NAG Library Routine Document

F12APF

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 F12ARF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 10 in F12ARF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12APF is an iterative solver 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.

2 Specification

```
 SUBROUTINE F12APF(IREVCM, RESID, V, LDV, X, MX, NSHIFT, COMM, ICOMM,
1                  IFAIL)
 INTEGER           IREVCM, LDV, NSHIFT, ICOMM(*), IFAIL
 complex*16        RESID(*), V(LDV,*), X(*), MX(*), 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, complex and nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, complex and nonsymmetric problems.

F12APF is a **reverse communication** routine, based on the ARPACK routine **znaupd**, using the Implicitly Restarted Arnoldi 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 nonsymmetric matrices is provided in Lehoucq and Scott (1996). This suite of routines offers the same functionality as the ARPACK software for complex nonsymmetric problems, but the interface design is quite different in order to make the option setting clearer and to simplify the interface of F12APF.

The setup routine F12ANF must be called before F12APF, the reverse communication iterative solver. Options may be set for F12APF by prior calls to the option setting routine F12ARF and a post-processing routine F12AQF must be called following a successful final exit from F12APF. F12ASF may be called following certain flagged intermediate exits from F12APF to provide additional monitoring information about the computation.

F12APF uses **reverse communication**, i.e., it returns repeatedly to the calling program with the parameter IREVCM (see Section 5) set to specified values which require the calling program to carry out one of the following tasks:

– compute the matrix-vector product $y = \mathrm{OP}x$, where OP is defined by the computational mode;

– compute the matrix-vector product $y = Bx$;

– notify the completion of the computation;

– allow the calling program to monitor the solution.

The problem type to be solved (standard or generalized), the spectrum of eigenvalues of interest, the mode used (regular, regular inverse, shifted inverse, shifted real or shifted imaginary) and other options can all be set using the option setting routine F12ARF (see Section 10.2 in F12ARF for details on setting options and of the default settings).

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

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the **parameter IREVCM**. Between intermediate exits and re-entries, **all parameters other than X, MX and COMM must remain unchanged**.

1: IREVCM – INTEGER *Input/Output*

On initial entry: IREVCM $= 0$, otherwise an error condition will be raised.

On intermediate re-entry: must be unchanged from its previous exit value. Changing IREVCM to any other value between calls will result in an error.

On intermediate exit: has the following meanings.

-1 The calling program must compute the matrix-vector product $y = \mathrm{OP}x$, where x is stored in X (by default) or in the array COMM (starting from the location given by the first element of ICOMM) when the option Pointers $=$ Yes is set in a prior call to F12ARF. The result y is returned in X (by default) or in the array COMM (starting from the location given by the second element of ICOMM) when the option Pointers $=$ Yes is set.

1 The calling program must compute the matrix-vector product $y = \mathrm{OP}x$. This is similar to the case IREVCM $= -1$ except that the result of the matrix-vector product $y = Bx$ (as required in some computational modes) has already been computed and is available in MX (by default) or in the array COMM (starting from the location given by the third element of ICOMM) when the option Pointers $=$ Yes is set.

2 The calling program must compute the matrix-vector product $y = Bx$, where x is stored in X and y is returned in MX (by default) or in the array COMM (starting from the location given by the second element of ICOMM) when the option Pointers $=$ Yes is set.

3 Compute the NSHIFT complex shifts. This value of IREVCM will only arise if the optional parameter Supplied Shifts is set in a prior call to F12ARF which is intended for experienced users only; the default and recommended option is to use exact shifts (see Lehoucq *et al.* (1998) for details).

4 Monitoring step: a call to F12ASF can now be made to return the number of Arnoldi iterations, the number of converged Ritz values, the array of converged values, and the corresponding Ritz estimates.

On final exit: IREVCM $= 5$: F12APF has completed its tasks. The value of IFAIL determines whether the iteration has been successfully completed, or whether errors have been detected. On successful completion F12AQF must be called to return the requested eigenvalues and eigenvectors (and/or Schur vectors).

Constraint: on initial entry, IREVCM $= 0$; on re-entry IREVCM must remain unchanged.

2: RESID($*$) – ***complex*16*** array *Input/Output*

Note: the dimension of the array RESID must be at least N (see F12ANF).

On initial entry: need not be set unless the option Initial Residual has been set in a prior call to F12ARF in which case RESID should contain an initial residual vector, possibly from a previous run.

On intermediate re-entry: must be unchanged from its previous exit. Changing RESID to any other value between calls may result in an error exit.

On intermediate exit: contains the current residual vector.

On final exit: contains the final residual vector.

3: V(LDV,$*$) – ***complex*16*** array *Input/Output*

Note: the second dimension of the array V must be at least max$(1, \mathrm{NCV})$ (see F12ANF).

On initial entry: need not be set.

On intermediate re-entry: must be unchanged from its previous exit.

On intermediate exit: contains the current set of Arnoldi basis vectors.

On final exit: contains the final set of Arnoldi basis vectors.

4: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F12APF is called.

Constraint: LDV $\geq$ N.

5: X($*$) – ***complex*16*** array *Input/Output*

Note: the dimension of the array X must be at least N if Pointers = No (default) and at least 1 if Pointers = Yes (see F12ANF).

On initial entry: need not be set, it is used as a convenient mechanism for accessing elements of COMM.

On intermediate re-entry: if Pointers = Yes, X need not be set.

If Pointers = No, X must contain the result of $y = \mathrm{OP}x$ when IREVCM returns the value -1 or $+1$. It must return the computed shifts when IREVCM returns the value 3.

On intermediate exit: if Pointers = Yes, X is not referenced.

If Pointers = No, X contains the vector x when IREVCM returns the value -1 or $+1$.

On final exit: does not contain useful data.

6: MX($*$) – ***complex*16*** array *Input/Output*

Note: the dimension of the array MX must be at least N if Pointers = No (default) and at least 1 if Pointers = Yes (see F12ANF).

On initial entry: need not be set, it is used as a convenient mechanism for accessing elements of COMM.

On intermediate re-entry: if Pointers = Yes, MX need not be set.

If Pointers = No, MX must contain the result of $y = Bx$ when IREVCM returns the value 2.

On intermediate exit: if Pointers = Yes, MX is not referenced.

If Pointers = No, MX contains the vector Bx when IREVCM returns the value $+1$.

On final exit: does not contain any useful data.

7: NSHIFT – INTEGER *Output*

On intermediate exit: if the option Supplied Shifts is set and IREVCM returns a value of 3, NSHIFT returns the number of complex shifts required.

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

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

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

On exit: contains data defining the current state of the iterative process.

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

Note: the dimension of the array ICOMM must be at least $\max(1, \mathrm{LICOMM})$ (see F12ANF).

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

On exit: contains data defining the current state of the iterative process.

10: IFAIL – INTEGER *Input/Output*

On initial 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.

On final exit: $\mathrm{IFAIL} = 0$ 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, because for this routine the values of the output parameters may be useful even if $\mathrm{IFAIL} \neq 0$ on exit, the recommended value is -1. **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 $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On initial entry, the maximum number of iterations ≤ 0, the option Iteration Limit has been set to a nonpositive value.

$\mathrm{IFAIL} = 2$

The options Generalized and Regular are incompatible.

$\mathrm{IFAIL} = 3$

The option Initial Residual was selected but the starting vector held in RESID is zero.

$\mathrm{IFAIL} = 4$

The maximum number of iterations has been reached. Some Ritz values may have converged; a subsequent call to F12AQF will return the number of converged values and the converged values.

$\mathrm{IFAIL} = 5$

No shifts could be applied during a cycle of the implicitly restarted Arnoldi iteration. One possibility is to increase the size of NCV relative to NEV (see Section 5 in F12ANF for details of these parameters).

IFAIL $= 6$

Could not build a factorization. Consider changing NCV or NEV in the initialization routine (see Section 5 in F12ANF for details of these parameters).

IFAIL $= 7$

Unexpected error in internal call to compute eigenvalues and corresponding error bounds of the current upper Hessenberg matrix. Please contact NAG.

IFAIL $= 8$

Either the initialization routine F12ANF has not been called prior to the first call of this routine or a communication array has become corrupted.

IFAIL $= 9$

An unexpected error has occurred. Please contact NAG.

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 Further Comments

None.

9 Example

This example solves $Ax = \lambda x$ in shift-invert 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 shift used is a complex number.

9.1 Program Text

