NAG Fortran 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.

1 Purpose

F12ARF is an option setting routine in a suite of routines consisting of F12ARF, F12ANF, F12APF, F12AQF 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*16          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) a mandatory keyword;

(b) a phrase that qualifies the keyword;

(c) a number that specifies an INTEGER or ***double precision*** 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.

The setup routine F12ANF must be called prior to the first call to F12ARF and all calls to F12ARF must preceed 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 above and in Section 10).

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

ICOMM, on initial entry, must remain unchanged following a call to the setup routine F12ANF.

3: COMM($*$) – ***complex*16*** array *Communication Array*

COMM, on initial entry, must remain unchanged following a call to the setup routine F12ANF.

4: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

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, for users 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.**

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 F12ANF has not been called or a communication array has become corrupted.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

The 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 21 Release. NAG Copyright 2004.
*     .. Parameters ..
      INTEGER          IMON, LICOMM, NERR, NIN, NOUT
      PARAMETER        (IMON=0,LICOMM=140,NERR=6,NIN=5,NOUT=6)
      INTEGER          MAXN, MAXNCV, LDV
      PARAMETER        (MAXN=256,MAXNCV=30,LDV=MAXN)
      INTEGER          LCOMM
      PARAMETER        (LCOMM=3*MAXN+3*MAXNCV*MAXNCV+5*MAXNCV+60)
      COMPLEX *16      ONE, TWO, FOUR, SIX
      PARAMETER        (ONE=(1.0D+0,0.0D+0),TWO=(2.0D+0,0.0D+0),
     +                 FOUR=(4.0D+0,0.0D+0),SIX=(6.0D+0,0.0D+0))
*     .. Local Scalars ..
      COMPLEX *16      H, RHO, S, S1, S2, S3, SIGMA
      INTEGER          IFAIL, IFAIL1, INFO, IREVCM, J, N, NCONV, NCV,
     +                 NEV, NITER, NSHIFT, NX
*     .. Local Arrays ..
      COMPLEX *16      AX(MAXN), COMM(LCOMM), D(MAXNCV,2), DD(MAXN),
     +                 DL(MAXN), DU(MAXN), DU2(MAXN), MX(MAXN),
     +                 RESID(MAXN), V(LDV,MAXNCV), X(MAXN)
      INTEGER          ICOMM(LICOMM), IPIV(MAXN)
*     .. External Functions ..
      DOUBLE PRECISION DZNRM2
      EXTERNAL         DZNRM2
*     .. External Subroutines ..
      EXTERNAL         F12ANF, F12APF, F12AQF, F12ARF, F12ASF, MV,
     +                 ZCOPY, ZGTTRF, ZGTTRS
*     .. Intrinsic Functions ..
      INTRINSIC        DCMPLX
*     .. 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
      IF (N.LT.1 .OR. N.GT.MAXN) THEN
         WRITE (NOUT,99999) 'N is out of range: N = ', N
      ELSE IF (NCV.GT.MAXNCV) THEN
         WRITE (NOUT,99999) 'NCV is out of range: NCV = ', NCV
      ELSE
         IFAIL = 0
         CALL F12ANF(N,NEV,NCV,ICOMM,LICOMM,COMM,LCOMM,IFAIL)
*     Set the mode.
         CALL F12ARF('SHIFTED INVERSE',ICOMM,COMM,IFAIL)
*     Set problem type.
         CALL F12ARF('GENERALIZED',ICOMM,COMM,IFAIL)
*      SIGMA = ONE
         SIGMA = (5.0D+2,0.0D0)
         RHO = (1.0D+1,0.0D0)
         H = ONE/DCMPLX(N+1)
         S = RHO/TWO
         S1 = -ONE/H - S - SIGMA*H/SIX
         S2 = TWO/H - FOUR*SIGMA*H/SIX
         S3 = -ONE/H + S - SIGMA*H/SIX
*
         DO 20 J = 1, N - 1
            DL(J) = S1
            DD(J) = S2
            DU(J) = S3
   20    CONTINUE
```

