

# NAG Fortran Library Routine Document

## F02HDF

**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

F02HDF computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian-definite generalized eigenproblem.

### 2 Specification

```

SUBROUTINE F02HDF( ITYPE, JOB, UPLO, N, A, LDA, B, LDB, W, RWORK, WORK,
1                LWORK, IFAIL)
  INTEGER          ITYPE, N, LDA, LDB, LWORK, IFAIL
  real           W(*), RWORK(*)
  complex       A(LDA,*), B(LDB,*), WORK(LWORK)
  CHARACTER*1     JOB, UPLO

```

### 3 Description

This routine computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian-definite generalized eigenproblem of one of the following types:

1.  $Az = \lambda Bz$
2.  $ABz = \lambda z$
3.  $BAz = \lambda z$

Here  $A$  and  $B$  are Hermitian, and  $B$  must be positive-definite.

The method involves implicitly inverting  $B$ ; hence if  $B$  is ill-conditioned with respect to inversion, the results may be inaccurate (see Section 7).

Note that the matrix  $Z$  of eigenvectors is not unitary, but satisfies the following relationships for the three types of problem above:

1.  $Z^H BZ = I$
2.  $Z^H BZ = I$
3.  $Z^H B^{-1} Z = I$

### 4 References

Golub G H and van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Parlett B N (1980) *The Symmetric Eigenvalue Problem* Prentice-Hall

### 5 Parameters

1: ITYPE – INTEGER *Input*

*On entry:* indicates the type of problem, as follows:

- if ITYPE = 1, the problem is  $Az = \lambda Bz$ ;
- if ITYPE = 2, the problem is  $ABz = \lambda z$ ;

if  $ITYPE = 3$ , the problem is  $BAz = \lambda z$ .

*Constraint:*  $ITYPE = 1, 2$  or  $3$ .

- 2: JOB – CHARACTER\*1 *Input*  
*On entry:* indicates whether eigenvectors are to be computed as follows:  
 if JOB = 'N', then only eigenvalues are computed;  
 if JOB = 'V', then eigenvalues and eigenvectors are computed.  
*Constraint:* JOB = 'N' or 'V'.
- 3: UPLO – CHARACTER\*1 *Input*  
*On entry:* indicates whether the upper or lower triangular parts of  $A$  and  $B$  are stored as follows:  
 if UPLO = 'U', then the upper triangular parts of  $A$  and  $B$  are stored;  
 if UPLO = 'L', then the lower triangular parts of  $A$  and  $B$  are stored.  
*Constraint:* UPLO = 'U' or 'L'.
- 4: N – INTEGER *Input*  
*On entry:*  $n$ , the order of the matrices  $A$  and  $B$ .  
*Constraint:*  $N \geq 0$ .
- 5: A(LDA,\*) – **complex** array *Input/Output*  
**Note:** the second dimension of the array  $A$  must be at least  $\max(1, N)$ .  
*On entry:* the  $n$  by  $n$  Hermitian matrix  $A$ . If UPLO = 'U', the upper triangle of  $A$  must be stored and the elements of the array below the diagonal need not be set; if UPLO = 'L', the lower triangle of  $A$  must be stored and the elements of the array above the diagonal need not be set.  
*On exit:* If JOB = 'V',  $A$  contains the matrix  $Z$  of eigenvectors, with the  $i$ th column holding the eigenvector  $z_i$  associated with the eigenvalue  $\lambda_i$  (stored in  $W(i)$ ). If JOB = 'N', the original contents of  $A$  are overwritten.
- 6: LDA – INTEGER *Input*  
*On entry:* the first dimension of the array  $A$  as declared in the (sub)program from which F02HDF is called.  
*Constraint:*  $LDA \geq \max(1, N)$ .
- 7: B(LDB,\*) – **complex** array *Input/Output*  
**Note:** the second dimension of the array  $B$  must be at least  $\max(1, N)$ .  
*On entry:* the  $n$  by  $n$  Hermitian positive-definite matrix  $B$ . If UPLO = 'U', the upper triangle of  $B$  must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of  $B$  must be stored and the elements of the array above the diagonal are not referenced.  
*On exit:* the upper or lower triangle of  $B$  (as specified by UPLO) is overwritten by the triangular factor  $U$  or  $L$  from the Cholesky factorization of  $B$  as  $U^H U$  or  $LL^H$ .
- 8: LDB – INTEGER *Input*  
*On entry:* the first dimension of the array  $B$  as declared in the (sub)program from which F02HDF is called.  
*Constraint:*  $LDB \geq \max(1, N)$ .

