NAG Library Routine Document

F12ANF

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.

1 Purpose

F12ANF is a setup routine in a suite of routines consisting of F12ANF, F12APF, F12AQF, F12ARF and F12ASF. It is used to find some of the eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by complex nonsymmetric matrices.

The suite of routines is suitable for the solution of large sparse, standard or generalized, nonsymmetric complex eigenproblems where only a few eigenvalues from a selected range of the spectrum are required.

2 Specification

```
SUBROUTINE F12ANF (N, NEV, NCV, ICOMM, LICOMM, COMM, LCOMM, IFAIL)

INTEGER               N, NEV, NCV, ICOMM(max(1,LICOMM)), LICOMM, LCOMM,        &
                      IFAIL
COMPLEX (KIND=nag_wp) COMM(max(1,LCOMM))
```

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, complex and nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, complex and nonsymmetric problems.

F12ANF is a setup routine which must be called before F12APF, the reverse communication iterative solver, and before F12ARF, the options setting routine. F12AQF is a post-processing routine that must be called following a successful final exit from F12APF, while F12ASF can be used to return additional monitoring information during the computation.

This setup routine initializes the communication arrays, sets (to their default values) all options that can be set by you via the option setting routine F12ARF, and checks that the lengths of the communication arrays as passed by you are of sufficient length. For details of the options available and how to set them see Section 10.1 in F12ARF.

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: N – INTEGER *Input*

On entry: the order of the matrix A (and the order of the matrix B for the generalized problem) that defines the eigenvalue problem.

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

2: NEV – INTEGER *Input*

On entry: the number of eigenvalues to be computed.

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

3: NCV – INTEGER *Input*

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

4: ICOMM($\max(1, \mathrm{LICOMM})$) – INTEGER array *Communication Array*

On exit: contains data to be communicated to the other routines in the suite.

5: LICOMM – INTEGER *Input*

On entry: the dimension of the array ICOMM as declared in the (sub)program from which F12ANF is called.

If $\mathrm{LICOMM} = -1$, a workspace query is assumed and the routine only calculates the required dimensions of ICOMM and COMM, which it returns in ICOMM(1) and COMM(1) respectively.

Constraint: $\mathrm{LICOMM} \geq 140$ or $\mathrm{LICOMM} = -1$.

6: COMM($\max(1, \mathrm{LCOMM})$) – COMPLEX (KIND=nag_wp) array *Communication Array*

On exit: contains data to be communicated to the other routines in the suite.

7: LCOMM – INTEGER *Input*

On entry: the dimension of the array COMM as declared in the (sub)program from which F12ANF is called.

If $\mathrm{LCOMM} = -1$, a workspace query is assumed and the routine only calculates the dimensions of ICOMM and COMM required by F12APF, which it returns in ICOMM(1) and COMM(1) respectively.

Constraint: $\mathrm{LCOMM} \geq 3 \times \mathrm{N} + 3 \times \mathrm{NCV} \times \mathrm{NCV} + 5 \times \mathrm{NCV} + 60$ or $\mathrm{LCOMM} = -1$.

8: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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 ≤ 0.

IFAIL $= 2$

On entry, NEV ≤ 0.

IFAIL $= 3$

On entry, NCV $<$ NEV $+ 2$ or NCV $>$ N.

IFAIL $= 4$

On entry, LICOMM < 140 and LICOMM $\neq -1$.

IFAIL $= 5$

On entry, LCOMM $< 3 \times$ N $+ 3 \times$ NCV $\times$ NCV $+ 5 \times$ NCV $+ 60$ and LCOMM $\neq -1$.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example solves $Ax = \lambda x$ in regular mode, where A is obtained from the standard central difference discretization of the convection-diffusion operator $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \rho \frac{\partial u}{\partial x}$ on the unit square, with zero Dirichlet boundary conditions. The eigenvalues of largest magnitude are found.

