NAG Fortran Library Routine Document

F12FBF

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 F12FDF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 10 of the document for F12FDF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12FBF is an iterative solver in a suite of routines consisting of F12FBF, F12FAF, F12FCF, F12FDF and F12FEF. It is used to find some of the eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by real symmetric matrices.

2 Specification

```
      SUBROUTINE F12FBF (IREVCM, RESID, V, LDV, X, MX, NSHIFT, COMM, ICOMM,
     1                   IFAIL)
      INTEGER            IREVCM, LDV, NSHIFT, ICOMM(*), IFAIL
      double precision   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, real and symmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and symmetric problems.

F12FBF is a **reverse communication** routine, based on the ARPACK routine **dsaupd**, using the Implicitly Restarted Arnoldi iteration method, which for symmetric problems reduces to a variant of the Lanczos 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 symmetric matrices is provided in Lehoucq and Scott (1996). This suite of routines offers the same functionality as the ARPACK software for real symmetric problems, but the interface design is quite different in order to make the option setting clearer to you and to simplify the interface of F12FBF.

The setup routine F12FAF must be called before F12FBF, the reverse communication iterative solver. Options may be set for F12FBF by prior calls to the option setting routine F12FDF and a post-processing routine F12FCF must be called following a successful final exit from F12FBF. F12FEF, may be called following certain flagged, intermediate exits from F12FBF to provide additional monitoring information about the computation.

F12FBF 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, Buckling or Cayley) and other options can all be set using the option setting routine F12FDF.

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 F12FDF. 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 real and imaginary parts of the shifts where the real parts are to be returned in the first NSHIFT locations of the array X and the imaginary parts are to be returned in the first NSHIFT locations of the array MX. Only complex conjugate pairs of shifts may be applied and the pairs must be placed in consecutive locations. This value of IREVCM will only arise if the optional parameter **Supplied Shifts** is set in a prior call to F12FDF which is intended for experienced users only; the default and recommended option is to use exact shifts (see Lehoucq *et al.* (1998) for details and guidance on the choice of shift strategies).

4 Monitoring step: a call to F12FEF can now be made to return the number of Arnoldi iterations, the number of converged Ritz values, their real and imaginary parts, and the corresponding Ritz estimates.

On final exit: IREVCM $= 5$: F12FBF 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 F12FCF 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($*$) – ***double precision*** array *Input/Output*

Note: the dimension of the array RESID must be at least N (see F12FAF).

On initial entry: need not be set unless the option **Initial Residual** has been set in a prior call to F12FDF 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,$*$) – ***double precision*** array *Input/Output*

Note: the second dimension of the array V must be at least $\max(1, \mathrm{NCV})$ (see F12FAF).

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 F12FBF is called.

Constraint: LDV $\geq$ N.

5: X($*$) – ***double precision*** 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.

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 real parts of 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($*$) – ***double precision*** 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.

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. It must return the imaginary parts of the computed shifts when IREVCM returns the value 3.

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($*$) – ***double precision*** array *Communication Array*

Note: the dimension of the array COMM must be at least $\max(1, \text{LCOMM})$ (see F12FAF).

On initial entry: must remain unchanged following a call to the setup routine F12FAF.

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 $\max(1, \text{LICOMM})$ (see F12FAF).

On initial entry: must remain unchanged following a call to the setup routine F12FAF.

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

10: 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 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, 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 initial entry, the maximum number of iterations ≤ 0, the option **Iteration Limit** has been set to a non-positive value.

IFAIL = 2

The options **Generalized** and **Regular** are incompatible.

IFAIL = 3

Eigenvalues from both ends of the spectrum were requested, but the number of eigenvalues requested is one.

IFAIL = 4

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

IFAIL = 5

The maximum number of iterations has been reached. Some Ritz values may have converged; a subsequent call to F12FCF will return the number of converged values and the converged values.

IFAIL = 6

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 of the document for F12FAF for details of these parameters).

IFAIL = 7

Could not build a factorization. Consider changing NCV or NEV in the initialization routine (see Section 5 of the document for F12FAF for details of these parameters).

IFAIL = 8

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

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

For this routine two examples are presented, with a main program and two example problems given in the (sub)programs EX1 and EX2.

Example 1 (EX1)

The example solves $Ax = \lambda x$ in shift-invert mode, where A is obtained from the standard central difference discretization of the one-dimensional Laplacian operator $\frac{\partial^2 u}{\partial x^2}$ with zero Dirichlet boundary conditions. Eigenvalues closest to the shift $\sigma = 0$ are sought.

Example 2 (EX2)

This example illustrates the use of F12FBF to compute the leading terms in the singular-value decomposition of a real general matrix A. The example finds a few of the largest singular values (σ) and corresponding right singular values (ν) for the matrix A by solving the symmetric problem:

$$\left(A^{\mathrm{T}}A\right)\nu = \sigma\nu.$$

Here A is the m by n real matrix derived from the simplest finite difference discretization of the two-dimensional kernal $k(s,t)dt$ where

$$k(s,t) = \begin{cases} s(t-1) & \text{if } 0 \leq s \leq t \leq 1 \\ t(s-1) & \text{if } 0 \leq t < s \leq 1 \end{cases}.$$

Note: this formulation is appropriate for the case $m \geq n$. Reverse the rules of A and A^{T} in the case of $m < n$.

9.1 Program Text