- 9:  $W(*)$  – *real* array *Output*  
**Note:** the dimension of the array  $W$  must be at least  $\max(1, N)$ .  
*On exit:* the eigenvalues in ascending order.
- 10:  $RWORK(*)$  – *real* array *Workspace*  
**Note:** the dimension of the array  $RWORK$  must be at least  $\max(1, 3 \times N)$ .
- 11:  $WORK(LWORK)$  – *complex* array *Workspace*  
 12:  $LWORK$  – INTEGER *Input*  
*On entry:* the dimension of the array  $WORK$  as declared in the (sub)program from which F02HDF is called. On some high-performance computers, increasing the dimension of  $WORK$  will enable the routine to run faster; a value of  $64 \times N$  should allow near-optimal performance on almost all machines.  
**Constraint:**  $LWORK \geq \max(1, 2 \times N)$ .
- 13:  $IFAIL$  – 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,  $ITYPE \neq 1, 2$  or 3,  
 or  $JOB \neq 'N'$  or  $'V'$ ,  
 or  $UPLO \neq 'U'$  or  $'L'$ ,  
 or  $N < 0$ ,  
 or  $LDA < \max(1, N)$ ,  
 or  $LDB < \max(1, N)$ ,  
 or  $LWORK < \max(1, 2 \times N)$ .

$IFAIL = 2$

The  $QR$  algorithm failed to compute all the eigenvalues.

$IFAIL = 3$

The matrix  $B$  is not positive-definite.

$IFAIL = 4$

For some  $i$ ,  $A(i, i)$  has a non-zero imaginary part (thus  $A$  is not Hermitian).

$IFAIL = 5$

For some  $i$ ,  $B(i, i)$  has a non-zero imaginary part (thus  $B$  is not Hermitian).

## 7 Accuracy

If  $\lambda_i$  is an exact eigenvalue, and  $\tilde{\lambda}_i$  is the corresponding computed value, then for problems of the form  $Az = \lambda Bz$ ,

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\epsilon\|A\|_2\|B^{-1}\|_2;$$

and for problems of the form  $ABz = \lambda z$  or  $BAz = \lambda z$ ,

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\epsilon\|A\|_2\|B\|_2.$$

Here  $c(n)$  is a modestly increasing function of  $n$ , and  $\epsilon$  is the *machine precision*.

If  $z_i$  is the corresponding exact eigenvector, and  $\tilde{z}_i$  is the corresponding computed eigenvector, then the angle  $\theta(\tilde{z}_i, z_i)$  between them is bounded as follows:

for problems of the form  $Az = \lambda Bz$ ,

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\|A\|_2\|B^{-1}\|_2(\kappa_2(B))^{1/2}}{\min_{i \neq j} |\lambda_i - \lambda_j|};$$

and for problems of the form  $ABz = \lambda z$  or  $BAz = \lambda z$ ,

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\|A\|_2\|B\|_2(\kappa_2(B))^{1/2}}{\min_{i \neq j} |\lambda_i - \lambda_j|}.$$

Here  $\kappa_2(B)$  is the condition number of  $B$  with respect to inversion defined by:  $\kappa_2(B) = \|B\|_2\|B^{-1}\|_2$ . Thus the accuracy of a computed eigenvector depends on the gap between its eigenvalue and all the other eigenvalues, and also on the condition of  $B$ .

## 8 Further Comments

The routine calls routines from LAPACK in Chapter F08. It first reduces the problem to an equivalent standard eigenproblem  $Cy = \lambda y$ . It then reduces  $C$  to real tridiagonal form  $T$ , using a unitary similarity transformation:  $C = QTQ^H$ . To compute eigenvalues only, the routine uses a root-free variant of the symmetric tridiagonal  $QR$  algorithm to reduce  $T$  to a diagonal matrix  $A$ . If eigenvectors are required, the routine first forms the unitary matrix  $Q$  that was used in the reduction to tridiagonal form; it then uses the symmetric tridiagonal  $QR$  algorithm to reduce  $T$  to  $A$ , using a real orthogonal transformation:  $T = SAS^T$ ; and at the same time accumulates the matrix  $Y = QS$ , which is the matrix of eigenvectors of  $C$ . Finally it transforms the eigenvectors of  $C$  back to those of the original generalized problem.

