is real and is stored in

V(i,j), for $i = 1,2,\ldots,n$. For complex conjugate pairs of eigenvalues, $w_{j+1} = \bar{w}_j$, the real and imaginary parts of the corresponding eigenvectors are stored, respectively, in V(i,j) and V(i,j), for $i = 1,2,\ldots,n$. The imaginary parts stored are for the first of the conjugate pair of eigenvectors; the other eigenvector in the pair is obtained by negating these imaginary parts.

14: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F02EKF is called.

Constraint: LDV $\geq$ N.

15: RESID(NEV + 1) – 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 F02EKF, 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 F02EKF, 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.

IFAIL $= 3$

On entry, in shifted inverse mode, the jth diagonal element of A is not defined, for $j = \langle value \rangle$.

IFAIL $= 4$

On entry, for $i = \langle value \rangle$, ICOLZP$(i) = \langle value \rangle$ and ICOLZP$(i+1) = \langle value \rangle$.
Constraint: ICOLZP$(i) <$ ICOLZP$(i+1)$.

On entry, ICOLZP$(1) = \langle value \rangle$.
Constraint: ICOLZP$(1) = 1$.

On entry, ICOLZP$(\mathrm{N}+1) = \langle value \rangle$ and NNZ $= \langle value \rangle$.
Constraint: ICOLZP$(\mathrm{N}+1) =$ NNZ $+ 1$.

IFAIL $= 5$

On entry, in specification of column $\langle value \rangle$, and for $j = \langle value \rangle$, IROWIX$(j) = \langle value \rangle$ and IROWIX$(j+1) = \langle value \rangle$.
Constraint: IROWIX$(j) <$ IROWIX$(j+1)$.

IFAIL $= 6$

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

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 $+ 1$.

IFAIL $= 8$

On entry, the matrix $A - \sigma \times I$ is nearly numerically singular and could not be inverted. Try perturbing the value of σ. Norm of matrix $= \langle value \rangle$, Reciprocal condition number $= \langle value \rangle$.

On entry, the matrix $A - \sigma \times I$ 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 $= 21$

The maximum number of iterations ≤ 0, the optional parameter **Iteration Limit** has been set to $\langle value \rangle$.

IFAIL $= 22$

An internal call to F12ABF returned with IFAIL $= 2$.
This error should not occur. Please contact NAG.

IFAIL $= 23$

An internal call to F12ABF returned with IFAIL $= 3$.

IFAIL $= 24$

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 $= 25$

No shifts could be applied during a cycle of the implicitly restarted Arnoldi iteration.

IFAIL $= 26$

Could not build an Arnoldi factorization. The size of the current Arnoldi factorization $= \langle value \rangle$.

IFAIL $= 27$

Error in internal call to compute eigenvalues and corresponding error bounds of the current upper Hessenberg matrix.
Please contact NAG.

IFAIL $= 32$

An internal call to F12ACF returned with IFAIL $= 2$.

IFAIL $= 33$

The number of eigenvalues found to sufficient accuracy is zero.

IFAIL $= 34$

Internal inconsistency in the number of converged Ritz values. Number counted $= \langle value \rangle$, number expected $= \langle value \rangle$.

IFAIL $= 35$

During calculation of a real Schur form, there was a failure to compute $\langle value \rangle$ eigenvalues in a total of $\langle value \rangle$ iterations.

IFAIL $= 36$

The computed Schur form could not be reordered by an internal call.
This routine returned with IFAIL $= \langle value \rangle$.
Please contact NAG.

IFAIL $= 37$

In calculating eigenvectors, an internal call returned with an error.
Please contact NAG.

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

F02EKF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F02EKF 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

F02EKF calls routines based on the ARPACK suite in Chapter F12. These routines use an implicitly restarted Arnoldi 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 Arnoldi process. Each iteration is therefore cheap computationally, relative to the alternative, **Shifted Inverse Real**, 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 Real** mode should be used.

If the **Shifted Inverse Real** mode is used (via a call to F12ADF in OPTION) then the matrix $A - \sigma I$ is used in linear system solves by the Arnoldi process. This is first factorized internally using the direct LU factorization routine F11MEF. The condition number of $A - \sigma I$ is then calculated by a call to F11MGF. If the condition number is too big then the matrix is considered to be nearly singular, i.e., σ is an approximate eigenvalue of A, and the routine exits with an error. In this situation it is normally sufficient to perturb σ by a small amount and call F02EKF again. After successful factorization, subsequent solves are performed by calls to F11MFF.

