NAG Fortran Library Routine Document

F02SDF

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

F02SDF finds the eigenvector corresponding to a given real eigenvalue for the generalized problem $Ax = \lambda Bx$, or for the standard problem $Ax = \lambda x$, where A and B are real band matrices.

2 Specification

```
 SUBROUTINE F02SDF(N, MA1, MB1, A, IA, B, IB, SYM, RELEP, RMU, VEC, D,
1                  IWORK, WORK, LWORK, IFAIL)
 INTEGER           N, MA1, MB1, IA, IB, IWORK(N), LWORK, IFAIL
 real              A(IA,N), B(IB,N), RELEP, RMU, VEC(N), D(30),
1                  WORK(LWORK)
 LOGICAL           SYM
```

3 Description

Given an approximation μ to a real eigenvalue λ of the generalized eigenproblem $Ax = \lambda Bx$, this routine attempts to compute the corresponding eigenvector by inverse iteration.

The routine first computes lower and upper triangular factors, L and U, of $A - \mu B$, using Gaussian elimination with interchanges, and then solves the equation $Ux = e$, where $e = (1, 1, 1, \ldots, 1)^T$ – this is the first half iteration.

There are then three possible courses of action depending on the input value of D(1).

1. D(1) = 0.

   This setting should be used if λ is an ill-conditioned eigenvalue (provided the matrix elements do not vary widely in order of magnitude). In this case it is essential to accept only a vector found after one half iteration, and μ must be a very good approximation to λ. If acceptable growth is achieved in the solution of $Ux = e$, then the normalised x is accepted as the eigenvector. If not, columns of an orthogonal matrix are tried in turn in place of e. If none of these give acceptable growth, the routine fails, indicating that μ was not a sufficiently good approximation to λ.

2. D(1) > 0.

   This setting should be used if μ is moderately close to an eigenvalue which is not ill-conditioned (provided the matrix elements do not differ widely in order of magnitude). If acceptable growth is achieved in the solution of $Ux = e$, the normalised x is accepted as the eigenvector. If not, inverse iteration is performed. Up to 30 iterations are allowed to achieve a vector and a correction to μ which together give acceptably small residuals.

3. D(1) < 0.

   This setting should be used if the elements of A and B vary widely in order of magnitude. Inverse iteration is performed, but a different convergence criterion is used.

See Section 8.3 for further details.

Note that the bandwidth of the matrix A must not be less than the bandwidth of B. If this is not so, either A must be filled out with zeros, or matrices A and B may be reversed and $1/\mu$ supplied as an approximation to the eigenvalue $1/\lambda$. Also it is assumed that A and B each have the same number of sub-diagonals as super-diagonals. If this is not so, they must be filled out with zeros. If A and B are **both** symmetric, only the upper triangles need be supplied.

4 References

Peters G and Wilkinson J H (1979) Inverse iteration, ill-conditioned equations and Newton's method *SIAM Rev.* **21** 339–360

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

Wilkinson J H (1972) Inverse iteration in theory and practice *Symposia Mathematica Volume X* 361–379 Istituto Nazionale di Alta Matematica, Monograf, Bologna

Wilkinson J H (1974) Notes on inverse iteration and ill-conditioned eigensystems *Acta Univ. Carolin. Math. Phys.* **1–2** 173–177

Wilkinson J H (1979) Kronecker's canonical form and the QZ algorithm *Linear Algebra Appl.* **28** 285–303

5 Parameters

1: N – INTEGER *Input*

On entry: n, the order of the matrices A and B.

Constraint: N $\geq$ 1.

2: MA1 – INTEGER *Input*

On entry: the value $m_A + 1$, where m_A is the number of non-zero lines on each side of the diagonal of A. Thus the total bandwidth of A is $2m_A + 1$.

Constraint: $1 \leq$ MA1 $\leq$ N.

3: MB1 – INTEGER *Input*

On entry: if MB1 ≤ 0, then B is assumed to be the unit matrix. Otherwise MB1 must specify the value $m_B + 1$, where m_B is the number of non-zero lines on each side of the diagonal of B. Thus the total bandwidth of B is $2m_B + 1$.

Constraint: MB1 $\leq$ MA1.

