

NAG Toolbox for MATLAB

f08sq

1 Purpose

f08sq computes all the eigenvalues and, optionally, the eigenvectors of a complex generalized Hermitian-definite eigenproblem, of the form

$$Az = \lambda Bz, \quad ABz = \lambda z \quad \text{or} \quad BAz = \lambda z,$$

where A and B are Hermitian and B is also positive-definite. If eigenvectors are desired, it uses a divide-and-conquer algorithm.

2 Syntax

```
[a, b, w, info] = f08sq(itype, jobz, uplo, a, b, 'n', n)
```

3 Description

f08sq first performs a Cholesky factorization of the matrix B as $B = U^H U$, when **uplo** = 'U' or $B = LL^H$, when **uplo** = 'L'. The generalized problem is then reduced to a standard symmetric eigenvalue problem

$$Cx = \lambda x,$$

which is solved for the eigenvalues and, optionally, the eigenvectors; the eigenvectors are then backtransformed to give the eigenvectors of the original problem.

For the problem $Az = \lambda Bz$, the eigenvectors are normalized so that the matrix of eigenvectors, z , satisfies

$$Z^H A Z = \Lambda \quad \text{and} \quad Z^H B Z = I,$$

where Λ is the diagonal matrix whose diagonal elements are the eigenvalues. For the problem $ABz = \lambda z$ we correspondingly have

$$Z^{-1} A Z^{-H} = \Lambda \quad \text{and} \quad Z^H B Z = I,$$

and for $BAz = \lambda z$ we have

$$Z^H A Z = \Lambda \quad \text{and} \quad Z^H B^{-1} Z = I.$$

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D 1999 *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia URL: <http://www.netlib.org/lapack/lug>

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

5 Parameters

5.1 Compulsory Input Parameters

1: **itype** – **int32** scalar

Specifies the problem type to be solved.

itype = 1

$$Az = \lambda Bz.$$

itype = 2

$$ABz = \lambda z.$$

itype = 3

$$BAz = \lambda z.$$

2: **jobz** – string

If **jobz** = 'N', compute eigenvalues only.

If **jobz** = 'V', compute eigenvalues and eigenvectors.

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

3: **uplo** – string

If **uplo** = 'U', the upper triangles of A and B are stored.

If **uplo** = 'L', the lower triangles 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 $\max(1, n)$

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

The n by n Hermitian matrix A .

If **uplo** = 'U', the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = 'L', the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

5: **b(ldb,*)** – complex array

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

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

The Hermitian matrix B :

if **uplo** = 'U', the leading n by n upper triangular part of **b** contains the upper triangular part of the matrix B ;

if **uplo** = 'L', the leading n by n lower triangular part of **b** contains the lower triangular part of the matrix B .

5.2 Optional Input Parameters

1: **n** – int32 scalar

Default: The first dimension of the arrays **a**, **b** and the second dimension of the arrays **a**, **b**. (An error is raised if these dimensions are not equal.)

n , the order of the matrices A and B .

Constraint: $n \geq 0$.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, work, lwork, rwork, lrwork, iwork, liwork

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 **jobz** = 'V', then if **info** = 0, **a** contains the matrix Z of eigenvectors. The eigenvectors are normalized as follows:

if **itype** = 1 or 2, $Z^H B Z = I$;
if **itype** = 3, $Z^H B^{-1} Z = I$.

If **jobz** = 'N', the upper triangle (if **uplo** = 'U') or the lower triangle (if **uplo** = 'L') of **a**, including the diagonal, is destroyed.

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 $\max(1, \mathbf{n})$

If **info** $\leq \mathbf{n}$, the part of **b** containing the matrix contains the triangular factor U or L from the Cholesky factorization $\mathbf{b} = U^H U$ or $\mathbf{b} = L L^H$.

3: **w(*)** – double array

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

If **info** = 0, the eigenvalues in ascending order.

4: **info** – int32 scalar

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

info = $-i$

If **info** = $-i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **itype**, 2: **jobz**, 3: **uplo**, 4: **n**, 5: **a**, 6: **lda**, 7: **b**, 8: **ldb**, 9: **w**, 10: **work**, 11: **lwork**, 12: **rwork**, 13: **lrwork**, 14: **iwor**, 15: **liwork**, 16: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

info > 0

f07fr or f08fq returned an error code:

$\leq \mathbf{n}$ if **info** = i , f08fq failed to converge; i off-diagonal elements of an intermediate tridiagonal form did not converge to zero;

$> \mathbf{n}$ if **info** = $\mathbf{n} + i$, for $1 \leq i \leq \mathbf{n}$, then the leading minor of order i of B is not positive-definite. The factorization of B could not be completed and no eigenvalues or eigenvectors were computed.

7 Accuracy

If B is ill-conditioned with respect to inversion, then the error bounds for the computed eigenvalues and vectors may be large, although when the diagonal elements of B differ widely in magnitude the

eigenvalues and eigenvectors may be less sensitive than the condition of B would suggest. See Section 4.10 of Anderson *et al.* 1999 for details of the error bounds.

The example program below illustrates the computation of approximate error bounds.

8 Further Comments

The total number of floating-point operations is proportional to n^3 .

The real analogue of this function is f08sc.

9 Example

```
itype = int32(2);
jobz = 'Vectors';
uplo = 'Upper';
a = [complex(-7.36, +0), complex(0.77, -0.43), complex(-0.64, -0.92),
      complex(3.01, -6.97);
      complex(0, 0), complex(3.49, +0), complex(2.19, +4.45), complex(1.9,
+3.73);
      complex(0, 0), complex(0, 0), complex(0.12, +0), complex(2.88, -
3.17);
      complex(0, +0), complex(0, 0), complex(0, 0), complex(-2.54, +0)];
b = [complex(3.23, +0), complex(1.51, -1.92), complex(1.9, +0.84),
      complex(0.42, +2.5);
      complex(0, 0), complex(3.58, +0), complex(-0.23, +1.11), complex(-
1.18, +1.37);
      complex(0, 0), complex(0, 0), complex(4.09, +0), complex(2.33, -
0.14);
      complex(0, 0), complex(0, 0), complex(0, 0), complex(4.29, +0)];
[aOut, bOut, w, info] = f08sq(itype, jobz, uplo, a, b)
```

```
aOut =
    0.2729 - 0.2791i    0.1219 - 0.1054i    0.9130 - 2.1011i   -0.1738 -
0.1110i
   -0.1187 + 0.1377i    0.3886 + 0.0797i   -0.8539 + 0.1787i    0.3785 +
0.0873i
    0.0548 - 0.0077i   -0.3047 + 0.4414i   -0.9232 + 0.9837i    0.1111 -
0.1894i
   -0.3177             0.2285             1.4319             0.3000
bOut =
    1.7972             0.8402 - 1.0683i    1.0572 + 0.4674i    0.2337 +
1.3910i
         0             1.3164             -0.4702 - 0.3131i    0.0834 -
0.0368i
         0             0             1.5604             0.9360 -
0.9900i
         0             0             0             0.6603
w =
   -61.7321
    -6.6195
     0.0725
    43.1883
info =
      0
```