

NAG Library Routine Document

F12AAF

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

F12AAF is a setup routine in a suite of routines consisting of F12AAF, F12ABF, F12ACF, F12ADF and F12AEF. It is used to find some of the eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by real nonsymmetric matrices.

The suite of routines is suitable for the solution of large sparse, standard or generalized, nonsymmetric eigenproblems where only a few eigenvalues from a selected range of the spectrum are required.

2 Specification

```
SUBROUTINE F12AAF(N, NEV, NCV, ICOMM, LICOMM, COMM, LCOMM, IFAIL)
  INTEGER          N, NEV, NCV, ICOMM(max(1,LICOMM)), LICOMM, LCOMM,
1 IFAIL
  double precision COMM(max(1,LCOMM))
```

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 nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and nonsymmetric problems.

F12AAF is a setup routine which must be called before F12ABF, the reverse communication iterative solver, and before F12ADF, the options setting routine. F12ACF is a post-processing routine that must be called following a successful final exit from F12ABF, while F12AEF can be used to return additional monitoring information during the computation.

This setup routine initializes the communication arrays, sets (to their default values) all options that can be set by you via the option setting routine F12ADF, and checks that the lengths of the communication arrays as passed by you are of sufficient length. For details of the options available and how to set them see Section 10.2 in F12ADF.

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, Philadelphia

5 Parameters

- 1: N – INTEGER *Input*
On entry: the order of the matrix A (and the order of the matrix B for the generalized problem) that defines the eigenvalue problem.
Constraint: $N > 0$.

- 2: NEV – INTEGER *Input*
On entry: the number of eigenvalues to be computed.
Constraint: $0 < NEV < N - 1$.

- 3: NCV – INTEGER *Input*
On entry: the number of Arnoldi basis vectors to use during the computation.

At present there is no *a priori* analysis to guide the selection of NCV relative to NEV. However, it is recommended that $NCV \geq 2 \times NEV + 1$. If many problems of the same type are to be solved, you should experiment with varying NCV while keeping NEV fixed for a given test problem. This will usually decrease the required number of matrix-vector operations but it also increases the work and storage required to maintain the orthogonal basis vectors. The optimal ‘cross-over’ with respect to CPU time is problem dependent and must be determined empirically.

Constraint: $NEV + 1 < NCV \leq N$.

- 4: ICOMM(max(1, LICOMM)) – INTEGER array *Communication Array*
On exit: contains data to be communicated to the other routines in the suite.

- 5: LICOMM – INTEGER *Input*
On entry: the dimension of the array ICOMM as declared in the (sub)program from which F12AAF is called.

If LICOMM = -1, a workspace query is assumed and the routine only calculates the required dimensions of ICOMM and COMM, which it returns in ICOMM(1) and COMM(1) respectively.

Constraint: LICOMM ≥ 140 or LICOMM = -1.

- 6: COMM(max(1, LCOMM)) – **double precision** array *Communication Array*
On exit: contains data to be communicated to the other routines in the suite.

- 7: LCOMM – INTEGER *Input*
On entry: the dimension of the array COMM as declared in the (sub)program from which F12AAF is called.

If LCOMM = -1, a workspace query is assumed and the routine only calculates the dimensions of ICOMM and COMM required by F12ABF, which it returns in ICOMM(1) and COMM(1) respectively.

Constraint: LCOMM $\geq 3 \times N + 3 \times NCV \times NCV + 6 \times NCV + 60$ or LCOMM = -1.

- 8: 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.
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, 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.**

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, $N \leq 0$.

$IFAIL = 2$

On entry, $NEV \leq 0$.

$IFAIL = 3$

On entry, $NCV < NEV + 2$ or $NCV > N$.

$IFAIL = 4$

On entry, $LICOMM < 140$ and $LICOMM \neq -1$.

$IFAIL = 5$

On entry, $LCOMM < 3 \times N + 3 \times NCV \times NCV + 6 \times NCV + 60$ and $LCOMM \neq -1$.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example solves $Ax = \lambda x$ in regular 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, where $\rho = 100$.