4: A(IA,N) – ***real*** array *Input/Output*

On entry: the n by n band matrix A. The m_A sub-diagonals must be stored in the first m_A rows of the array; the diagonal in the $(m_A + 1)$th row; and the m_A super-diagonals in rows $m_A + 2$ to $2m_A + 1$. Each row of the matrix must be stored in the corresponding column of the array. For example, if $n = 6$ and $m_A = 2$ the storage scheme is:

$$\begin{array}{cccccc}
* & * & a_{31} & a_{42} & a_{53} & a_{64} \\
* & a_{21} & a_{32} & a_{43} & a_{54} & a_{65} \\
a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\
a_{12} & a_{23} & a_{34} & a_{45} & a_{56} & * \\
a_{13} & a_{24} & a_{35} & a_{46} & * & *
\end{array}.$$

Elements of the array marked $*$ need not be set. The following code assigns the matrix elements within the band to the correct elements of the array:

```
      DO 20 J = 1, N
         DO 10 I = MAX(1,J-MA1+1), MIN(N,J+MA1-1)
            A(I-J+MA1,J) = matrix(J,I)
   10    CONTINUE
   20 CONTINUE
```

If SYM = .TRUE. (see below) (i.e., both A and B are symmetric), only the lower triangle of A need be stored in the first MA1 rows of the array.

On exit: details of the factorization of $A - \bar{\lambda}B$, where $\bar{\lambda}$ is an estimate of the eigenvalue.

5: IA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F02SDF is called.

Constraint: IA $\geq 2 \times$ MA1 $- 1$.

6: B(IB,N) – ***real*** array *Input/Output*

On entry: if MB1 > 0, then B must contain the n by n band matrix B, stored in the same way as A. If SYM = .TRUE., then only the lower triangle of B need be stored in the first MB1 rows of the array. If MB1 ≤ 0, the array is not used.

On exit: elements in the top-left corner, and in the bottom right corner if SYM = .FALSE., are set to zero; otherwise the array is unchanged.

7: IB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F02SDF is called.

Constraints:

if SYM = .FALSE., IB $\geq 2 \times$ MB1 $- 1$,
if SYM = .TRUE., IB $\geq$ MB1.

8: SYM – LOGICAL *Input*

On entry: if SYM = .TRUE., then both A and B are assumed to be symmetric and only their upper triangles need be stored. Otherwise SYM must be set to .FALSE..

9: RELEP – ***real*** *Input*

On entry: the relative error of the coefficients of the given matrices A and B. If the value of RELEP is less than the ***machine precision***, the ***machine precision*** is used instead.

10: RMU – ***real*** *Input*

On entry: μ, an approximation to the eigenvalue for which the corresponding eigenvector is required.

11: VEC(N) – ***real*** array *Output*

On exit: the eigenvector, normalised so that the largest element is unity, corresponding to the improved eigenvalue RMU + D(30).

12: D(30) – ***real*** array *Input/Output*

On entry: D(1) must be set to indicate the type of problem (see Section 3):

D$(1) > 0.0$

Indicates a well-conditioned eigenvalue.

D$(1) = 0.0$

Indicates an ill-conditioned eigenvalue.

D$(1) < 0.0$

Indicates that the matrices have elements varying widely in order of magnitude.

On exit: if D$(1) \neq 0.0$ on entry, the successive corrections to μ are given in D(i), for $i = 1, 2, \ldots, k$, where $k + 1$ is the total number of iterations performed. The final correction is also given in the last position, D(30), of the array. The remaining elements of D are set to zero. If D$(1) = 0.0$ on entry, no corrections to μ are computed and D(i) is set to 0.0, for $i = 1, 2, \ldots, 30$. Thus in all 3 cases the best available approximation to the eigenvalue is RMU + D(30).

13: IWORK(N) – INTEGER array *Workspace*

14: WORK(LWORK) – ***real*** array *Workspace*

15: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F02SDF is called.

Constraints:

if $D(1) \neq 0.0$, on entry, LWORK $\geq$ N $\times$ (MA1 + 1),
if $D(1) = 0.0$, on entry, LWORK $\geq 2 \times$ N.

16: 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

On entry, N < 1,
or MA1 < 1,
or MA1 > N,
or IA $< 2 \times$ MA1 $-$ 1,
or IB < MB1 when SYM = .TRUE.,
or IB $< 2 \times$ MB1 $-$ 1 when SYM = .FALSE. (IB is not checked if MB1 ≤ 0).