```
         DD(N) = S2
*
         CALL ZGTTRF(N,DL,DD,DU,DU2,IPIV,INFO)
         IF (INFO.NE.0) THEN
            WRITE (NERR,99998) INFO
            GO TO 80
         END IF
*
         IREVCM = 0
         IFAIL = -1
   40    CONTINUE
         CALL F12APF(IREVCM,RESID,V,LDV,X,MX,NSHIFT,COMM,ICOMM,IFAIL)
         IF (IREVCM.NE.5) THEN
            IF (IREVCM.EQ.-1) THEN
*           Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x
               CALL MV(NX,X,AX)
               CALL ZCOPY(N,AX,1,X,1)
               CALL ZGTTRS('N',N,1,DL,DD,DU,DU2,IPIV,X,N,INFO)
               IF (INFO.NE.0) THEN
                  WRITE (NERR,99997) INFO
                  GO TO 80
               END IF
            ELSE IF (IREVCM.EQ.1) THEN
*           Perform  x <--- OP*x = inv[A-SIGMA*M]*M*x,
*           MX stored in COMM from location IPNTR(3)
               CALL ZGTTRS('N',N,1,DL,DD,DU,DU2,IPIV,MX,N,INFO)
               CALL ZCOPY(N,MX,1,X,1)
               IF (INFO.NE.0) THEN
                  WRITE (NERR,99997) INFO
                  GO TO 80
               END IF
            ELSE IF (IREVCM.EQ.2) THEN
*           Perform  y <--- M*x
               CALL MV(NX,X,AX)
               CALL ZCOPY(N,AX,1,X,1)
            ELSE IF (IREVCM.EQ.4 .AND. IMON.NE.0) THEN
*           Output monitoring information
               CALL F12ASF(NITER,NCONV,D,D(1,2),ICOMM,COMM)
               WRITE (6,99996) NITER, NCONV, DZNRM2(NEV,D(1,2),1)
            END IF
            GO TO 40
         END IF
         IF (IFAIL.EQ.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,99994) NCONV, SIGMA
            DO 60 J = 1, NCONV
               WRITE (NOUT,99993) J, D(J,1)
   60       CONTINUE
         ELSE
            WRITE (NOUT,99995) IFAIL
         END IF
   80    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,'** Error status returned by ZGTTRF, INFO =',I12)
99997 FORMAT (1X,'** Error status returned by ZGTTRS, INFO =',I12)
99996 FORMAT (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o',
     +       'f estimates =',E16.8)
99995 FORMAT (1X,' NAG Routine F12APF Returned with IFAIL = ',I6)
99994 FORMAT (1X,/' The ',I4,' generalized Ritz values closest to',' (',
     +       F7.3,',',F7.3,') are:',/)
99993 FORMAT (1X,I8,5X,'( ',F10.4,' , ',F10.4,' )')
      END
*
      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].
*
*     .. Parameters ..
      COMPLEX *16   ONE, FOUR, SIX
      PARAMETER     (ONE=(1.0D+0,0.0D+0),FOUR=(4.0D+0,0.0D+0),
     +              SIX=(6.0D+0,0.0D+0))
*     .. Scalar Arguments ..
      INTEGER       NX
*     .. Array Arguments ..
      COMPLEX *16   V(NX*NX), W(NX*NX)
*     .. Local Scalars ..
      COMPLEX *16   H
      INTEGER       J, N
*     .. External Subroutines ..
      EXTERNAL      ZSCAL
*     .. Intrinsic Functions ..
      INTRINSIC     DCMPLX
*     .. Executable Statements ..
      N = NX*NX
      W(1) = (FOUR*V(1)+V(2))/SIX
      DO 20 J = 2, N - 1
         W(J) = (V(J-1)+FOUR*V(J)+V(J+1))/SIX
   20 CONTINUE
      W(N) = (V(N-1)+FOUR*V(N))/SIX
*
      H = ONE/DCMPLX(N+1)
      CALL ZSCAL(N,H,W,1)
      RETURN
      END
```

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, the user need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped by users who wish to use the default values for *all* optional parameters. A complete list of optional parameters and their default values is given in Section 10.1.

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

All optional parameters not specified by the user are set to their default values. Optional parameters specified by the user are unaltered by F12APF and F12AQF (unless they define invalid values) and so remain in effect for subsequent calls unless altered by the user.

10.1 Optional parameter checklist and default values

The following list gives the valid options. For each option, we give the keyword, any essential optional qualifiers and the default value. A definition for each option can be found in Section 10.2. The minimum abbreviation of each keyword is underlined. The qualifier may be omitted. The letters i and r denote INTEGER and ***double precision*** values required with certain options. The number ϵ is a generic notation for ***machine precision*** (see X02AJF).

Optional Parameters	Default Values
Advisory	Default = the value returned by X04ABF
Defaults	
Exact Shifts	Default = **Exact Shifts**
Generalized	
Initial Residual	
Iteration Limit	Default = 300
Largest Imaginary	
Largest Magnitude	Default = **Largest Magnitude**
Largest Real	
List	Default = **Nolist**
Monitoring	Default = -1
Nolist	
Pointers	Default = No
Print Level	Default = 0
Random Residual	Default = **Random Residual**
Regular	Default = **Regular**
Regular Inverse	
Shifted Inverse	
Smallest Imaginary	
Smallest Magnitude	
Smallest Real	
Standard	Default = **Standard**
Supplied Shifts	
Tolerance	Default = ϵ
Vectors	Default = Schur

10.2 Description of the Optional Parameters

Advisory i Default = the value returned by X04ABF

The output channel for advisory messages.

Defaults

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

Exact Shifts Default = **Exact Shifts**
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 option **Exact Shifts** is strongly recommended over the alternative option **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 by experienced users 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 a 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 Magnitude**
Largest Real
Largest Imaginary
Smallest Magnitude
Smallest Real
Smallest Imaginary

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 option **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.

List Default = **Nolist**
Nolist

Normally each optional parameter specification is not printed to the advisory channel as it is supplied. **List** may be used to enable printing and **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 **Print Level** in this section 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 of 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. If you want to suppress all output, set **Print Level** $= 0$.
- ≥ 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, then at each iteration, the length and additional steps of the current Arnoldi factorization and the number of converged Ritz values; during re-orthogonalisation, the norm of initial/restarted starting vector.

≥ 20 Problem and timing statistics on final exit from F12APF. If **Monitoring** > 0, 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-orthogonalisation.

≥ 30 If **Monitoring** > 0, 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-orthogonalisation, 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, some indices.

≥ 40 If **Monitoring** > 0, 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 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.

Note that setting **Print Level** ≥ 30 can result in very lengthy **Monitoring** output.

Random Residual Default = **Random Residual**
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 **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 **Random Residual**.

Regular Default = **Regular**
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 IREVCM $= 1$ 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 = **Standard**
Generalized

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

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 the array argument V is passed to F12AQF in place of Z then the Schur vectors in V are overwritten by the eigenvectors computed by F12AQF.