NAG Library Routine Document

F12FEF

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 F12FDF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 10 in F12FDF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12FEF can be used to return additional monitoring information during computation. It is in a suite of routines which includes F12FAF, F12FBF, F12FCF and F12FDF.

2 Specification

```
SUBROUTINE F12FEF (NITER, NCONV, RITZ, RZEST, ICOMM, COMM)

INTEGER             NITER, NCONV, ICOMM(*)
REAL (KIND=nag_wp)  RITZ(*), 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 symmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and symmetric problems.

On an intermediate exit from F12FBF with IREVCM $= 4$, F12FEF may be called to return monitoring information on the progress of the Arnoldi iterative process. The information returned by F12FEF 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.

F12FEF 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 a parameter value (see Lehoucq *et al.* (1998) for details of ARPACK routines). F12FEF should not be called at any time other than immediately following an IREVCM $= 4$ return from F12FBF.

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 Parameters

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: RITZ($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RITZ must be at least NCV (see F12FAF).

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

4: RZEST($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RZEST must be at least NCV (see F12FAF).

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

5: 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 F12FAF).

On entry: the array ICOMM output by the preceding call to F12FBF.

6: 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 F12FAF).

On entry: the array COMM output by the preceding call to F12FBF.

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 Further Comments

None.

9 Example

This example solves $Kx = \lambda K_G x$ using the **Buckling** option (see F12FDF, where K and K_G are obtained by the finite element method applied to the one-dimensional discrete Laplacian operator $\frac{\partial^2 u}{\partial x^2}$ on $[0, 1]$, with zero Dirichlet boundary conditions using piecewise linear elements. The shift, σ, is a real number, and the operator used in the Buckling iterative process is $\text{OP} = \text{inv}(K - \sigma K_G) \times K$ and $B = K$.

9.1 Program Text

```
!   F12FEF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module f12fefe_mod

!     F12FEF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: four = 4.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: six = 6.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: two = 2.0_nag_wp
      Integer, Parameter                   :: imon = 0, ipoint = 0,          &
                                              licomm = 140, nin = 5, nout = 6
    Contains
      Subroutine av(n,v,w)

!       .. Use Statements ..
        Use nag_library, Only: dscal
!       .. 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 ..
        Real (Kind=nag_wp)                   :: h
        Integer                              :: j
!       .. Intrinsic Procedures ..
        Intrinsic                            :: real
!       .. Executable Statements ..
        h = one/real(n+1,kind=nag_wp)
        w(1) = two*v(1) - v(2)
        Do j = 2, n - 1
          w(j) = -v(j-1) + two*v(j) - v(j+1)
        End Do
        j = n
        w(j) = -v(j-1) + two*v(j)
!       The NAG name equivalent of dscal is f06edf
        Call dscal(n,one/h,w,1)
        Return
      End Subroutine av
    End Module f12fefe_mod
    Program f12fefe

!     F12FEF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: dcopy, dgttrf, dgttrs, dnrm2, f12faf, f12fbf,     &
                             f12fcf, f12fdf, f12fef, nag_wp
      Use f12fefe_mod, Only: av, four, imon, ipoint, licomm, nin, nout, one,   &
                             six, two
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: h, r1, r2, sigma
      Integer                              :: ifail, info, irevcm, j, lcomm,   &
                                              ldv, n, nconv, ncv, nev, niter,  &
                                              nshift
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: ad(:), adl(:), adu(:), adu2(:),  &
                                              comm(:), d(:,:), mx(:),          &
                                              resid(:), v(:,:), x(:)
      Integer                              :: icomm(licomm)
      Integer, Allocatable                 :: ipiv(:)
```