IFAIL = 2

On entry, MA1 < MB1. Either fill out A with zeros, or reverse the roles of A and B, and replace RMU by its reciprocal, i.e., solve $Bx = \lambda^{-1}Ax$.

IFAIL = 3

On entry, LWORK $< 2 \times$ N when $D(1) = 0.0$,
or LWORK < N$\times$ (MA1 + 1) when $D(1) \neq 0.0$.

IFAIL = 4

A is null. If B is non-singular, all the eigenvalues are zero and any set of N orthogonal vectors forms the eigensolution.

IFAIL = 5

B is null. If A is non-singular, all the eigenvalues are infinite, and the columns of the unit matrix are eigenvectors.

IFAIL = 6

On entry, A and B are both null. The eigensolution is arbitrary.

IFAIL = 7

D(1) $\neq$ 0.0 on entry and convergence is not achieved in 30 iterations. Either the eigenvalue is ill-conditioned or RMU is a poor approximation to the eigenvalue. See Section 8.3.

IFAIL = 8

D(1) = 0.0 on entry and no eigenvector has been found after min(N, 5) back-substitutions. RMU is not a sufficiently good approximation to the eigenvalue.

IFAIL = 9

D(1) < 0.0 on entry and RMU is too inaccurate for the solution to converge.

7 Accuracy

The eigensolution is exact for some problem

$$(A+E)x = \mu(B+F)x,$$

where $\|E\|, \|F\|$ are of the order of $\eta(\|A\| + \mu\|B\|)$, where η is the value used for RELEP.

8 Further Comments

8.1 Timing

The time taken by the routine is approximately proportional to $n(2m_A+1)^2$ for factorization, and to $n(2m_A+1)$ for each iteration.

8.2 Storage

The storage of the matrices A and B is designed for efficiency on a paged machine.

This routine will work with full matrices but it will do so inefficiently, particularly in respect of storage requirements.

8.3 Algorithmic Details

Inverse iteration is performed according to the rule

$$(A-\mu B)y_{r+1} = Bx_r$$

$$x_{r+1} = \frac{1}{\alpha_{r+1}} y_{r+1}$$

where α_{r+1} is the element of y_{r+1} of largest magnitude.

Thus:

$$(A-\mu B)x_{r+1} = \frac{1}{\alpha_{r+1}} Bx_r.$$

Hence the residual corresponding to x_{r+1} is very small if $|\alpha_{r+1}|$ is very large (see Peters and Wilkinson (1979)). The first half iteration, $Uy_1 = e$, corresponds to taking $L^{-1}PBx_0 = e$.

If μ is a very accurate eigenvalue, then there should always be an initial vector x_0 such that one half iteration gives a small residual and thus a good eigenvector. If the eigenvalue is ill-conditioned, then second and subsequent iterated vectors may not be even remotely close to an eigenvector of a neighbouring problem (see pages 374–376 of Wilkinson (1972) and Wilkinson (1974)). In this case it is essential to accept only a vector obtained after one half iteration.

However, for well-conditioned eigenvalues, there is no loss in performing more than one iteration (see page 376 of Wilkinson (1972)), and indeed it will be necessary to iterate if μ is not such a good approximation to the eigenvalue. When the iteration has converged, y_{r+1} will be some multiple of x_r, $y_{r+1} = \beta_{r+1}x_r$, say.

Therefore

$$(A - \mu B)\beta_{r+1}x_r = Bx_r,$$

giving

$$\left(A - \left(\mu + \frac{1}{\beta_{r+1}}\right)B\right)x_r = 0.$$

Thus $\mu + \frac{1}{\beta_{r+1}}$ is a better approximation to the eigenvalue. β_{r+1} is obtained as the element of y_{r+1} which corresponds to the element of largest magnitude, $+1$, in x_r. The routine terminates when $\left\|\left(A - \left(\mu + \frac{1}{\beta_r}\right)B\right)x_r\right\|$ is of the order of the ***machine precision*** relative to $\|A\| + |\mu|\|B\|$.

If the elements of A and B vary widely in order of magnitude, then $\|A\|$ and $\|B\|$ are excessively large and a different convergence test is required. The routine terminates when the difference between successive corrections to μ is small relative to μ.

