NAG Library Routine Document

F12AEF

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, then the option setting routine F12ADF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 11 in F12ADF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12AEF can be used to return additional monitoring information during computation. It is in a suite of routines consisting of F12AAF, F12ABF, F12ACF, F12ADF and F12AEF.

2 Specification

```
SUBROUTINE F12AEF (NITER, NCONV, RITZR, RITZI, RZEST, ICOMM, COMM)
INTEGER            NITER, NCONV, ICOMM(*)
REAL (KIND=nag_wp) RITZR(*), RITZI(*), RZEST(*), COMM(*)
```

3 Description

The suite of routines is designed to calculate some of the eigenvalues, λ, (and optionally the corresponding eigenvectors, x) of a standard eigenvalue problem $Ax = \lambda x$, or of a generalized eigenvalue problem $Ax = \lambda Bx$ of order n, where n is large and the coefficient matrices A and B are sparse, real and nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and nonsymmetric problems.

On an intermediate exit from F12ABF with IREVCM $= 4$, F12AEF may be called to return monitoring information on the progress of the Arnoldi iterative process. The information returned by F12AEF is:

– the number of the current Arnoldi iteration;

– the number of converged eigenvalues at this point;

– the real and imaginary parts of the converged eigenvalues;

– the error bounds on the converged eigenvalues.

F12AEF does not have an equivalent routine from the ARPACK package which prints various levels of detail of monitoring information through an output channel controlled via an argument value (see Lehoucq *et al.* (1998) for details of ARPACK routines). F12AEF should not be called at any time other than immediately following an IREVCM $= 4$ return from F12ABF.

4 References

Lehoucq R B (2001) Implicitly restarted Arnoldi methods and subspace iteration *SIAM Journal on Matrix Analysis and Applications* **23** 551–562

Lehoucq R B and Scott J A (1996) An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices *Preprint MCS-P547-1195* Argonne National Laboratory

Lehoucq R B and Sorensen D C (1996) Deflation techniques for an implicitly restarted Arnoldi iteration *SIAM Journal on Matrix Analysis and Applications* **17** 789–821

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: NITER – INTEGER *Output*

On exit: the number of the current Arnoldi iteration.

2: NCONV – INTEGER *Output*

On exit: the number of converged eigenvalues so far.

3: RITZR(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RITZR must be at least NCV (see F12AAF).

On exit: the first NCONV locations of the array RITZR contain the real parts of the converged approximate eigenvalues.

4: RITZI(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RITZI must be at least NCV (see F12AAF).

On exit: the first NCONV locations of the array RITZI contain the imaginary parts of the converged approximate eigenvalues.

5: RZEST(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RZEST must be at least NCV (see F12AAF).

On exit: the first NCONV locations of the array RZEST contain the Ritz estimates (error bounds) on the converged approximate eigenvalues.

6: ICOMM(∗) – INTEGER array *Communication Array*

Note: the dimension of the array ICOMM must be at least $\max(1, \text{LICOMM})$, where LICOMM is passed to the setup routine (see F12AAF).

On entry: the array ICOMM output by the preceding call to F12ABF.

7: COMM(∗) – REAL (KIND=nag_wp) array *Communication Array*

Note: the dimension of the array COMM must be at least $\max(1, \text{LCOMM})$, where LCOMM is passed to the setup routine (see F12AAF).

On entry: the array COMM output by the preceding call to F12ABF.

6 Error Indicators and Warnings

None.

7 Accuracy

A Ritz value, λ, is deemed to have converged if its Ritz estimate $\leq$ **Tolerance** $\times |\lambda|$. The default **Tolerance** used is the ***machine precision*** given by X02AJF.

8 Parallelism and Performance

F12AEF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example solves $Ax = \lambda Bx$ in shifted-real mode, where A is the tridiagonal matrix with 2 on the diagonal, -2 on the subdiagonal and 3 on the superdiagonal. The matrix B is the tridiagonal matrix with 4 on the diagonal and 1 on the off-diagonals. The shift sigma, σ, is a complex number, and the operator used in the shifted-real iterative process is $\mathrm{OP} = \mathrm{real}\big((A - \sigma B)_{-1} B\big)$.

