NAG Library Routine Document

F12ADF

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

1 Purpose

F12ADF is an option setting routine that may be used to supply individual optional parameters to F12ABF and F12ACF. These are part of a suite of routines that also includes: F12AAF and F12AEF. The initialization routine F12AAF **must** have been called prior to calling F12ADF.

2 Specification

```
SUBROUTINE F12ADF (STR, ICOMM, COMM, IFAIL)

INTEGER            ICOMM(*), IFAIL
REAL (KIND=nag_wp) COMM(*)
CHARACTER(*)       STR
```

3 Description

F12ADF may be used to supply values for optional parameters to F12ABF and F12ACF. It is only necessary to call F12ADF for those parameters whose values are to be different from their default values. One call to F12ADF sets one parameter value.

Each optional parameter is defined by a single character string consisting of one or more items. The items associated with a given option must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
'Vectors = None'
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

– a mandatory keyword;

– a phrase that qualifies the keyword;

– a number that specifies an integer or real value. Such numbers may be up to 16 contiguous characters in Fortran's I, F, E or D format.

F12ADF does not have an equivalent routine from the ARPACK package which passes options by directly setting values to scalar parameters or to specific elements of array arguments. F12ADF is intended to make the passing of options more transparent and follows the same principle as the single option setting routines in Chapter E04.

The setup routine F12AAF must be called prior to the first call to F12ADF and all calls to F12ADF must precede the first call to F12ABF, the reverse communication iterative solver.

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 10.

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: STR – CHARACTER(*) *Input*

On entry: a single valid option string (as described in Section 3 and Section 10).

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

Note: the dimension of the array ICOMM must be at least max(1, LICOMM) (see F12AAF).

On initial entry: must remain unchanged following a call to the setup routine F12AAF.

On exit: contains data on the current options set.

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

Note: the dimension of the array COMM must be at least 60.

On initial entry: must remain unchanged following a call to the setup routine F12AAF.

On exit: contains data on the current options set.

4: 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

The string passed in STR contains an ambiguous keyword.

IFAIL = 2

The string passed in STR contains a keyword that could not be recognized.

IFAIL = 3

The string passed in STR contains a second keyword that could not be recognized.

IFAIL = 4

The initialization routine F12AAF has not been called or a communication array has become corrupted.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example solves $Ax = \lambda Bx$ in shifted-inverse mode, where A and B are derived from the finite element discretization of the one-dimensional convection-diffusion operator $\frac{d^2u}{dx^2} + \rho\frac{du}{dx}$ on the interval $[0, 1]$, with zero Dirichlet boundary conditions.

The shift σ is a real number, and the operator used in the shifted-inverse iterative process is $\mathrm{OP} = (A - \sigma B)^{-1}B$.