In practice one does not necessarily know if the given problem is well-conditioned or ill-conditioned. In order to provide some information on the condition of the eigenvalue or the accuracy of μ in the event of failure, successive values of $\frac{1}{\beta_r}$ are stored in the vector D when D(1) is non-zero on input. If these values appear to be converging steadily, then it is likely that μ was a poor approximation to the eigenvalue and it is worth trying again with RMU + D(30) as the initial approximation. If the values in D vary considerably in magnitude, then the eigenvalue is ill-conditioned.

A discussion of the significance of the singularity of A and/or B is given in relation to the QZ algorithm in Wilkinson (1979).

9 Example

Given the generalized eigenproblem $Ax = \lambda Bx$ where

$$A = \begin{pmatrix} 1 & 1 & 2 & & \\ -1 & 2 & 1 & 2 & \\ & -1 & 3 & 1 & 2 \\ & & -1 & 4 & 1 \\ & & & -1 & 5 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 5 & 1 & & & \\ 1 & 4 & 2 & & \\ & 2 & 3 & 2 & \\ & & 2 & 2 & 1 \\ & & & 1 & 1 \end{pmatrix}$$

find the eigenvector corresponding to the approximate eigenvalue -12.33.

Although B is symmetric, A is not, so SYM must be set to .FALSE. and all the elements of B in the band must be supplied to the routine. A (as written above) has 1 sub-diagonal and 2 super-diagonals, so MA1 must be set to 3 and A filled out with an additional sub-diagonal of zeros. Each row of the matrices is read in as data in turn.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F02SDF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX, MAMAX, MBMAX, IA, IB, LWORK
      PARAMETER        (NMAX=10,MAMAX=5,MBMAX=5,IA=2*MAMAX+1,
     +                 IB=2*MBMAX+1,LWORK=NMAX*(MAMAX+2))
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             RMU
      INTEGER          I, IFAIL, J, K, K1, K2, MA, MB, N
*     .. Local Arrays ..
      real             A(IA,NMAX), B(IB,NMAX), D(30), WORK(LWORK),
     +                 X(NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F02SDF
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F02SDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, MA, MB
      IF (N.GT.0 .AND. N.LE.NMAX .AND. MA.GE.0 .AND. MA.LE.MAMAX .AND.
     +    MB.LE.MBMAX) THEN
         DO 20 I = 1, N
            K1 = MA + 1 - MIN(MA,I-1)
            K2 = MA + 1 + MIN(MA,N-I)
            READ (NIN,*) (A(K,I),K=K1,K2)
   20    CONTINUE
         DO 40 I = 1, N
            K1 = MB + 1 - MIN(MB,I-1)
            K2 = MB + 1 + MIN(MB,N-I)
            READ (NIN,*) (B(K,I),K=K1,K2)
   40    CONTINUE
         READ (NIN,*) RMU, D(1)
         IFAIL = 1
*
         CALL F02SDF(N,MA+1,MB+1,A,IA,B,IB,.FALSE.,0.0e0,RMU,X,D,IWORK,
     +               WORK,LWORK,IFAIL)
*
         WRITE (NOUT,*)
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999) 'Error in F02SDF. IFAIL =', IFAIL
            IF (IFAIL.EQ.7 .OR. IFAIL.EQ.9) THEN
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Successive corrections to RMU were'
               WRITE (NOUT,*)
               DO 60 J = 1, 29
                  IF (D(J).EQ.0.0e0) STOP
                  WRITE (NOUT,99996) D(J)
   60          CONTINUE
            END IF
         ELSE
            WRITE (NOUT,99998) 'Corrected eigenvalue = ', RMU + D(30)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Eigenvector is'
            WRITE (NOUT,99997) (X(I),I=1,N)
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,I2)
99998 FORMAT (1X,A,F8.4)
99997 FORMAT (1X,5F9.4)
99996 FORMAT (1X,e20.4)
      END
```

9.2 Program Data

```
F02SDF Example Program Data
  5  2  1
  1.0  1.0  2.0
 -1.0  2.0  1.0  2.0
  0.0 -1.0  3.0  1.0  2.0
  0.0 -1.0  4.0  1.0
  0.0 -1.0  5.0
  5.0  1.0
  1.0  4.0  2.0
  2.0  3.0  2.0
  2.0  2.0  1.0
  1.0  1.0
      -12.33 1.0
```

9.3 Program Results

```
 F02SDF Example Program Results

 Corrected eigenvalue = -12.3394

 Eigenvector is
   -0.0572   0.3951  -0.8427   1.0000  -0.6540
```