```
*     F12FBF Example Program Text
*     Mark 21 Release. NAG Copyright 2004.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. External Subroutines ..
      EXTERNAL         EX1, EX2
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F12FBF Example Program Results'
      CALL EX1
      CALL EX2
      STOP
      END
*
      SUBROUTINE EX1
*     .. Parameters ..
      INTEGER          IMON, LICOMM, NIN, NOUT
      PARAMETER        (IMON=0,LICOMM=134,NIN=5,NOUT=6)
      INTEGER          MAXN, MAXNCV, LDV
      PARAMETER        (MAXN=256,MAXNCV=30,LDV=MAXN)
      INTEGER          LCOMM
      PARAMETER        (LCOMM=3*MAXN+MAXNCV*MAXNCV+8*MAXNCV+60)
      DOUBLE PRECISION ONE, TWO, ZERO
      PARAMETER        (ONE=1.0D+0,TWO=2.0D+0,ZERO=0.0D+0)
*     .. Local Scalars ..
      DOUBLE PRECISION H2, SIGMA
      INTEGER          IFAIL, INFO, IREVCM, J, N, NCONV, NCV, NEV,
     +                 NITER, NSHIFT
*     .. Local Arrays ..
      DOUBLE PRECISION AD(MAXN), ADL(MAXN), ADU(MAXN), ADU2(MAXN),
     +                 COMM(LCOMM), D(MAXNCV,2), MX(MAXN), RESID(MAXN),
     +                 V(LDV,MAXNCV), X(MAXN)
      INTEGER          ICOMM(LICOMM), IPIV(MAXN)
*     .. External Functions ..
      DOUBLE PRECISION DNRM2
      EXTERNAL         DNRM2
*     .. External Subroutines ..
      EXTERNAL         DCOPY, DGTTRF, DGTTRS, F12FAF, F12FBF, F12FCF,
     +                 F12FDF, F12FEF
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'F12FBF Example 1'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*)
      READ (NIN,*) N, NEV, NCV
      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 F12FAF(N,NEV,NCV,ICOMM,LICOMM,COMM,LCOMM,IFAIL)
*     Set the region of the spectrum that is required.
         CALL F12FDF('LARGEST MAGNITUDE',ICOMM,COMM,IFAIL)
*     Use the Shifted Inverse mode.
         CALL F12FDF('SHIFTED INVERSE',ICOMM,COMM,IFAIL)
*
         H2 = ONE/DBLE((N+1)*(N+1))
         SIGMA = ZERO
         DO 20 J = 1, N
            AD(J) = TWO/H2 - SIGMA
            ADL(J) = -ONE/H2
   20    CONTINUE
         CALL DCOPY(N,ADL,1,ADU,1)
         CALL DGTTRF(N,ADL,AD,ADU,ADU2,IPIV,INFO)
```