9.1 Program Text

```
!   F12ADF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module f12adfe_mod

!     F12ADF 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, 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].

!       .. Use Statements ..
        Use nag_library, Only: dscal
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: v(n)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: h, vm1, vv
        Integer                              :: j
!       .. Intrinsic Procedures ..
        Intrinsic                            :: real
!       .. Executable Statements ..
        vm1 = v(1)
```

```
        v(1) = (four*v(1)+v(2))/six
        Do j = 2, n - 1
          vv = v(j)
          v(j) = (vm1+four*vv+v(j+1))/six
          vm1 = vv
        End Do
        v(n) = (vm1+four*v(n))/six

        h = one/real(n+1,kind=nag_wp)
!       The NAG name equivalent of dscal is f06edf
        Call dscal(n,h,v,1)
        Return
      End Subroutine mv
    End Module f12adfe_mod

    Program f12adfe

!     F12ADF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: dgttrf, dgttrs, dnrm2, f12aaf, f12abf, f12acf,    &
                             f12adf, f12aef, nag_wp
      Use f12adfe_mod, Only: four, imon, mv, nin, nout, one, six, two
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: h, rho, s, s1, s2, s3, sigmai,   &
                                             sigmar
      Integer                             :: ifail, ifail1, info, irevcm, j,  &
                                             lcomm, ldv, licomm, n, nconv,    &
                                             ncv, nev, niter, nshift, nx
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: comm(:), d(:,:), dd(:), dl(:),   &
                                             du(:), du2(:), mx(:), resid(:),  &
                                             v(:,:), x(:)
      Integer, Allocatable                :: icomm(:), ipiv(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: real
!     .. Executable Statements ..
      Write (nout,*) 'F12ADF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nx, nev, ncv, rho, sigmar, sigmai
      n = nx*nx
      ldv = n
      licomm = 140
      lcomm = 3*n + 3*ncv*ncv + 6*ncv + 60
      Allocate (comm(lcomm),d(ncv,3),dd(n),dl(n),du(n),du2(n),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.
      Call f12adf('SHIFTED REAL',icomm,comm,ifail)
!     Set problem type
      ifail = 0
      Call f12adf('GENERALIZED',icomm,comm,ifail)
!     Construct C = A - SIGMA*I, and factor C using DGTTRF/F07CDF.
      h = one/real(n+1,kind=nag_wp)
      s = rho/two
      s1 = -one/h - s - sigmar*h/six
      s2 = two/h - four*sigmar*h/six
      s3 = -one/h + s - sigmar*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 dgttrf is f07cdf
      Call dgttrf(n,dl,dd,du,du2,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)
!           The NAG name equivalent of dgttrs is f07cef
            Call dgttrs('N',n,1,dl,dd,du,du2,ipiv,x,n,info)
          Case (1)
!           Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x.
            Call dgttrs('N',n,1,dl,dd,du,du2,ipiv,mx,n,info)
            x(1:n) = mx(1:n)
          Case (2)
!           Perform  y <--- M*x
            Call mv(n,x)
          Case (4)
            If (imon/=0) Then
!             Output monitoring information
              Call f12aef(niter,nconv,d,d(1,2),d(1,3),icomm,comm)
!             The NAG name equivalent of dnrm2 is f06ejf
              Write (6,99999) niter, nconv, dnrm2(nev,d(1,3),1)
            End If
          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)
!       Print computed eigenvalues.
        Write (nout,99998) nconv
        Do j = 1, nconv
          Write (nout,99997) j, d(j,1), d(j,2)
        End Do
      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 ', &
        'unity are:'/)
99997 Format (1X,I8,5X,'( ',F12.4,' , ',F12.4,' )')
    End Program f12adfe
```

9.2 Program Data

```
F12ADF Example Program Data
 10 4 10 10.0 1.0 0.0 : Values for NX NEV NCV RHO SIGMAR and SIGMAI
```

9.3 Program Results

```
 F12ADF Example Program Results


 The    4 generalized Ritz values closest to unity are:

        1     (      34.8634 ,       0.0000 )
        2     (      64.4479 ,       0.0000 )
        3     (     113.7872 ,       0.0000 )
        4     (     182.9293 ,       0.0000 )
```

10 Optional Parameters

Several optional parameters for the computational routines F12ABF and F12ACF define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of F12ABF and F12ACF these optional parameters have associated *default values* that are appropriate for most problems. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 10.1.

Advisory
Defaults
Exact Shifts
Generalized
Initial Residual
Iteration Limit
Largest Imaginary
Largest Magnitude
Largest Real
List
Monitoring
Nolist
Pointers
Print Level
Random Residual
Regular
Regular Inverse
Shifted Inverse Imaginary
Shifted Inverse Real
Smallest Imaginary
Smallest Magnitude
Smallest Real
Standard
Supplied Shifts
Tolerance
Vectors

Optional parameters may be specified by calling F12ADF before a call to F12ABF, but after a call to F12AAF. One call is necessary for each optional parameter.

All optional parameters you do not specify are set to their default values. Optional parameters you specify are unaltered by F12ABF and F12ACF (unless they define invalid values) and so remain in effect for subsequent calls unless you alter them.

10.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined;

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for ***machine precision*** (see X02AJF).

Keywords and character values are case and white space insensitive.