10.1 Program Text

```
!   F12AEF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module f12aefe_mod

!     F12AEF 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                              :: av, mv
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: four = 4.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: three = 3.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: two = 2.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: zero = 0.0_nag_wp
      Integer, Parameter, Public          :: nin = 5, nout = 6
    Contains
      Subroutine mv(n,v)
!       Compute the in-place matrix vector multiplication X<---M*X,
!       where M is mass matrix formed by using piecewise linear elements
!       on [0,1].

!       .. Scalar Arguments ..
        Integer, Intent (In)              :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: v(n)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                :: vm1, vv
        Integer                           :: j
!       .. Executable Statements ..
        vm1 = v(1)
        v(1) = four*v(1) + v(2)
        Do j = 2, n - 1
          vv = v(j)
          v(j) = vm1 + four*vv + v(j+1)
          vm1 = vv
        End Do
        v(n) = vm1 + four*v(n)
        Return
      End Subroutine mv

      Subroutine av(n,v,w)

!       .. Scalar Arguments ..
        Integer, Intent (In)              :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: v(n)
        Real (Kind=nag_wp), Intent (Out) :: w(n)
!       .. Local Scalars ..
        Integer                           :: j
!       .. Executable Statements ..
        w(1) = two*v(1) + three*v(2)
        Do j = 2, n - 1
          w(j) = -two*v(j-1) + two*v(j) + three*v(j+1)
        End Do
```

```
        w(n) = -two*v(n-1) + two*v(n)
        Return
      End Subroutine av
    End Module f12aefe_mod

    Program f12aefe

!     F12AEF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: ddot, dnrm2, f06bnf, f12aaf, f12abf, f12acf,      &
                             f12adf, f12aef, nag_wp, zgttrf, zgttrs
      Use f12aefe_mod, Only: av, four, mv, nin, nout, three, two, zero
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: c1, c2, c3, csig
      Real (Kind=nag_wp)               :: deni, denr, nev_nrm, numi, numr,     &
                                          sigmai, sigmar
      Integer                          :: ifail, ifail1, info, irevcm, j,      &
                                          lcomm, ldv, licomm, n, nconv, ncv,   &
                                          nev, niter, nshift
      Logical                          :: first
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: cdd(:), cdl(:), cdu(:), cdu2(:),   &
                                          ctemp(:)
      Real (Kind=nag_wp), Allocatable  :: ax(:), comm(:), d(:,:), mx(:),       &
                                          resid(:), v(:,:), x(:)
      Integer, Allocatable             :: icomm(:), ipiv(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: cmplx, real
!     .. Executable Statements ..
      Write (nout,*) 'F12AEF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nev, ncv, sigmar, sigmai
      ldv = n
      licomm = 140
      lcomm = 3*n + 3*ncv*ncv + 6*ncv + 60

      Allocate (cdd(n),cdl(n),cdu(n),cdu2(n),ctemp(n),ax(n),comm(lcomm),       &
        d(ncv,3),mx(n),resid(n),v(ldv,ncv),x(n),icomm(licomm),ipiv(n))

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f12aaf(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     Set the mode.
      ifail = 0
      Call f12adf('SHIFTED REAL',icomm,comm,ifail)

!     Set problem type
      Call f12adf('GENERALIZED',icomm,comm,ifail)

!     Solve A*x = lambda*B*x in shift-invert mode.
!     The shift, sigma, is a complex number (sigmar, sigmai).
!     OP = Real_Part{inv[A-(SIGMAR,SIGMAI)*M]*M and  B = M.
      csig = cmplx(sigmar,sigmai,kind=nag_wp)
      c1 = cmplx(-two,kind=nag_wp) - csig
      c2 = cmplx(two,kind=nag_wp) - cmplx(four,kind=nag_wp)*csig
      c3 = cmplx(three,kind=nag_wp) - csig

      cdl(1:n-1) = c1
      cdd(1:n-1) = c2
      cdu(1:n-1) = c3
      cdd(n) = c2

!     The NAG name equivalent of zgttrf is f07crf
      Call zgttrf(n,cdl,cdd,cdu,cdu2,ipiv,info)
```

```
      irevcm = 0
      ifail = -1
loop: Do
        Call f12abf(irevcm,resid,v,ldv,x,mx,nshift,comm,icomm,ifail)

