NAG Library Routine Document

F12FCF

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 11 in F12FDF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12FCF is a post-processing routine in a suite of routines which includes F12FAF, F12FBF, F12FDF and F12FEF. F12FCF must be called following a final exit from F12FBF.

2 Specification

```
SUBROUTINE F12FCF (NCONV, D, Z, LDZ, SIGMA, RESID, V, LDV, COMM, ICOMM,    &
                   IFAIL)

INTEGER            NCONV, LDZ, LDV, ICOMM(*), IFAIL
REAL (KIND=nag_wp) D(*), Z(LDZ,*), SIGMA, RESID(*), V(LDV,*), 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.

Following a call to F12FBF, F12FCF returns the converged approximations to eigenvalues and (optionally) the corresponding approximate eigenvectors and/or an orthonormal basis for the associated approximate invariant subspace. The eigenvalues (and eigenvectors) are selected from those of a standard or generalized eigenvalue problem defined by real symmetric matrices. There is negligible additional cost to obtain eigenvectors; an orthonormal basis is always computed, but there is an additional storage cost if both are requested.

F12FCF is based on the routine **dseupd** from the ARPACK package, which uses the Implicitly Restarted Lanczos iteration method. The method is described in Lehoucq and Sorensen (1996) and Lehoucq (2001) while its use within the ARPACK software is described in great detail in Lehoucq *et al.* (1998). An evaluation of software for computing eigenvalues of sparse symmetric matrices is provided in Lehoucq and Scott (1996). This suite of routines offers the same functionality as the ARPACK software for real symmetric problems, but the interface design is quite different in order to make the option setting clearer and to simplify some of the interfaces.

F12FCF, is a post-processing routine that must be called following a successful final exit from F12FBF. F12FCF uses data returned from F12FBF and options, set either by default or explicitly by calling F12FDF, to return the converged approximations to selected eigenvalues and (optionally):

– the corresponding approximate eigenvectors;

– an orthonormal basis for the associated approximate invariant subspace;

– both.

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: NCONV – INTEGER *Output*

On exit: the number of converged eigenvalues as found by F12FBF.

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

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

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

3: $Z(LDZ,*)$ – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array Z must be at least NCV if the default option **Vectors** $=$ RITZ has been selected and at least 1 if the option **Vectors** $=$ NONE or SCHUR has been selected (see F12FAF).

On exit: if the default option **Vectors** $=$ RITZ (see F12FDF) has been selected then Z contains the final set of eigenvectors corresponding to the eigenvalues held in D. The real eigenvector associated with an eigenvalue is stored in the corresponding column of Z.

4: LDZ – INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which F12FCF is called.

Constraints:

if the default option **Vectors** $=$ Ritz has been selected, $\mathrm{LDZ} \geq \mathrm{N}$;
if the option **Vectors** $=$ None or Schur has been selected, $\mathrm{LDZ} \geq 1$.

5: SIGMA – REAL (KIND=nag_wp) *Input*

On entry: if one of the **Shifted Inverse** (see F12FDF) modes has been selected then SIGMA contains the real shift used; otherwise SIGMA is not referenced.

6: $RESID(*)$ – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array RESID must be at least N (see F12FAF).

On entry: must not be modified following a call to F12FBF since it contains data required by F12FCF.

7: $V(LDV,*)$ – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array V must be at least $\max(1, \mathrm{NCV})$ (see F12FAF).

On entry: the NCV columns of V contain the Lanczos basis vectors for OP as constructed by F12FBF.

On exit: if the option **Vectors** $=$ SCHUR has been set, or the option **Vectors** $=$ RITZ has been set and a separate array Z has been passed (i.e., Z does not equal V), then the first NCONV columns of V will contain approximate Schur vectors that span the desired invariant subspace.

8: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F12FCF is called.

Constraint: LDV $\geq n$.

9: COMM($*$) – REAL (KIND=nag_wp) array *Communication Array*

Note: the dimension of the array COMM must be at least $\max(1, \mathrm{LCOMM})$ (see F12FAF).

On initial entry: must remain unchanged from the prior call to F12FAF.

On exit: contains data on the current state of the solution.

10: ICOMM($*$) – INTEGER array *Communication Array*

Note: the dimension of the array ICOMM must be at least $\max(1, \mathrm{LICOMM})$ (see F12FAF).

On initial entry: must remain unchanged from the prior call to F12FAF.

On exit: contains data on the current state of the solution.

11: 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, LDZ $< \max(1, \mathrm{N})$ or LDZ < 1 when no vectors are required.

IFAIL $= 2$

On entry, the option **Vectors** $=$ Select was selected, but this is not yet implemented.

IFAIL $= 3$

The number of eigenvalues found to sufficient accuracy prior to calling F12FCF, as communicated through the parameter ICOMM, is zero.

IFAIL $= 4$

The number of converged eigenvalues as calculated by F12FBF differ from the value passed to it through the parameter ICOMM.

IFAIL $= 5$

Unexpected error during calculation of a tridiagonal form: there was a failure to compute all the converged eigenvalues. Please contact NAG.

IFAIL $= 6$

The routine was unable to dynamically allocate sufficient internal workspace. Please contact NAG.

IFAIL $= 7$

An unexpected error has occurred. Please contact NAG.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The relative accuracy of a Ritz value, λ, is considered acceptable if its Ritz estimate $\leq$ **Tolerance** $\times |\lambda|$. The default **Tolerance** used is the ***machine precision*** given by X02AJF.

8 Parallelism and Performance

F12FCF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F12FCF 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

None.

10 Example

This example solves $Ax = \lambda Bx$ in regular mode, where A and B are obtained from the standard central difference discretization of the one-dimensional Laplacian operator $\frac{d^2u}{dx^2}$ on $[0, 1]$, with zero Dirichlet boundary conditions.

10.1 Program Text