Finally, F02EKF transforms the eigenvectors. Each eigenvector w (real or complex) is normalized so that $\|w\|_2 = 1$, and the element of largest absolute value is real.

The monitoring routine MONIT provides some basic information on the convergence of the Arnoldi iterations. Much greater levels of detail on the Arnoldi 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 **Monitoring** option may be used to select a monitoring ***file*** by setting the option to a file identification (unit) number associated with **Monitoring** (see X04ACF).

10 Example

This example computes the four eigenvalues of the matrix A which lie closest to the value $\sigma = 5.5$ on the real line, and their corresponding eigenvectors, where A is the tridiagonal matrix with elements

$$a_{ij} = \begin{cases} 2+i, & j = i \\ 3, & j = i-1 \\ -1+\rho/(2n+2), & j = i+1 \text{ with } \rho = 10.0. \end{cases}$$

10.1 Program Text

```
!   F02EKF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module f02ekfe_mod

!     F02EKF 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: f12adf
!       .. 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 f12adf(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 f12adf(rec,icomm,comm,ifail1)
          istat = max(istat,ifail1)
        End If
        If (iuser(3)>0) Then
          ifail1 = 1
          Call f12adf('Shifted Inverse Real',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 ..
        Complex (Kind=nag_wp), Intent (In) :: w(ncv)
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: rzest(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,',',E13.5,')',1X,E13.5)
99996   Format (1X,1X,I4,1X,'(',E13.5,',',E13.5,')')
      End Subroutine mymonit
    End Module f02ekfe_mod
    Program f02ekfe

!     Example problem for F02EKF.

!     .. Use Statements ..
      Use nag_library, Only: f02ekf, nag_wp, x02ajf
      Use f02ekfe_mod, Only: mymonit, myoption, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: three = 3.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: two = 2.0_nag_wp
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: h, rho, s, sigma
      Integer                          :: i, ifail, imon, k, ldv, maxit, mode, &
                                          n, nconv, ncv, nev, nnz, nx, prtlvl
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: w(:)
      Real (Kind=nag_wp), Allocatable  :: a(:), resid(:), v(:,:)
      Real (Kind=nag_wp)               :: ruser(1)
      Integer, Allocatable             :: icolzp(:), irowix(:)
      Integer                          :: iuser(4)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: real
!     .. Executable Statements ..
      Write (nout,*) 'F02EKF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
```

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

      n = nx*nx
      nnz = 3*n - 2
      ldv = n

      Allocate (resid(ncv),a(nnz),icolzp(n+1),irowix(nnz),w(ncv),v(ldv,ncv))

!     Construct  A in compressed column storage (CCS) format where:
!      A_{i,i}   = 2 + i
!      A_{i+1,i) = 3
!      A_{i,i+1} = rho/(2n+2) - 1

      h = one/real(n+1,kind=nag_wp)
      s = rho*h/two - one

      a(1) = two + one
      a(2) = three
      icolzp(1) = 1
      irowix(1) = 1
      irowix(2) = 2
      k = 3
      Do i = 2, n - 1
        icolzp(i) = k
        irowix(k) = i - 1
        irowix(k+1) = i
        irowix(k+2) = i + 1
        a(k) = s
        a(k+1) = two + real(i,kind=nag_wp)
        a(k+2) = three
        k = k + 3
      End Do
      icolzp(n) = k
      icolzp(n+1) = k + 2
      irowix(k) = n - 1
      irowix(k+1) = n
      a(k) = s
      a(k+1) = two + real(n,kind=nag_wp)

!     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

      If (prtlvl>0) Then
        imon = 0
      End If

      iuser(1) = prtlvl
      iuser(2) = maxit
      iuser(3) = mode
      iuser(4) = imon

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f02ekf(n,nnz,a,icolzp,irowix,nev,ncv,sigma,mymonit,myoption,nconv,  &
        w,v,ldv,resid,iuser,ruser,ifail)

      Write (nout,99999) nconv, sigma
      Do i = 1, nconv
        If (resid(i)>real(100*n,kind=nag_wp)*x02ajf()) Then
```

```
          Write (nout,99998) i, w(i), resid(i)
        Else
          Write (nout,99998) i, w(i)
        End If
      End Do

99999 Format (1X,/,' The ',I4,' Ritz values of closest to ',E13.5,' are:',/)
99998 Format (1X,I8,5X,'( ',E13.5,' , ',E13.5,' )',5X,E13.5)
    End Program f02ekfe
```

10.2 Program Data