```
*
         IREVCM = 0
         IFAIL = -1
   40    CONTINUE
         CALL F12FBF(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 <--- inv[A-sigma*I]*x
               CALL DGTTRS('N',N,1,ADL,AD,ADU,ADU2,IPIV,X,N,INFO)
            ELSE IF (IREVCM.EQ.4 .AND. IMON.NE.0) THEN
*           Output monitoring information
               CALL F12FEF(NITER,NCONV,D,D(1,2),ICOMM,COMM)
               WRITE (6,99998) NITER, NCONV, DNRM2(NEV,D(1,2),1)
            END IF
            GO TO 40
         END IF
         IF (IFAIL.EQ.0) THEN
*        Post-Process using F12FCF to compute eigenvalues/vectors.
            CALL F12FCF(NCONV,D,V,LDV,SIGMA,RESID,V,LDV,COMM,ICOMM,
     +                  IFAIL)
            WRITE (NOUT,99996) NCONV, SIGMA
            DO 60 J = 1, NCONV
               WRITE (NOUT,99995) J, D(J,1)
   60       CONTINUE
         ELSE
            WRITE (NOUT,99997) IFAIL
         END IF
      END IF
      RETURN
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o',
     +       'f estimates =',E16.8)
99997 FORMAT (1X,' NAG Routine F12FBF Returned with IFAIL = ',I6)
99996 FORMAT (1X,/' The ',I4,' Ritz values of closest to ',F8.4,' are:',
     +       /)
99995 FORMAT (1X,I8,5X,F12.4)
      END
*
      SUBROUTINE EX2
*     .. Parameters ..
      INTEGER          MAXM, MAXN, MAXNEV, MAXNCV, NIN, NOUT
      PARAMETER        (MAXM=500,MAXN=250,MAXNEV=10,MAXNCV=25,NIN=5,
     +                 NOUT=6)
      INTEGER          LCOMM, LDU, LDV, LICOMM
      PARAMETER        (LCOMM=3*MAXN+MAXNCV*MAXNCV+8*MAXNCV+60,
     +                  LDU=MAXM,LDV=MAXN,LICOMM=140)
*     .. Local Scalars ..
      DOUBLE PRECISION SIGMA, TEMP
      INTEGER          IFAIL, IFAIL1, IREVCM, J, M, N, NCONV, NCV, NEV,
     +                 NSHIFT
*     .. External Functions ..
      DOUBLE PRECISION DNRM2
      EXTERNAL         DNRM2
*     .. External Subroutines ..
      EXTERNAL         ATV, AV, DAXPY, DCOPY, DSCAL, F12FAF, F12FBF,
     +                 F12FCF, X04CAF
*     .. Local Arrays ..
      DOUBLE PRECISION AX(MAXM), COMM(LCOMM), D(MAXNCV,2), MX(MAXM),
     +                 RESID(MAXN), U(LDU,MAXNEV), V(LDV,MAXNCV),
     +                 X(MAXM)
      INTEGER          ICOMM(LICOMM)
*     .. Intrinsic Functions ..
      INTRINSIC        DSQRT
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'F12FBF Example 2'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NEV, NCV
```

```
      IF (N.LT.1 .OR. N.GT.MAXN) THEN
         WRITE (NOUT,99999) 'N is out of range: N = ', N
      ELSE IF (M.LT.1 .OR. M.GT.MAXM) THEN
         WRITE (NOUT,99999) 'M is out of range: M = ', M
      ELSE IF (NCV.GT.MAXNCV) THEN
         WRITE (NOUT,99999) 'NCV is out of range: NCV = ', NCV
      ELSE
*        Initialize for problem.
         IFAIL = 0
         CALL F12FAF(N,NEV,NCV,ICOMM,LICOMM,COMM,LCOMM,IFAIL)
*        Main reverse communication loop.
         IREVCM = 0
   20    CONTINUE
         CALL F12FBF(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 the operation X <- A'AX
               CALL AV(M,N,X,AX)
               CALL ATV(M,N,AX,X)
            END IF
            GO TO 20
         END IF
         IF (IFAIL.EQ.0) THEN
*           Post-Process using F12FCF.
*           Computed singular values may be extracted.
*           Singular vectors may also be computed now if desired.
            IFAIL1 = 0
            CALL F12FCF(NCONV,D,V,LDV,SIGMA,RESID,V,LDV,COMM,ICOMM,
     *                  IFAIL1)
*
*           Singular values (squared) are returned in the first column
*           of D and the corresponding right singular vectors are
*           returned in the first NEV columns of V.
*
            DO 40 J = 1, NCONV
               D(J,1) = DSQRT(D(J,1))
*
*              Compute the left singular vectors from the formula
*                     u = Av/sigma
*              u should have norm 1 so divide by norm(Av).
*
               CALL AV(M,N,V(1,J),AX)
               CALL DCOPY(M,AX,1,U(1,J),1)
               TEMP = 1.0D0/DNRM2(M,U(1,J),1)
               CALL DSCAL(M,TEMP,U(1,J),1)
*
*              Compute the residual norm ||A*v - sigma*u|| for the NCONV
*              accurately computed singular values and vectors.
*              Store the result in 2nd column of array D.
*
               CALL DAXPY(M,-D(J,1),U(1,J),1,AX,1)
               D(J,2) = DNRM2(M,AX,1)
*
   40       CONTINUE
*
*           Print computed residuals
*
            CALL X04CAF('G','N',NCONV,2,D,MAXNCV,
     +                  ' Singular values and direct residuals',IFAIL1)
         ELSE
            WRITE (NOUT,99999) IFAIL
         END IF
      END IF
      RETURN
*
99999 FORMAT (1X,' NAG Routine F12FBF Returned with IFAIL = ',I6)
      END
*
*     Matrix vector subroutines
*
      SUBROUTINE AV(M,N,X,W)
```

