NAG Library Routine Document

F12ARF

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

F12ARF is an option setting routine in a suite of routines consisting of F12ANF, F12APF, F12AQF, F12ARF and F12ASF, and may be used to supply individual optional parameters to F12APF and F12AQF. The initialization routine F12ANF **must** have been called prior to calling F12ARF.

2 Specification

```
SUBROUTINE F12ARF (STR, ICOMM, COMM, IFAIL)

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

3 Description

F12ARF may be used to supply values for optional parameters to F12APF and F12AQF. It is only necessary to call F12ARF for those parameters whose values are to be different from their default values. One call to F12ARF 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

```
'Pointers = Yes'
```

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.

F12ARF 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. F12ARF is intended to make the passing of options more transparent and follows the same principle as the single option setting routines in Chapter E04 (see E04NSF for an example).

The setup routine F12ANF must be called prior to the first call to F12ARF and all calls to F12ARF must precede the first call to F12APF, 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 $\max(1, \text{LICOMM})$ (see F12ANF).

On initial entry: must remain unchanged following a call to the setup routine F12ANF.

On exit: contains data on the current options set.

3: COMM($*$) – COMPLEX (KIND=nag_wp) array *Communication Array*

Note: the dimension of the array COMM must be at least $\max(1, \text{LCOMM})$ (see F12ANF).

On initial entry: must remain unchanged following a call to the setup routine F12ANF.

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: $\text{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\text{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\text{IFAIL} = 1$

The string passed in STR contains an ambiguous keyword.

$\text{IFAIL} = 2$

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 F12ANF 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.

9.1 Program Text

```
!   F12ARF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module f12arfe_mod

!     F12ARF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter     :: four = (4.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter     :: one = (1.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter     :: six = (6.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter     :: two = (2.0_nag_wp,0.0_nag_wp)
      Integer, Parameter                   :: imon = 0, 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 f12arfe_mod
      Program f12arfe

!       F12ARF Example Main Program

!       .. Use Statements ..
        Use nag_library, Only: dznrm2, f12anf, f12apf, f12aqf, f12arf, f12asf,   &
                               nag_wp, zgttrf, zgttrs
        Use f12arfe_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
        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,*) 'F12ARF 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 (comm(lcomm),ax(n),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 = (500.0_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
            Write (6,99997) niter, nconv, dznrm2(nev,d(1,2),1)
          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 =',E16.8)
  99996 Format (1X/' The ',I4,' generalized Ritz values closest to (',F7.3,',', &
          F7.3,') are:'/)
  99995 Format (1X,I8,5X,'( ',F10.4,' , ',F10.4,' )')
      End Program f12arfe
```

9.2 Program Data

```
F12ARF Example Program Data
 10 4 20 : Vaues for NX NEV and NCV
```

9.3 Program Results

```
F12ARF Example Program Results


The    4 generalized Ritz values closest to (500.000,  0.000) are:

       1     (   509.9390 ,     0.0000 )
       2     (   380.9092 ,     0.0000 )
       3     (   659.1558 ,     0.0000 )
       4     (   271.9412 ,    -0.0000 )
```

10 Optional Parameters

Several optional parameters for the computational routines F12APF and F12AQF define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of F12APF and F12AQF 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
Smallest Imaginary
Smallest Magnitude
Smallest Real
Standard
Supplied Shifts
Tolerance
Vectors

Optional parameters may be specified by calling F12ARF before a call to F12APF, but after a call to F12ANF. One call is necessary for each optional parameter.

Any optional parameters yoyu do not specify are set to their default values. Optional parameters you do specify are unaltered by F12APF and F12AQF (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 as part of the implicit restarting scheme. The shift strategy used by default and selected by the optional parameter **Exact Shifts** is strongly recommended over the alternative **Supplied Shifts**.

If **Exact Shifts** are used then these are computed internally by the algorithm in the implicit restarting scheme. This strategy is generally effective and cheaper to apply in terms of number of operations than using explicit shifts.

If **Supplied Shifts** are used then, during the Arnoldi iterative process, you must supply shifts through array arguments of F12APF when F12APF returns with IREVCM $= 3$; the complex shifts are returned in X (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 F12APF exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Arnoldi iterations has reached this limit then F12APF exits with an error; F12AQF 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 F12APF 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 F12APF, required data can be communicated to and from F12APF 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 F12APF. For example, when IREVCM $= 1$, F12APF 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 F12APF 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 F12ARF as follows.

$= 0$	No output except error messages.
> 0	The set of selected options.
$= 2$	Problem and timing statistics on final exit from F12APF.
≥ 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 F12APF. 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, F12APF requires an initial residual vector. By default F12APF 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 F12APF through the array argument RESID; this option can be set using optional parameter **Initial Residual**.

Regular Default
Regular Inverse
Shifted Inverse

These options define the computational mode which in turn defines the form of operation OP(x) to be performed when F12APF returns with IREVCM $= -1$ or 1 and the matrix-vector product Bx when F12APF 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	OP $= (A - \sigma I)^{-1}$

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	OP $= (A - \sigma B)^{-1}B$

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 F12AQF 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 F12AQF. 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 F12AQF, if Z is set equal to V (as in Section 9) then the Schur vectors in V are overwritten by the eigenvectors computed by F12AQF.