

NAG Library Routine Document

F01BVF

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

F01BVF transforms the generalized symmetric-definite eigenproblem $Ax = \lambda Bx$ to the equivalent standard eigenproblem $Cy = \lambda y$, where A , B and C are symmetric band matrices and B is positive definite. B must have been decomposed by F01BUF.

2 Specification

```
SUBROUTINE F01BVF (N, MA1, MB1, M3, K, A, LDA, B, LDB, V, LDV, W, IFAIL)
  INTEGER          N, MA1, MB1, M3, K, LDA, LDB, LDV, IFAIL
  REAL (KIND=nag_wp) A(LDA,N), B(LDB,N), V(LDV,M3), W(M3)
```

3 Description

A is a symmetric band matrix of order n and bandwidth $2m_A + 1$. The positive definite symmetric band matrix B , of order n and bandwidth $2m_B + 1$, must have been previously decomposed by F01BUF as $ULDL^T U^T$. F01BVF applies U , L and D to A , m_A rows at a time, restoring the band form of A at each stage by plane rotations. The argument k defines the change-over point in the decomposition of B as used by F01BUF and is also used as a change-over point in the transformations applied by this routine. For maximum efficiency, k should be chosen to be the multiple of m_A nearest to $n/2$. The resulting symmetric band matrix C is overwritten on A . The eigenvalues of C , and thus of the original problem, may be found using F08HEF (DSBTRD) and F08JFF (DSTERF). For selected eigenvalues, use F08HEF (DSBTRD) and F08JFF (DSTEBZ).

4 References

Crawford C R (1973) Reduction of a band-symmetric generalized eigenvalue problem *Comm. ACM* **16** 41–44

5 Arguments

- | | | |
|----|--|--------------|
| 1: | N – INTEGER | <i>Input</i> |
| | <i>On entry:</i> n , the order of the matrices A , B and C . | |
| 2: | MA1 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> $m_A + 1$, where m_A is the number of nonzero superdiagonals in A . Normally $MA1 \ll N$. | |
| 3: | MB1 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> $m_B + 1$, where m_B is the number of nonzero superdiagonals in B . | |
| | <i>Constraint:</i> $MB1 \leq MA1$. | |
| 4: | M3 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> the value of $3m_A + m_B$. | |

