NAG Toolbox for MATLAB

f02hd

1 Purpose

f02hd computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian-definite generalized eigenproblem.

2 Syntax

```
[a, b, w, ifail] = f02hd(itype, job, uplo, a, b, 'n', n)
```

3 Description

f02hd 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. $Z^{\mathrm{H}}BZ = I$
2. $Z^{\mathrm{H}}BZ = I$
3. $Z^{\mathrm{H}}B^{-1}Z = I$

4 References

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

Parlett B N 1998 *The Symmetric Eigenvalue Problem* SIAM, Philadelphia

5 Parameters

5.1 Compulsory Input Parameters

1: **itype – int32 scalar**

Indicates the type of problem.

itype $= 1$

The problem is $Az = \lambda Bz$;

itype $= 2$

The problem is $ABz = \lambda z$;

itype $= 3$

The problem is $BAz = \lambda z$.

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

2: **job – string**

Indicates whether eigenvectors are to be computed.

job = 'N'

Only eigenvalues are computed.

job = 'V'

Eigenvalues and eigenvectors are computed.

Constraint: **job** = 'N' or 'V'.

3: **uplo – string**

Indicates whether the upper or lower triangular parts of A and B are stored.

uplo = 'U'

The upper triangular parts of A and B are stored.

uplo = 'L'

The lower triangular parts of A and B are stored.

Constraint: **uplo** = 'U' or 'L'.

4: **a(lda,*) – complex array**

The first dimension of the array **a** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

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.

5: **b(ldb,*) – complex array**

The first dimension of the array **b** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

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.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The second dimension of the array **a** The second dimension of the array **b**.

n, the order of the matrices A and B.

Constraint: $\mathbf{n} \geq 0$.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, rwork, work, lwork

5.4 Output Parameters

1: **a(lda,$*$) – complex array**

The first dimension of the array **a** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

If **job** = 'V', **a** contains the matrix Z of eigenvectors, with the ith column holding the eigenvector z_i associated with the eigenvalue λ_i (stored in $\mathbf{w}(i)$).

If **job** = 'N', the original contents of **a** are overwritten.

2: **b(ldb,$*$) – complex array**

The first dimension of the array **b** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

The upper or lower triangle of B (as specified by **uplo**) contains the triangular factor U or L from the Cholesky factorization of B as $U^{\mathrm{H}}U$ or LL^{H}.

3: **w($*$) – double array**

Note: the dimension of the array **w** must be at least $\max(1, \mathbf{n})$.

The eigenvalues in ascending order.

4: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **itype** $\neq$ 1, 2 or 3,
or **job** $\neq$ 'N' or 'V',
or **uplo** $\neq$ 'U' or 'L',
or $\mathbf{n} < 0$,
or $\mathbf{lda} < \max(1, \mathbf{n})$,
or $\mathbf{ldb} < \max(1, \mathbf{n})$,
or $\mathbf{lwork} < \max(1, 2 \times \mathbf{n})$.

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, $\mathbf{a}(i,i)$ has a nonzero imaginary part (thus A is not Hermitian).

ifail $= 5$

For some i, $\mathbf{b}(i,i)$ has a nonzero imaginary part (thus B is not Hermitian).

7 Accuracy

If λ_i is an exact eigenvalue, and $\tilde{\lambda}_i$ is the corresponding computed value, then

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

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

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

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

Here $c(n)$ is a modestly increasing function of n, and ϵ 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 $Az = \lambda Bz$,

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

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

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

Here $\kappa_2(B)$ is the condition number of B with respect to inversion defined by: $\kappa_2(B) = \|B\|.\left\|B^{-1}\right\|$. 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

f02hd calls functions 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: $C = QTQ^{\mathrm{H}}$. To compute eigenvalues only, the function uses a root-free variant of the symmetric tridiagonal QR algorithm to reduce T to a diagonal matrix Λ. If eigenvectors are required, the function 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 Λ, using a real orthogonal transformation: $T = S\Lambda S^{\mathrm{T}}$; 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 $ABz = \lambda z$, $z^{\mathrm{H}}Bz = 1$;

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

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

9 Example

```
itype = int32(1);
job = 'Vectors';
uplo = 'L';
a = [complex(-7.36, 0), complex(0, 0), complex(0, 0), complex(0, 0);
     complex(0.77, 0.43), complex(3.49, 0), complex(0, 0), complex(0, 0);
         complex(-0.64,  0.92),  complex(2.19,  -4.45),  complex(0.12,  0),
complex(0, 0);
        complex(3.01,  +6.97),  complex(1.9,  -3.73),  complex(2.88,  +3.17),
complex(-2.54, +0)];
b = [complex(3.23, +0), complex(0, 0), complex(0, 0), complex(0, 0);
       complex(1.51,  +1.92),  complex(3.58,  0),  complex(0,  0),  complex(0,
0);
         complex(1.9,  -0.84),  complex(-0.23,  -1.11),  complex(4.09,  0),
complex(0, 0);
       complex(0.42,  -2.5),  complex(-1.18,  -1.37),  complex(2.33,  +0.14),
complex(4.29, +0)];
[aOut, bOut, w, ifail] = f02hd(itype, job, uplo, a, b)
```

```
aOut =
    1.7372 + 0.1062i    0.6876 - 0.1311i    0.0202 - 0.6459i    1.0300 +
0.6865i
   -0.3843 - 0.4933i    0.1127 + 0.0339i   -0.4747 - 0.1365i   -0.2598 -
0.6213i
   -0.8237 - 0.2991i   -0.9009 - 0.1270i   -0.3099 + 0.1248i   -0.4961 -
0.4533i
    0.2643 + 0.6276i    0.5314 + 0.6150i    0.6075 - 0.2735i   -0.3318 +
0.7843i
bOut =
   1.7972                  0                  0                  0
   0.8402 + 1.0683i   1.3164                  0                  0
   1.0572 - 0.4674i  -0.4702 + 0.3131i   1.5604                  0
   0.2337 - 1.3910i   0.0834 + 0.0368i   0.9360 + 0.9900i   0.6603
w =
   -5.9990
   -2.9936
    0.5047
    3.9990
ifail =
           0
```