        If (irevcm/=5) Then
          Select Case (irevcm)
          Case (-1)
!           Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x
            Call mv(n,x)
            ctemp(1:n) = cmplx(x(1:n),kind=nag_wp)
!           The NAG name equivalent of zgttrs is f07csf
            Call zgttrs('N',n,1,cdl,cdd,cdu,cdu2,ipiv,ctemp,n,info)
            x(1:n) = real(ctemp(1:n))
          Case (1)
!           Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x,
!           M*X stored in MX.
            ctemp(1:n) = cmplx(mx(1:n),kind=nag_wp)
!           The NAG name equivalent of zgttrs is f07csf
            Call zgttrs('N',n,1,cdl,cdd,cdu,cdu2,ipiv,ctemp,n,info)
            x(1:n) = real(ctemp(1:n))
          Case (2)
!           Perform  y <--- M*x
            Call mv(n,x)
          Case (4)
!           Output monitoring information
            Call f12aef(niter,nconv,d,d(1,2),d(1,3),icomm,comm)
!           The NAG name equivalent of dnrm2 is f06ejf
            nev_nrm = dnrm2(nev,d(1,3),1)
            Write (6,99999) niter, nconv, nev_nrm
          End Select
        Else
          Exit loop
        End If
      End Do loop
      If (ifail==0) Then

!       Post-Process using F12ACF to compute eigenvalues/vectors.
        ifail1 = 0
        Call f12acf(nconv,d,d(1,2),v,ldv,sigmar,sigmai,resid,v,ldv,comm,icomm, &
          ifail1)

        first = .True.
        Do j = 1, nconv
!         Use Rayleigh Quotient to recover eigenvalues of the original
!         problem.
!         The NAG name equivalent of ddot is f06eaf
          If (d(j,2)==zero) Then
!           Ritz value is real. x = v(:,j); eig = x'Ax/x'Mx.
            Call av(n,v(1,j),ax)
            numr = ddot(n,v(1,j),1,ax,1)
            mx(1:n) = v(1:n,j)
            Call mv(n,mx)
            denr = ddot(n,v(1,j),1,mx,1)
            d(j,1) = numr/denr
          Else If (first) Then
!           Ritz value is complex: x = v(:,j) - i v(:,j+1).

!           Compute x'(Ax):
!              first (xr,xi)'*(A xr)
            Call av(n,v(1,j),ax)
            numr = ddot(n,v(1,j),1,ax,1)
            numi = ddot(n,v(1,j+1),1,ax,1)
!           then add (xi,-xr)'*(A xi)
            Call av(n,v(1,j+1),ax)
            numr = numr + ddot(n,v(1,j+1),1,ax,1)
            numi = -numi + ddot(n,v(1,j),1,ax,1)

!           Compute x'(Mx) as above using mv in, place of av.
            mx(1:n) = v(1:n,j)
```

```
            Call mv(n,mx)
            denr = ddot(n,v(1,j),1,mx,1)
            deni = ddot(n,v(1,j+1),1,mx,1)
            mx(1:n) = v(1:n,j+1)
            Call mv(n,mx)
            denr = denr + ddot(n,v(1,j+1),1,mx,1)
            deni = -deni + ddot(n,v(1,j),1,mx,1)

!           Rayleigh quotient,  d=x'(Ax)/x'(Mx), (complex division).
            d(j,1) = (numr*denr+numi*deni)/f06bnf(denr,deni)
            d(j,2) = (numi*denr-numr*deni)/f06bnf(denr,deni)
            first = .False.
          Else
!           Second of complex conjugate pair.
            d(j,1) = d(j-1,1)
            d(j,2) = -d(j-1,2)
            first = .True.
          End If
        End Do
!       Print computed eigenvalues.
        Write (nout,99998) nconv, sigmar, sigmai
        Write (nout,99997)(j,d(j,1:2),j=1,nconv)
      End If

99999 Format (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o',       &
        'f estimates =',E12.4)
99998 Format (1X,/,' The ',I4,' generalized Ritz values closest to (',F8.4,    &
        ', ',F8.4,') are:',/)
99997 Format (1X,I8,5X,'(',F7.4,',',F7.4,')')
    End Program f12aefe
```

10.2 Program Data

```
F12AEF Example Program Data
 100 4 20 4.0D-1 6.0D-1 : Values for NX NEV NCV SIGMAR SIGMAI
```

10.3 Program Results

```
 F12AEF Example Program Results

 Iteration   1, No. converged =   0, norm of estimates =  0.1052E+00
 Iteration   2, No. converged =   0, norm of estimates =  0.1188E-02
 Iteration   3, No. converged =   0, norm of estimates =  0.1389E-05
 Iteration   4, No. converged =   0, norm of estimates =  0.3939E-08
 Iteration   5, No. converged =   0, norm of estimates =  0.1158E-10
 Iteration   6, No. converged =   0, norm of estimates =  0.5222E-13

 The    4 generalized Ritz values closest to (  0.4000,   0.6000) are:

        1     ( 0.5000,-0.5958)
        2     ( 0.5000, 0.5958)
        3     ( 0.5000,-0.6331)
        4     ( 0.5000, 0.6331)
```