```
!     .. Intrinsic Procedures ..
      Intrinsic                            :: real
!     .. Executable Statements ..
      Write (nout,*) 'F12FEF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nev, ncv

      lcomm = 3*n + ncv*ncv + 8*ncv + 60
      ldv = n
      Allocate (ad(n),adl(n),adu(n),adu2(n),comm(lcomm),d(ncv,2),mx(n), &
        resid(n),v(ldv,ncv),x(n),ipiv(n))

      ifail = 0
      Call f12faf(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     We are solving a generalized problem
      ifail = 0
      Call f12fdf('GENERALIZED',icomm,comm,ifail)
!     Indicate that we are using the buckling mode.
      Call f12fdf('BUCKLING',icomm,comm,ifail)
      If (ipoint==1) Then
        Call f12fdf('POINTERS=YES',icomm,comm,ifail)
      End If

      h = one/real(n+1,kind=nag_wp)
      r1 = (four/six)*h
      r2 = (one/six)*h
      sigma = one
      ad(1:n) = two/h - sigma*r1
      adl(1:n) = -one/h - sigma*r2
      adu(1:n) = adl(1:n)

!     The NAG name equivalent of dgttrf is f07cdf
      Call dgttrf(n,adl,ad,adu,adu2,ipiv,info)

      irevcm = 0
      ifail = -1
revcm: Do
        Call f12fbf(irevcm,resid,v,ldv,x,mx,nshift,comm,icomm,ifail)
        If (irevcm==5) Then
          Exit revcm
        Else If (irevcm==-1) Then
!         Perform  y <--- OP*x = inv[K-SIGMA*KG]*K*x
!         The NAG name equivalent of dgttrs is f07cef
          If (ipoint==0) Then
            Call av(n,x,mx)
            x(1:n) = mx(1:n)
            Call dgttrs('N',n,1,adl,ad,adu,adu2,ipiv,x,n,info)
          Else
            Call av(n,comm(icomm(1)),comm(icomm(2)))
            Call dgttrs('N',n,1,adl,ad,adu,adu2,ipiv,comm(icomm(2)),n,info)
          End If
        Else If (irevcm==1) Then
!         Perform y <-- OP*x = inv[K-sigma*KG]*K*x.
!         The NAG name equivalent of dgttrs is f07cef
          If (ipoint==0) Then
            x(1:n) = mx(1:n)
            Call dgttrs('N',n,1,adl,ad,adu,adu2,ipiv,x,n,info)
          Else
!           The NAG name equivalent of dcopy is f06eff
            Call dcopy(n,comm(icomm(3)),1,comm(icomm(2)),1)
            Call dgttrs('N',n,1,adl,ad,adu,adu2,ipiv,comm(icomm(2)),n,info)
          End If
        Else If (irevcm==2) Then
!         Perform  y <--- M*x.
          If (ipoint==0) Then
            Call av(n,x,mx)
          Else
            Call av(n,comm(icomm(1)),comm(icomm(2)))
          End If
        Else If (irevcm==4 .And. imon/=0) Then
!         Output monitoring information
          Call f12fef(niter,nconv,d,d(1,2),icomm,comm)
!         The NAG name equivalent of dnrm2 is f06ejf
          Write (6,99999) niter, nconv, dnrm2(nev,d(1,2),1)
        End If
      End Do revcm

      If (ifail==0) Then
!       Post-Process using F12FCF to compute eigenvalues/vectors.
        Call f12fcf(nconv,d,v,ldv,sigma,resid,v,ldv,comm,icomm,ifail)
        Write (nout,99998) nconv, sigma
        Write (nout,99997)(j,d(j,1),j=1,nconv)
      End If

99999 Format (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o', &
        'f estimates =',E16.8)
99998 Format (1X/' The ',I4,' generalized Ritz values closest to ',F8.4, &
        ' are:'/)
99997 Format (1X,I8,5X,F12.4)
    End Program f12fefe
```

9.2 Program Data

```
F12FEF Example Program Data
 100 4 10 : Values for N NEV and NCV
```

9.3 Program Results

```
 F12FEF Example Program Results


 The    4 generalized Ritz values closest to   1.0000 are:

        1          9.8704
        2         39.4912
        3         88.8909
        4        158.1175
```