```
F02EKF Example Program Data
 10      : nx, matrix order n = nx*nx
 4       : nev, number of eigenvalues requested
 20      : ncv, size of subspace
 10.0    : rho, parameter for determining A
 5.5     : sigma, shift (want eigenvalues close to sigma)
 0       : print level
 500     : maximum number of itrerations
 1       : mode (0 = regular, 1 = shifted inverse)
 1       : imon (0 = no monitoring, 1 = monitoring on)
```

10.3 Program Results

```
 F02EKF Example Program Results

  Arnoldi basis vectors used:  20
  The following Ritz values (mu) are related to the
  true eigenvalues (lambda) by lambda = sigma + 1/mu

  Iteration number    1
  Ritz values converged so far (   2) and their Ritz estimates:
     1 (  0.56992E+00,  0.88081E+00)   0.13008E-19
     2 (  0.56992E+00, -0.88081E+00)   0.13008E-19
  Next (unconverged) Ritz value:
     3 (  0.60777E+00,  0.00000E+00)

 The    5 Ritz values of closest to   0.55000E+01 are:

        1     (   0.60178E+01 ,  -0.80028E+00 )
        2     (   0.60178E+01 ,   0.80028E+00 )
        3     (   0.43431E+01 ,  -0.19456E+01 )
        4     (   0.43431E+01 ,   0.19456E+01 )
        5     (   0.71453E+01 ,   0.00000E+00 )
```

11 Optional Parameters

Internally F02EKF calls routines from the suite F12AAF, F12ABF, F12ACF, F12ADF and F12AEF. 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 F02EKF 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 F02EKF. OPTION must be coded such that one call to F12ADF 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

Defaults

Iteration Limit
Largest Imaginary
Largest Magnitude
Largest Real
List
Monitoring
Nolist
Print Level
Regular
Shifted Inverse Real
Smallest Imaginary
Smallest Magnitude
Smallest Real
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 $= 0$

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 Arnoldi iterations that can be performed before F02EKF exits with IFAIL $\neq 0$.

Largest Magnitude Default
Largest Imaginary
Largest Real
Smallest Imaginary
Smallest Magnitude
Smallest Real

The Arnoldi iterative method converges on a number of eigenvalues with given properties. The default is to compute the eigenvalues of largest magnitude using **Largest Magnitude**. Alternatively, eigenvalues may be chosen which have **Largest Real** part, **Largest Imaginary** part, **Smallest Magnitude**, **Smallest Real** part or **Smallest Imaginary** part.

Note that these options select the eigenvalue properties for eigenvalues of OP the linear operator determined by the computational mode and problem type.

Nolist Default
List

Normally each optional parameter specification is not printed to the advisory channel as it is supplied. Optional parameter **List** may be used to enable printing and optional parameter **Nolist** may be used to suppress the printing.

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 F02EKF as follows.

$= 0$ No output except error messages.

> 0 The set of selected options.

$= 2$ Problem and timing statistics when all calls to F12ABF have been completed.

≥ 5 A single line of summary output at each Arnoldi iteration.

≥ 10 If **Monitoring** > 0, then at each iteration, the length and additional steps of the current Arnoldi factorization and the number of converged Ritz values; during re-orthogonalization, the norm of initial/restarted starting vector.

≥ 20 Problem and timing statistics on final exit from F12ABF. If **Monitoring** > 0, then at each iteration, the number of shifts being applied, the eigenvalues and estimates of the Hessenberg matrix H, the size of the Arnoldi 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, 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 compressed upper Hessenberg matrix H; during re-orthogonalization, the initial/restarted starting vector; during the Arnoldi iteration loop, a restart is flagged and the number of the residual requiring iterative refinement; while applying shifts, the indices of the shifts being applied.

≥ 40 If **Monitoring** > 0, then during the Arnoldi iteration loop, the Arnoldi 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** > 0, then during Arnoldi iteration loop: norms of key components and the active column of H, norms of residuals during iterative refinement, the final upper Hessenberg matrix H; while applying shifts: number of shifts, shift values, block indices, updated matrix H; while computing eigenvalues of H: the matrix H, the computed eigenvalues and Ritz estimates.

Regular Default
Shifted Inverse Real

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

Given a standard eigenvalue problem in the form $Ax = \lambda x$ then the following modes are available with the appropriate operator OP(x).

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

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.