- 5: K – INTEGER *Input*
On entry: k , the change-over point in the transformations. It must be the same as the value used by F01BUF in the decomposition of B .
Suggested value: the optimum value is the multiple of m_A nearest to $n/2$.
Constraint: $MB1 - 1 \leq K \leq N$.
- 6: A(LDA,N) – REAL (KIND=nag_wp) array *Input/Output*
On entry: the upper triangle of the n by n symmetric band matrix A , with the diagonal of the matrix stored in the $(m_A + 1)$ th row of the array, and the m_A superdiagonals within the band stored in the first m_A rows of the array. Each column of the matrix is stored in the corresponding column of the array. For example, if $n = 6$ and $m_A = 2$, the storage scheme is
- | | | | | | |
|----------|----------|----------|----------|----------|----------|
| * | * | a_{13} | a_{24} | a_{35} | a_{46} |
| * | a_{12} | a_{23} | a_{34} | a_{45} | a_{56} |
| a_{11} | a_{22} | a_{33} | a_{44} | a_{55} | a_{66} |
- Elements in the top left corner of the array 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), J
 A(I-J+MA1,J) = matrix (I,J)
 10 CONTINUE
 20 CONTINUE

```
- On exit:* is overwritten by the corresponding elements of  $C$ .
- 7: LDA – INTEGER *Input*  
*On entry:* the first dimension of the array  $A$  as declared in the (sub)program from which F01BVF is called.  
*Constraint:*  $LDA \geq MA1$ .
- 8: B(LDB,N) – REAL (KIND=nag\_wp) array *Input/Output*  
*On entry:* the elements of the decomposition of matrix  $B$  as returned by F01BUF.  
*On exit:* the elements of  $B$  will have been permuted.
- 9: LDB – INTEGER *Input*  
*On entry:* the first dimension of the array  $B$  as declared in the (sub)program from which F01BVF is called.  
*Constraint:*  $LDB \geq MB1$ .
- 10: V(LDV,M3) – REAL (KIND=nag\_wp) array *Workspace*  
 11: LDV – INTEGER *Input*  
*On entry:* the first dimension of the array  $V$  as declared in the (sub)program from which F01BVF is called.  
*Constraint:*  $LDV \geq m_A + m_B$ .
- 12: W(M3) – REAL (KIND=nag\_wp) array *Workspace*
- 13: IFAIL – INTEGER *Input/Output*  
*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.  
 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 argument, the recommended value is 0. **When the value  $-1$  or  $1$  is used it is essential to test the value of IFAIL on exit.**

*On exit:* IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

## 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, MB1 > MA1.

IFAIL =  $-99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL =  $-399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL =  $-999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

In general the computed system is exactly congruent to a problem  $(A + E)x = \lambda(B + F)x$ , where  $\|E\|$  and  $\|F\|$  are of the order of  $\epsilon\kappa(B)\|A\|$  and  $\epsilon\kappa(B)\|B\|$  respectively, where  $\kappa(B)$  is the condition number of  $B$  with respect to inversion and  $\epsilon$  is the *machine precision*. This means that when  $B$  is positive definite but not well-conditioned with respect to inversion, the method, which effectively involves the inversion of  $B$ , may lead to a severe loss of accuracy in well-conditioned eigenvalues.

## 8 Parallelism and Performance

F01BVF is not threaded in any implementation.

## 9 Further Comments

The time taken by F01BVF is approximately proportional to  $n^2m_B^2$  and the distance of  $k$  from  $n/2$ , e.g.,  $k = n/4$  and  $k = 3n/4$  take 502% longer.

When  $B$  is positive definite and well-conditioned with respect to inversion, the generalized symmetric eigenproblem can be reduced to the standard symmetric problem  $Py = \lambda y$  where  $P = L^{-1}AL^{-T}$  and  $B = LL^T$ , the Cholesky factorization.

When  $A$  and  $B$  are of band form, especially if the bandwidth is small compared with the order of the matrices, storage considerations may rule out the possibility of working with  $P$  since it will be a full matrix in general. However, for any factorization of the form  $B = SS^T$ , the generalized symmetric problem reduces to the standard form

$$S^{-1}AS^{-T}(S^Tx) = \lambda(S^Tx)$$

and there does exist a factorization such that  $S^{-1}AS^{-T}$  is still of band form (see Crawford (1973)). Writing

$$C = S^{-1}AS^{-T} \quad \text{and} \quad y = S^T x$$

the standard form is  $Cy = \lambda y$  and the bandwidth of  $C$  is the maximum bandwidth of  $A$  and  $B$ .

Each stage in the transformation consists of two phases. The first reduces a leading principal sub-matrix of  $B$  to the identity matrix and this introduces nonzero elements outside the band of  $A$ . In the second, further transformations are applied which leave the reduced part of  $B$  unaltered and drive the extra elements upwards and off the top left corner of  $A$ . Alternatively,  $B$  may be reduced to the identity matrix starting at the bottom right-hand corner and the extra elements introduced in  $A$  can be driven downwards.

The advantage of the  $ULDL^T U^T$  decomposition of  $B$  is that no extra elements have to be pushed over the whole length of  $A$ . If  $k$  is taken as approximately  $n/2$ , the shifting is limited to halfway. At each stage the size of the triangular bumps produced in  $A$  depends on the number of rows and columns of  $B$  which are eliminated in the first phase and on the bandwidth of  $B$ . The number of rows and columns over which these triangles are moved at each step in the second phase is equal to the bandwidth of  $A$ .

In this routine,  $A$  is defined as being at least as wide as  $B$  and must be filled out with zeros if necessary as it is overwritten with  $C$ . The number of rows and columns of  $B$  which are effectively eliminated at each stage is  $m_A$ .

## 10 Example

This example finds the three smallest eigenvalues of  $Ax = \lambda Bx$ , where

$$A = \begin{pmatrix} 11 & 12 & & & & & & & & & \\ & 12 & 13 & & & & & & & & \\ & & 13 & 14 & & & & & & & \\ & & & 14 & 15 & & & & & & \\ & & & & 15 & 16 & & & & & \\ & & & & & 16 & 17 & & & & \\ & & & & & & 17 & 18 & & & \\ & & & & & & & 18 & 19 & & \\ & & & & & & & & 19 & 19 & \end{pmatrix}$$

$$B = \begin{pmatrix} 101 & 22 & & & & & & & & & \\ & 22 & 102 & 23 & & & & & & & \\ & & 23 & 103 & 24 & & & & & & \\ & & & 24 & 104 & 25 & & & & & \\ & & & & 25 & 105 & 26 & & & & \\ & & & & & 26 & 106 & 27 & & & \\ & & & & & & 27 & 107 & 28 & & \\ & & & & & & & 28 & 108 & 29 & \\ & & & & & & & & 29 & 109 & \end{pmatrix}.$$

### 10.1 Program Text