```
!   F12FCF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module f12fcfe_mod

!     F12FCF 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 :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: six = 6.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: zero = 0.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: two = 2.0_nag_wp
      Integer, Parameter, Public           :: imon = 0, licomm = 140, nin = 5, &
                                              nout = 6
    Contains
      Subroutine mv(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)*six)
        w(1) = four*v(1) + v(2)
        Do j = 2, n - 1
          w(j) = v(j-1) + four*v(j) + v(j+1)
        End Do
        j = n
        w(j) = v(j-1) + four*v(j)
!       The NAG name equivalent of dscal is f06edf
        Call dscal(n,h,w,1)
        Return
      End Subroutine mv

      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 f12fcfe_mod
    Program f12fcfe

!     F12FCF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: dgttrf, dgttrs, dnrm2, f12faf, f12fbf, f12fcf,    &
                             f12fdf, f12fef, nag_wp
      Use f12fcfe_mod, Only: av, four, imon, licomm, mv, nin, nout, one, six,  &
                             zero
!     .. 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,*) 'F12FCF 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)

      h = one/real(n+1,kind=nag_wp)
      r1 = (four/six)*h
      r2 = (one/six)*h
      ad(1:n) = r1
      adl(1:n) = 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 .Or. irevcm==1) Then
!         Perform  X <--- OP*x = inv[M]*A*x.
          Call av(n,x,mx)
          x(1:n) = mx(1:n)
```

```
!         The NAG name equivalent of dgttrs is f07cef
          Call dgttrs('N',n,1,adl,ad,adu,adu2,ipiv,x,n,info)
        Else If (irevcm==2) Then
!         Perform  MX <--- M*x.
          Call mv(n,x,mx)
        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.
        sigma = zero
        ifail = 0
        Call f12fcf(nconv,d,v,ldv,sigma,resid,v,ldv,comm,icomm,ifail)
        Write (nout,99998) nconv
        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 of largest magn', &
        'itude are:'/)
99997 Format (1X,I8,5X,F9.1)
    End Program f12fcfe
```

10.2 Program Data

```
F12FCF Example Program Data
 100 4 10 : Values for N NEV and NCV
```

10.3 Program Results

```
 F12FCF Example Program Results


 The    4 generalized Ritz values of largest magnitude are:

        1      121003.5
        2      121616.6
        3      122057.5
        4      122323.2
```