Advisory i Default $=$ the value returned by X04ABF

The destination for advisory messages.

Defaults

This special keyword may be used to reset all optional parameters to their default values.

Exact Shifts Default
Supplied Shifts

During the Arnoldi iterative process, shifts are applied internally as part of the implicit restarting scheme. The shift strategy used by default and selected by the **Exact Shifts** is strongly recommended over the alternative **Supplied Shifts** (see Lehoucq *et al.* (1998) for details of shift strategies).

If **Exact Shifts** are used then these are computed internally by the algorithm in the implicit restarting scheme.

If **Supplied Shifts** are used then, during the Arnoldi iterative process, you must supply shifts through array arguments of F12ABF when F12ABF returns with IREVCM $= 3$; the real and imaginary parts of the shifts are supplied in X and MX respectively (or in COMM when the option **Pointers** $=$ YES is set). This option should only be used if you are an experienced user since this requires some algorithmic knowledge and because more operations are usually required than for the implicit shift scheme. Details on the use of explicit shifts and further references on shift strategies are available in Lehoucq *et al.* (1998).

Iteration Limit i Default $= 300$

The limit on the number of Arnoldi iterations that can be performed before F12ABF exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Arnoldi iterations has reached this limit then F12ABF exits with an error; F12ACF can still be called subsequently to return the number of converged eigenvalues, the converged eigenvalues and, if requested, the corresponding eigenvectors.

Largest Magnitude Default
Largest Imaginary
Largest Real
Smallest Imaginary
Smallest Magnitude
Smallest Real

The Arnoldi iterative method converges on a number of eigenvalues with given properties. The default is for F12ABF to compute the eigenvalues of largest magnitude using **Largest Magnitude**. Alternatively, eigenvalues may be chosen which have **Largest Real** part, **Largest Imaginary** part,**Smallest Magnitude**, **Smallest Real** part or **Smallest Imaginary** part.

Note that these options select the eigenvalue properties for eigenvalues of OP (and B for **Generalized** problems), the linear operator determined by the computational mode and problem type.

Nolist Default
List

Normally each optional parameter specification is not printed to the advisory channel as it is supplied. Optional parameter **List** may be used to enable printing and optional parameter **Nolist** may be used to suppress the printing.

Monitoring i Default $= -1$

If $i > 0$, monitoring information is output to channel number i during the solution of each problem; this may be the same as the **Advisory** channel number. The type of information produced is dependent on the value of **Print Level**, see the description of the optional parameter **Print Level** for details of the information produced. Please see X04ACF to associate a file with a given channel number.

Pointers Default $=$ NO

During the iterative process and reverse communication calls to F12ABF, required data can be communicated to and from F12ABF in one of two ways. When **Pointers** $=$ NO is selected (the default) then the array arguments X and MX are used to supply you with required data and used to return computed values back to F12ABF. For example, when IREVCM $= 1$, F12ABF returns the vector x in X and the matrix-vector product Bx in MX and expects the result or the linear operation OP(x) to be returned in X.

If **Pointers** $=$ YES is selected then the data is passed through sections of the array argument COMM. The section corresponding to X when **Pointers** $=$ NO begins at a location given by the first element of ICOMM; similarly the section corresponding to MX begins at a location given by the second element of ICOMM. This option allows F12ABF to perform fewer copy operations on each intermediate exit and entry, but can also lead to less elegant code in the calling program.

Print Level i Default $= 0$

This controls the amount of printing produced by F12ADF as follows.

$= 0$ No output except error messages.

> 0 The set of selected options.

$= 2$ Problem and timing statistics on final exit from F12ABF.

≥ 5 A single line of summary output at each Arnoldi iteration.

≥ 10 If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the length and additional steps of the current Arnoldi factorization and the number of converged Ritz values; during re-orthogonalization, the norm of initial/restarted starting vector.

≥ 20 Problem and timing statistics on final exit from F12ABF. If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the number of shifts being applied, the eigenvalues and estimates of the Hessenberg matrix H, the size of the Arnoldi basis, the wanted Ritz values and associated Ritz estimates and the shifts applied; vector norms prior to and following re-orthogonalization.