```
*     F12APF 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, ZERO
      PARAMETER        (ONE=(1.0D+0,0.0D+0),TWO=(2.0D+0,0.0D+0),
     +                 ZERO=(0.0D+0,0.0D+0))
*     .. Local Scalars ..
      COMPLEX *16      H, H2, RHO, S, S1, S2, S3, SIGMA
      INTEGER          IFAIL, IFAIL1, INFO, IREVCM, J, N, NCONV, NCV,
     +                 NEV, NITER, NSHIFT, NX
*     .. Local Arrays ..
      COMPLEX *16      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, ZGTTRF,
     +                 ZGTTRS
*     .. Intrinsic Functions ..
```

```
      INTRINSIC          CMPLX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F12APF 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 = 1
         CALL F12ANF(N,NEV,NCV,ICOMM,LICOMM,COMM,LCOMM,IFAIL)
*
         IF (IFAIL.EQ.0) THEN
*           Set the mode.
            IFAIL = 0
            CALL F12ARF('SHIFTED INVERSE',ICOMM,COMM,IFAIL)
            SIGMA = ZERO
            RHO = (1.0D+1,0.0D+0)
            H = ONE/CMPLX(N+1,KIND=KIND(H))
            H2 = H*H
            S = RHO/TWO
*
            S1 = -ONE/H2 - S/H
            S2 = TWO/H2 - SIGMA
            S3 = -ONE/H2 + S/H
*
            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 .OR. IREVCM.EQ.1) THEN
*                 Perform  x <--- OP*x = inv[A-SIGMA*I]*x
                  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.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
               DO 60 J = 1, NCONV
                  WRITE (NOUT,99993) J, D(J,1)
   60          CONTINUE
```

```
            END IF
         ELSE
            WRITE (NOUT,99995) IFAIL
         END IF
   80    CONTINUE
      END IF
*
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,' ** F12ANF returned with IFAIL = ',I5)
99994 FORMAT (1X,/' The ',I4,' Ritz values of smallest magnitude are:',
     +       /)
99993 FORMAT (1X,I8,5X,'( ',F12.4,' , ',F12.4,' )')
      END
```

9.2 Program Data

```
F12APF Example Program Data
 10 4 20 : Vaues for NX NEV and NCV
```

9.3 Program Results

```
 F12APF Example Program Results


 The    4 Ritz values of smallest magnitude are:

        1     (      34.8720 ,       0.0000 )
        2     (      64.4326 ,       0.0000 )
        3     (     113.6685 ,       0.0000 )
        4     (     182.5320 ,       0.0000 )
```