```

Program f01bvfe

! F01BVF Example Program Text

! Mark 26 Release. NAG Copyright 2016.

! .. Use Statements ..
 Use nag_library, Only: dsbtrd, dstebz, f01buf, f01bvf, nag_wp
! .. Implicit None Statement ..
 Implicit None
! .. Parameters ..
 Real (Kind=nag_wp), Parameter :: zero = 0.0_nag_wp
 Integer, Parameter :: incl = 1, nin = 5, nout = 6
! .. Local Scalars ..

```

```

 Real (Kind=nag_wp) :: abstol
 Integer :: i, ifail, info, j, k, lda, ldb, ldv, &
 m, m1, m2, m3, ma1, mb1, n, nsplit
! .. Local Arrays ..
 Real (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), d(:), e(:), r(:), &
 v(:,:), w(:), work(:)
 Integer, Allocatable :: iblock(:), isplit(:), iwork(:)
! .. Intrinsic Procedures ..
 Intrinsic :: max
! .. Executable Statements ..
 Write (nout,*) 'F01BVF Example Program Results'
! Skip heading in data file
 Read (nin,*)
 Read (nin,*) n, ma1, mb1
 lda = ma1
 ldb = mb1
 ldv = ma1 + mb1 - 2
 m3 = 3*ma1 + mb1 - 4
 Allocate (a(lda,n),b(ldb,n),d(n),e(n),r(n),v(ldv,m3),w(m3),work(4*n), &
 iblock(n),isplit(n),iwork(3*n))
 Read (nin,*)((a(j,i),j=max(1,ma1+1-i),ma1),i=1,n)
 Read (nin,*)((b(j,i),j=max(1,mb1+1-i),mb1),i=1,n)
 k = n/2

! ifail: behaviour on error exit
! =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
 ifail = 0
 Call f01buf(n,mb1,k,b,ldb,w,ifail)

 ifail = 0
 Call f01bvf(n,ma1,mb1,m3,k,a,lda,b,ldb,v,ldv,w,ifail)

! The NAG name equivalent of dsbtrd is f08hef
 Call dsbtrd('N','U',n,ma1-1,a,lda,d,e,w,inc1,work,info)

 abstol = zero
 Read (nin,*) m1, m2

! The NAG name equivalent of dstebz is f08jjf
 Call dstebz('I','E',n,zero,zero,m1,m2,abstol,d,e,m,nsplit,r,iblock, &
 isplit,work,iwork,info)

 Write (nout,*)
 Write (nout,*) 'Selected eigenvalues'
 Write (nout,99999) r(1:m)

99999 Format (1X,8F9.4)
 End Program f01bvfe

```

## 10.2 Program Data

F01BVF Example Program Data

```

 9 2 2 : n, ma1, mb1
11
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19 : a
101
22 102
23 103
24 104
25 105

```

```
26 106
27 107
28 108
29 109 : b
1 3 : m1, m2
```

### 10.3 Program Results

F01BVF Example Program Results

Selected eigenvalues  
-0.2643 -0.1530 -0.0418

---