NAG Library Routine Document

F12ASF

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 F12ARF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 11 in F12ARF for a detailed description of the specification of the optional parameters*.

1 Purpose

F12ASF can be used to return additional monitoring information during computation. It is in a suite of routines consisting of F12ANF, F12APF, F12AQF, F12ARF and F12ASF.

2 Specification

```
SUBROUTINE F12ASF (NITER, NCONV, RITZ, RZEST, ICOMM, COMM)
INTEGER                NITER, NCONV, ICOMM(*)
COMPLEX (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 complex eigenvalue problem $Ax = \lambda x$, or of a generalized complex eigenvalue problem $Ax = \lambda Bx$ of order n, where n is large and the coefficient matrices A and B are sparse and complex. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense complex problems.

On an intermediate exit from F12APF with IREVCM $= 4$, F12ASF may be called to return monitoring information on the progress of the Arnoldi iterative process. The information returned by F12ASF is:

– the number of the current Arnoldi iteration;

– the number of converged eigenvalues at this point;

– the converged eigenvalues;

– the error bounds on the converged eigenvalues.

F12ASF 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). F12ASF should not be called at any time other than immediately following an IREVCM $= 4$ return from F12APF.

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: RITZ(∗) – COMPLEX (KIND=nag_wp) array *Output*

Note: the dimension of the array RITZ must be at least NCV (see F12ANF).

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

4: RZEST(∗) – COMPLEX (KIND=nag_wp) array *Output*

Note: the dimension of the array RZEST must be at least NCV (see F12ANF).

On exit: the first NCONV locations of the array RZEST contain the complex Ritz estimates on the 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 F12ANF).

On entry: the array ICOMM output by the preceding call to F12APF.

6: COMM(∗) – COMPLEX (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 F12ANF).

On entry: the array COMM output by the preceding call to F12APF.

6 Error Indicators and Warnings

None.

7 Accuracy

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

8 Parallelism and Performance

F12ASF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example solves $Ax = \lambda Bx$ in shifted-inverse mode, where A and B are obtained from the standard central difference discretization of the one-dimensional convection-diffusion operator $\frac{d^2u}{dx^2} + \rho\frac{du}{dx}$ on $[0, 1]$, with zero Dirichlet boundary conditions. The shift, σ, is a complex number, and the operator used in the shifted-inverse iterative process is $\text{OP} = \text{inv}(A - \sigma B) \times B$.

10.1 Program Text

```
!   F12ASF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module f12asfe_mod

!     F12ASF 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                             :: mv
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter, Public :: four = (4.0_nag_wp,          &
                                          0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter, Public :: one = (1.0_nag_wp,0.0_nag_wp &
                                          )
      Complex (Kind=nag_wp), Parameter, Public :: six = (6.0_nag_wp,0.0_nag_wp &
                                          )
      Complex (Kind=nag_wp), Parameter, Public :: two = (2.0_nag_wp,0.0_nag_wp &
                                          )
      Integer, Parameter, Public       :: imon = 1, licomm = 140, nerr = 6,    &
                                          nin = 5, nout = 6
    Contains
      Subroutine mv(nx,v,w)
!       Compute the out-of--place matrix vector multiplication Y<---M*X,
!       where M is mass matrix formed by using piecewise linear elements
!       on [0,1].

!       .. Use Statements ..
        Use nag_library, Only: zscal
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: nx
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In) :: v(nx*nx)
        Complex (Kind=nag_wp), Intent (Out) :: w(nx*nx)
!       .. Local Scalars ..
        Complex (Kind=nag_wp)          :: h
        Integer                        :: j, n
!       .. Intrinsic Procedures ..
        Intrinsic                      :: cmplx
!       .. Executable Statements ..
        n = nx*nx
        w(1) = (four*v(1)+v(2))/six
        Do j = 2, n - 1
          w(j) = (v(j-1)+four*v(j)+v(j+1))/six
        End Do
        w(n) = (v(n-1)+four*v(n))/six

        h = one/cmplx(n+1,kind=nag_wp)
!       The NAG name equivalent of zscal is f06gdf
        Call zscal(n,h,w,1)
        Return
      End Subroutine mv
    End Module f12asfe_mod

    Program f12asfe

!     F12ASF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: dznrm2, f12anf, f12apf, f12aqf, f12arf, f12asf,   &
                             nag_wp, zgttrf, zgttrs
      Use f12asfe_mod, Only: four, imon, licomm, mv, nerr, nin, nout, one,     &
                             six, two
!     .. Implicit None Statement ..
```