9.1 Program Text

```
*      F12AAF Example Program Text
*      Mark 21 Release. NAG Copyright 2004.
*      .. Parameters ..
      INTEGER          LICOMM, NIN, NOUT
      PARAMETER        (LICOMM=140,NIN=5,NOUT=6)
      INTEGER          MAXN, MAXNCV, LDV
      PARAMETER        (MAXN=256,MAXNCV=30,LDV=MAXN)
      INTEGER          LCOMM
      PARAMETER        (LCOMM=3*MAXN+3*MAXNCV*MAXNCV+6*MAXNCV+60)
      INTEGER          IMON, IPOINT
      PARAMETER        (IMON=0,IPOINT=0)
*      .. Local Scalars ..
      DOUBLE PRECISION SIGMAI, SIGMAR
      INTEGER          I, IFAIL, IFAIL1, IREVCM, N, NCONV, NCV, NEV,
+      NITER, NSHIFT, NX
*      .. Local Arrays ..
      DOUBLE PRECISION AX(MAXN), COMM(LCOMM), D(MAXNCV,3), MX(MAXN),
```

```

+          RESID(MAXN), V(LDV,MAXNCV), X(MAXN)
+      INTEGER          ICOMM(LICOMM)
*      .. External Functions ..
+      DOUBLE PRECISION DNRM2
+      EXTERNAL          DNRM2
*      .. External Subroutines ..
+      EXTERNAL          AV, DCOPY, F12AAF, F12ABF, F12ACF, F12ADF, F12AEF
*      .. Executable Statements ..
+      WRITE (NOUT,*) 'F12AAF 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 F12AAF(N,NEV,NCV,ICOMM,LICOMM,COMM,LCOMM,IFAIL)
*
+          IF (IFAIL.EQ.0) THEN
*              Set the region of the spectrum that is required.
+              IFAIL = 0
+              CALL F12ADF('SMALLEST MAG',ICOMM,COMM,IFAIL)
+              IF (IPOINT.NE.0) THEN
*                  Use pointers to workspace in calculating matrix vector products
*                  rather than interfacing through the array X.
+                  IFAIL = 0
+                  CALL F12ADF('POINTERS=YES',ICOMM,COMM,IFAIL)
+              END IF
*
+              IREVCM = 0
+              IFAIL = -1
20          CONTINUE
+          CALL F12ABF(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 matrix vector multiplication y <--- Op*x
+                  IF (IPOINT.EQ.0) THEN
+                      CALL AV(NX,X,AX)
+                      CALL DCOPY(N,AX,1,X,1)
+                  ELSE
+                      CALL AV(NX,COMM(ICOMM(1)),COMM(ICOMM(2)))
+                  END IF
+              ELSE IF (IREVCM.EQ.4 .AND. IMON.NE.0) THEN
*                  Set IMON=1 to output monitoring information.
+                  CALL F12AEF(NITER,NCONV,D,D(1,2),D(1,3),ICOMM,COMM)
+                  WRITE (6,99998) NITER, NCONV, DNRM2(NEV,D(1,3),1)
+              END IF
+              GO TO 20
+          END IF
+          IF (IFAIL.EQ.0) THEN
*              Post-Process using F12ACF to compute eigenvalues and
*              (by default) the corresponding eigenvectors.
+              IFAIL1 = 0
+              CALL F12ACF(NCONV,D,D(1,2),V,LDV,SIGMAR,SIGMAI,RESID,V,
+                  LDV,COMM,ICOMM,IFAIL1)
+              WRITE (NOUT,99996) NCONV
+              DO 40 I = 1, NCONV
+                  WRITE (NOUT,99995) I, D(I,1), D(I,2)
40          CONTINUE
+          END IF
+          ELSE
+              WRITE (NOUT,99997) IFAIL
+          END IF
+      END IF
*
+      99999 FORMAT (1X,A,I5)
+      99998 FORMAT (1X,'Iteration',1X,I3,',', No. converged =',1X,I3,',', norm o',

```

```

+          'f estimates =',E16.8)
99997 FORMAT (1X,' ** F12AAF returned with IFAIL = ',I5)
99996 FORMAT (1X,/' The ',I4,' Ritz values of smallest magnitude are:',
+          /)
99995 FORMAT (1X,I8,5X,'( ',F12.4,' , ',F12.4,' )')
END

*
SUBROUTINE AV(NX,V,W)
*
.. Scalar Arguments ..
INTEGER          NX
*
.. Array Arguments ..
DOUBLE PRECISION V(NX*NX), W(NX*NX)
*
.. Local Scalars ..
DOUBLE PRECISION NX2
INTEGER          J, LO
*
.. External Subroutines ..
EXTERNAL          DAXPY, TV
*
.. Intrinsic Functions ..
INTRINSIC          DBLE
*
.. Executable Statements ..
NX2 = -DBLE((NX+1)*(NX+1))
CALL TV(NX,V(1),W(1))
CALL DAXPY(NX,NX2,V(NX+1),1,W(1),1)
DO 20 J = 2, NX - 1
    LO = (J-1)*NX
    CALL TV(NX,V(LO+1),W(LO+1))
    CALL DAXPY(NX,NX2,V(LO-NX+1),1,W(LO+1),1)
    CALL DAXPY(NX,NX2,V(LO+NX+1),1,W(LO+1),1)
20 CONTINUE
LO = (NX-1)*NX
CALL TV(NX,V(LO+1),W(LO+1))
CALL DAXPY(NX,NX2,V(LO-NX+1),1,W(LO+1),1)
RETURN
END

*
SUBROUTINE TV(NX,X,Y)
*
Compute the matrix vector multiplication  $y \leftarrow T \cdot x$  where T is a nx
*
by nx tridiagonal matrix with constant diagonals (DD, DL and DU).
*
.. Parameters ..
DOUBLE PRECISION HALF, RHO
PARAMETER          (HALF=0.5D0,RHO=1.0D2)
*
.. Scalar Arguments ..
INTEGER          NX
*
.. Array Arguments ..
DOUBLE PRECISION X(NX), Y(NX)
*
.. Local Scalars ..
DOUBLE PRECISION DD, DL, DU, NX1, NX2
INTEGER          J
*
.. Intrinsic Functions ..
INTRINSIC          DBLE
*
.. Executable Statements ..
NX1 = DBLE(NX+1)
NX2 = NX1*NX1
DD = 4.0D0*NX2
DL = -NX2 - HALF*RHO*NX1
DU = -NX2 + HALF*RHO*NX1
Y(1) = DD*X(1) + DU*X(2)
DO 20 J = 2, NX - 1
    Y(J) = DL*X(J-1) + DD*X(J) + DU*X(J+1)
20 CONTINUE
Y(NX) = DL*X(NX-1) + DD*X(NX)
RETURN
END

```

9.2 Program Data

F12AAF Example Program Data

10 10 30 : Values for NX NEV and NCV

9.3 Program Results

F12AAF Example Program Results

The 10 Ritz values of smallest magnitude are:

1	(	251.8027	,	152.7109	)
2	(	251.8027	,	-152.7109	)
3	(	280.4166	,	152.7109	)
4	(	280.4166	,	-152.7109	)
5	(	325.5237	,	152.7109	)
6	(	325.5237	,	-152.7109	)
7	(	383.4696	,	152.7109	)
8	(	383.4696	,	-152.7109	)
9	(	449.5598	,	152.7109	)
10	(	449.5598	,	-152.7109	)