≥ 30 If **Monitoring** > 0, **Monitoring** is set, then on final iteration, the norm of the residual; when computing the Schur form, the eigenvalues and Ritz estimates both before and after sorting; for each iteration, the norm of residual for compressed factorization and the compressed upper Hessenberg matrix H; during re-orthogonalization, the initial/restarted starting vector; during the Arnoldi iteration loop, a restart is flagged and the number of the residual requiring iterative refinement; while applying shifts, the indices of the shifts being applied.

≥ 40 If **Monitoring** > 0, **Monitoring** is set, then during the Arnoldi iteration loop, the Arnoldi vector number and norm of the current residual; while applying shifts, key measures of progress and the order of H; while computing eigenvalues of H, the last rows of the Schur and eigenvector matrices; when computing implicit shifts, the eigenvalues and Ritz estimates of H.

≥ 50 If **Monitoring** is set, then during Arnoldi iteration loop: norms of key components and the active column of H, norms of residuals during iterative refinement, the final upper Hessenberg matrix H; while applying shifts: number of shifts, shift values, block indices, updated matrix H; while computing eigenvalues of H: the matrix H, the computed eigenvalues and Ritz estimates.

Random Residual Default
Initial Residual

To begin the Arnoldi iterative process, F12ABF requires an initial residual vector. By default F12ABF provides its own random initial residual vector; this option can also be set using optional parameter **Random Residual**. Alternatively, you can supply an initial residual vector (perhaps from a previous computation) to F12ABF through the array argument RESID; this option can be set using optional parameter **Initial Residual**.

Regular Default
Regular Inverse
Shifted Inverse Imaginary
Shifted Inverse Real

These options define the computational mode which in turn defines the form of operation OP(x) to be performed when F12ABF returns with IREVCM $= -1$ or 1 and the matrix-vector product Bx when F12ABF returns with IREVCM $= 2$.

Given a **Standard** eigenvalue problem in the form $Ax = \lambda x$ then the following modes are available with the appropriate operator OP(x).

Regular	OP $= A$
Shifted Inverse Real	OP $= (A - \sigma I)^{-1}$ where σ is real

Given a **Generalized** eigenvalue problem in the form $Ax = \lambda Bx$ then the following modes are available with the appropriate operator OP(x).

Regular Inverse	OP $= B^{-1}A$
Shifted Inverse Real with real shift	OP $= (A - \sigma B)^{-1}B$, where σ is real
Shifted Inverse Real with complex shift	OP $= \text{Real}\left((A - \sigma B)^{-1}B\right)$, where σ is complex
Shifted Inverse Imaginary	OP $= \text{Imag}\left((A - \sigma B)^{-1}B\right)$, where σ is complex

Standard Default
Generalized

The problem to be solved is either a standard eigenvalue problem, $Ax = \lambda x$, or a generalized eigenvalue problem, $Ax = \lambda Bx$. The optional parameter **Standard** should be used when a standard eigenvalue problem is being solved and the optional parameter **Generalized** should be used when a generalized eigenvalue problem is being solved.

Tolerance r Default $= \epsilon$

An approximate eigenvalue has deemed to have converged when the corresponding Ritz estimate is within **Tolerance** relative to the magnitude of the eigenvalue.

Vectors Default = RITZ

The routine F12ACF can optionally compute the Schur vectors and/or the eigenvectors corresponding to the converged eigenvalues. To turn off computation of any vectors the option **Vectors** = NONE should be set. To compute only the Schur vectors (at very little extra cost), the option **Vectors** = SCHUR should be set and these will be returned in the array argument V of F12ACF. To compute the eigenvectors (Ritz vectors) corresponding to the eigenvalue estimates, the option **Vectors** = RITZ should be set and these will be returned in the array argument Z of F12ACF, if Z is set equal to V (as in Section 9) then the Schur vectors in V are overwritten by the eigenvectors computed by F12ACF.