```
      Implicit None
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: h, rho, s, s1, s2, s3, sigma
      Real (Kind=nag_wp)               :: nev_nrm
      Integer                          :: ifail, ifail1, info, irevcm, j,      &
                                          lcomm, ldv, n, nconv, ncv, nev,      &
                                          niter, nshift, nx
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: ax(:), comm(:), d(:,:), dd(:),     &
                                          dl(:), du(:), du2(:), mx(:),         &
                                          resid(:), v(:,:), x(:)
      Integer                          :: icomm(licomm)
      Integer, Allocatable             :: ipiv(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: cmplx
!     .. Executable Statements ..
      Write (nout,*) 'F12ASF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nx, nev, ncv

      n = nx*nx
      lcomm = 3*n + 3*ncv*ncv + 5*ncv + 60
      ldv = n
      Allocate (ax(n),comm(lcomm),d(ncv,2),dd(n),dl(n),du(n),du2(n),mx(n),     &
        resid(n),v(ldv,ncv),x(n),ipiv(n))

      ifail = 0
      Call f12anf(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     Set the mode.
      ifail = 0
      Call f12arf('SHIFTED INVERSE',icomm,comm,ifail)
!     Set problem type.
      Call f12arf('GENERALIZED',icomm,comm,ifail)
      sigma = (5000.0_nag_wp,0.0_nag_wp)
      rho = (10.0_nag_wp,0.0_nag_wp)
      h = one/cmplx(n+1,kind=nag_wp)
      s = rho/two
      s1 = -one/h - s - sigma*h/six
      s2 = two/h - four*sigma*h/six
      s3 = -one/h + s - sigma*h/six

      dl(1:n-1) = s1
      dd(1:n-1) = s2
      du(1:n-1) = s3
      dd(n) = s2

!     The NAG name equivalent of zgttrf is f07crf
      Call zgttrf(n,dl,dd,du,du2,ipiv,info)
      If (info/=0) Then
        Write (nerr,99999) info
        Go To 100
      End If

      irevcm = 0
      ifail = -1
revcm: Do
        Call f12apf(irevcm,resid,v,ldv,x,mx,nshift,comm,icomm,ifail)
        If (irevcm==5) Then
          Exit revcm
        Else If (irevcm==-1) Then
!         Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x
          Call mv(nx,x,ax)
          x(1:n) = ax(1:n)
!         The NAG name equivalent of zgttrs is f07csf
          Call zgttrs('N',n,1,dl,dd,du,du2,ipiv,x,n,info)
          If (info/=0) Then
            Write (nerr,99998) info
            Exit revcm
```

```
          End If
        Else If (irevcm==1) Then
!         Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x,
!         MX stored in COMM from location IPNTR(3)
!         The NAG name equivalent of zgttrs is f07csf
          Call zgttrs('N',n,1,dl,dd,du,du2,ipiv,mx,n,info)
          x(1:n) = mx(1:n)
          If (info/=0) Then
            Write (nerr,99998) info
            Exit revcm
          End If
        Else If (irevcm==2) Then
!         Perform  y <--- M*x
          Call mv(nx,x,ax)
          x(1:n) = ax(1:n)
        Else If (irevcm==4 .And. imon/=0) Then
!         Output monitoring information
          Call f12asf(niter,nconv,d,d(1,2),icomm,comm)
!         The NAG name equivalent of dznrm2 is f06jjf
          nev_nrm = dznrm2(nev,d(1,2),1)
          Write (6,99997) niter, nconv, nev_nrm
        End If
      End Do revcm

      If (ifail==0 .And. info==0) Then
!       Post-Process using F12AQF to compute eigenvalues/vectors.
        ifail1 = 0
        Call f12aqf(nconv,d,v,ldv,sigma,resid,v,ldv,comm,icomm,ifail1)
        Write (nout,99996) nconv, sigma
        Write (nout,99995)(j,d(j,1),j=1,nconv)
      End If
100   Continue

99999 Format (1X,'** Error status returned by ZGTTRF, INFO =',I12)
99998 Format (1X,'** Error status returned by ZGTTRS, INFO =',I12)
99997 Format (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o',       &
        'f estimates =',E12.4)
99996 Format (1X,/,' The ',I4,' generalized Ritz values closest to (',F8.3,    &
        ',',F8.3,') are:',/)
99995 Format (1X,I8,5X,'( ',F10.4,' , ',F10.4,' )')
    End Program f12asfe
```

10.2 Program Data

```
F12ASF Example Program Data
 16 4 10 : Vaues for NX NEV and NCV
```

10.3 Program Results

```
 F12ASF Example Program Results

 Iteration   1, No. converged =   0, norm of estimates =  0.5947E-06
 Iteration   2, No. converged =   1, norm of estimates =  0.1478E-08
 Iteration   3, No. converged =   2, norm of estimates =  0.3293E-10
 Iteration   4, No. converged =   2, norm of estimates =  0.5941E-13
 Iteration   5, No. converged =   2, norm of estimates =  0.8408E-15
 Iteration   6, No. converged =   3, norm of estimates =  0.8134E-17

 The    4 generalized Ritz values closest to (5000.000,   0.000) are:

        1     (  4829.8497 ,     0.0000 )
        2     (  5279.5223 ,    -0.0000 )
        3     (  4400.6310 ,     0.0000 )
        4     (  5749.7160 ,    -0.0000 )
```