9.1 Program Text

```
!   F12ANF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.


    Module f12anfe_mod

!     F12ANF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: imon = 0, ipoint = 0,           &
                                              licomm = 140, nin = 5, nout = 6
    Contains
```

```
      Subroutine tv(nx,x,y)
!       Compute the matrix vector multiplication y<---T*x where T is a nx
!       by nx tridiagonal matrix.

!       .. Parameters ..
        Complex (Kind=nag_wp), Parameter     :: four = (4.0_nag_wp,0.0_nag_wp)
        Complex (Kind=nag_wp), Parameter     :: half = (0.5_nag_wp,0.0_nag_wp)
        Complex (Kind=nag_wp), Parameter     :: rho = (100.0_nag_wp,0.0_nag_wp)
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: nx
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In)   :: x(nx)
        Complex (Kind=nag_wp), Intent (Out)  :: y(nx)
!       .. Local Scalars ..
        Complex (Kind=nag_wp)                :: dd, dl, du, h, h2
        Integer                              :: j
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cmplx
!       .. Executable Statements ..
        h = cmplx(nx+1,kind=nag_wp)
        h2 = h*h
        dd = four*h2
        dl = -h2 - half*rho*h
        du = -h2 + half*rho*h

        y(1) = dd*x(1) + du*x(2)
        Do j = 2, nx - 1
          y(j) = dl*x(j-1) + dd*x(j) + du*x(j+1)
        End Do
        y(nx) = dl*x(nx-1) + dd*x(nx)
        Return
      End Subroutine tv
      Subroutine av(nx,v,w)

!       .. Use Statements ..
        Use nag_library, Only: zaxpy
!       .. 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)                :: h2
        Integer                              :: j, lo
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cmplx
!       .. Executable Statements ..
        h2 = cmplx(-(nx+1)*(nx+1),kind=nag_wp)

        Call tv(nx,v(1),w(1))
!       The NAG name equivalent of zaxpy is f06gcf
        Call zaxpy(nx,h2,v(nx+1),1,w(1),1)

        Do j = 2, nx - 1
          lo = (j-1)*nx
          Call tv(nx,v(lo+1),w(lo+1))
          Call zaxpy(nx,h2,v(lo-nx+1),1,w(lo+1),1)
          Call zaxpy(nx,h2,v(lo+nx+1),1,w(lo+1),1)
        End Do

        lo = (nx-1)*nx
        Call tv(nx,v(lo+1),w(lo+1))
        Call zaxpy(nx,h2,v(lo-nx+1),1,w(lo+1),1)

        Return
      End Subroutine av
    End Module f12anfe_mod
    Program f12anfe

!     F12ANF Example Main Program
```

```
!     .. Use Statements ..
      Use nag_library, Only: dznrm2, f12anf, f12apf, f12aqf, f12arf, f12asf,   &
                             nag_wp
      Use f12anfe_mod, Only: av, imon, ipoint, licomm, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Complex (Kind=nag_wp)                :: sigma
      Integer                              :: i, ifail, ifail1, irevcm, lcomm, &
                                              ldv, n, nconv, ncv, nev, niter,  &
                                              nshift, nx
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable   :: ax(:), comm(:), d(:,:), mx(:),   &
                                              resid(:), v(:,:), x(:)
      Integer                              :: icomm(licomm)
!     .. Executable Statements ..
      Write (nout,*) 'F12ANF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nx, nev, ncv

      n = nx*nx
      ldv = n
      lcomm = 3*n + 3*ncv*ncv + 5*ncv + 60
      Allocate (ax(n),comm(lcomm),d(ncv,2),mx(n),resid(n),v(n,ncv),x(n))

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

      If (ipoint/=0) Then
!       Use pointers to Workspace in calculating matrix vector
!       products rather than interfacing through the array X.
        ifail = 0
        Call f12arf('POINTERS=YES',icomm,comm,ifail)
      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 .Or. irevcm==1) Then
!         Perform matrix vector multiplication y <--- Op*x.
          If (ipoint==0) Then
            Call av(nx,x,ax)
            x(1:n) = ax(1:n)
          Else
            Call av(nx,comm(icomm(1)),comm(icomm(2)))
          End If
        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
          Write (6,99999) niter, nconv, dznrm2(nev,d(1,2),1)
        End If
      End Do revcm

      If (ifail==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,99998) nconv
        Write (nout,99997)(i,d(i,1),i=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,' Ritz values of largest magnitude are:'/)
99997 Format (1X,I8,5X,'( ',F12.4,' , ',F12.4,' )')
    End Program f12anfe
```

9.2 Program Data

```
F12ANF Example Program Data
 10 4 20 : Vaues for NX NEV and NCV
```

9.3 Program Results

```
 F12ANF Example Program Results


 The    4 Ritz values of largest magnitude are:

        1     (     716.1973 ,   -1029.5838 )
        2     (     716.1973 ,    1029.5838 )
        3     (     687.5834 ,   -1029.5838 )
        4     (     687.5834 ,    1029.5838 )
```