```
*
*     Computes  w <- A*x.
*
*     .. Parameters ..
      DOUBLE PRECISION ONE, ZERO
      PARAMETER        (ONE=1.0D+0,ZERO=0.0D+0)
*     .. Scalar Arguments ..
      INTEGER          M, N
*     .. Array Arguments ..
      DOUBLE PRECISION W(M), X(N)
*     .. Local Scalars ..
      DOUBLE PRECISION H, K, S, T
      INTEGER          I, J
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      CONTINUE
      H = ONE/DBLE(M+1)
      K = ONE/DBLE(N+1)
      DO 20 I = 1, M
         W(I) = ZERO
   20 CONTINUE
      T = ZERO
*
      DO 80 J = 1, N
         T = T + K
         S = ZERO
         DO 40 I = 1, J
            S = S + H
            W(I) = W(I) + K*S*(T-ONE)*X(J)
   40    CONTINUE
         DO 60 I = J + 1, M
            S = S + H
            W(I) = W(I) + K*T*(S-ONE)*X(J)
   60    CONTINUE
   80 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE ATV(M,N,W,Y)
*
*     Computes  y <- A'*w.
*
*     .. Parameters ..
      DOUBLE PRECISION ONE, ZERO
      PARAMETER        (ONE=1.0D+0,ZERO=0.0D+0)
*     .. Scalar Arguments ..
      INTEGER          M, N
*     .. Array Arguments ..
      DOUBLE PRECISION W(M), Y(N)
*     .. Local Scalars ..
      DOUBLE PRECISION H, K, S, T
      INTEGER          I, J
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      CONTINUE
      H = ONE/DBLE(M+1)
      K = ONE/DBLE(N+1)
      DO 20 I = 1, N
         Y(I) = ZERO
   20 CONTINUE
      T = ZERO
*
      DO 80 J = 1, N
         T = T + K
         S = ZERO
         DO 40 I = 1, J
            S = S + H
            Y(J) = Y(J) + K*S*(T-ONE)*W(I)
```

```
   40    CONTINUE
         DO 60 I = J + 1, M
            S = S + H
            Y(J) = Y(J) + K*T*(S-ONE)*W(I)
   60    CONTINUE
   80 CONTINUE
*
      RETURN
      END
```

9.2 Program Data

```
F12FBF Example Program Data
Example 1
 100 4 10 : Values for N NEV and NCV
Example 2
 500 100 4 10 : Values for M N NEV and NCV
```

9.3 Program Results

```
 F12FBF Example Program Results

 F12FBF Example 1


 The    4 Ritz values of closest to   0.0000 are:

        1           9.8688
        2          39.4657
        3          88.7620
        4         157.7101

 F12FBF Example 2

  Singular values and direct residuals
              1            2
 1    4.1012E-02   8.5939E-18
 2    6.0488E-02   2.6233E-17
 3    1.1784E-01   3.8941E-17
 4    5.5723E-01   5.0486E-17
```