Each eigenvector  $z$  is normalized so that:

for problems of the form  $Az = \lambda Bz$  or  $ABz = \lambda z$ ,  $z^H Bz = 1$ ;

for problems of the form  $BAz = \lambda z$ ,  $z^H B^{-1}z = 1$ .

The time taken by the routine is approximately proportional to  $n^3$ .

## 9 Example

To compute all the eigenvalues and eigenvectors of the problem  $Az = \lambda Bz$ , where

$$A = \begin{pmatrix} -7.36 + 0.00i & 0.77 - 0.43i & -0.64 - 0.92i & 3.01 - 6.97i \\ 0.77 + 0.43i & 3.49 + 0.00i & 2.19 + 4.45i & 1.90 + 3.73i \\ -0.64 + 0.92i & 2.19 - 4.45i & 0.12 + 0.00i & 2.88 - 3.17i \\ 3.01 + 6.97i & 1.90 - 3.73i & 2.88 + 3.17i & -2.54 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

## 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.

```
*      F02HDF Example Program Text
*      Mark 16 Release. NAG Copyright 1992.
*      .. Parameters ..
INTEGER          NIN, NOUT
PARAMETER       (NIN=5,NOUT=6)
INTEGER          NMAX, LDA, LDB, LWORK
PARAMETER       (NMAX=8,LDA=NMAX,LDB=NMAX,LWORK=64*NMAX)
*      .. Local Scalars ..
INTEGER          I, IFAIL, ITYPE, J, N
CHARACTER        UPLO
*      .. Local Arrays ..
complex        A(LDA,NMAX), B(LDB,NMAX), WORK(LWORK)
real          RWORK(3*NMAX), W(NMAX)
CHARACTER        CLABS(1), RLABS(1)
*      .. External Subroutines ..
EXTERNAL         F02HDF, X04DBF
*      .. Executable Statements ..
WRITE (NOUT,*) 'F02HDF Example Program Results'
*      Skip heading in data file
READ (NIN,*)
READ (NIN,*) N
IF (N.LE.NMAX) THEN
*
*      Read A and B from data file
*
      READ (NIN,*) UPLO
      IF (UPLO.EQ.'U') THEN
        READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
        READ (NIN,*) ((B(I,J),J=I,N),I=1,N)
      ELSE IF (UPLO.EQ.'L') THEN
        READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
        READ (NIN,*) ((B(I,J),J=1,I),I=1,N)
      END IF
*
*      Compute eigenvalues and eigenvectors
*
      ITYPE = 1
      IFAIL = 0
*
      CALL F02HDF(ITYPE,'Vectors',UPLO,N,A,LDA,B,LDB,W,RWORK,WORK,
+              LWORK,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Eigenvalues'
      WRITE (NOUT,99999) (W(I),I=1,N)
      WRITE (NOUT,*)
*
      CALL X04DBF('General',' ',N,N,A,LDA,'Bracketed','F7.4',
+              'Eigenvectors','Integer',RLABS,'Integer',CLABS,80,
+              0,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT (3X,4(F12.4,6X))
      END
```

## 9.2 Program Data

F02HDF Example Program Data

```

4                                     :Value of N
'L'                                   :Value of UPLO
(-7.36, 0.00)
( 0.77, 0.43) ( 3.49, 0.00)
(-0.64, 0.92) ( 2.19,-4.45) ( 0.12, 0.00)
( 3.01, 6.97) ( 1.90,-3.73) ( 2.88, 3.17) (-2.54, 0.00) :End of matrix A
( 3.23, 0.00)
( 1.51, 1.92) ( 3.58, 0.00)
( 1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
( 0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix B

```

## 9.3 Program Results

F02HDF Example Program Results

Eigenvalues

```

-5.9990          -2.9936          0.5047          3.9990

```

Eigenvectors

```

          1          2          3          4
1 ( 1.7372, 0.1062) ( 0.4889,-0.5010) ( 0.6164, 0.1937) ( 0.2310,-1.2161)
2 (-0.3843,-0.4933) ( 0.1118,-0.0367) ( 0.2596,-0.4203) (-0.4710, 0.4814)
3 (-0.8237,-0.2991) (-0.8115, 0.4114) (-0.0365,-0.3321) (-0.2242, 0.6335)
4 ( 0.2643, 0.6276) ( 0.7877, 0.2002) ( 0.0994, 0.6588) ( 0.8515, 0.0000)

```

---