NAG Toolbox for MATLAB

f08sp

1 Purpose

f08sp computes selected eigenvalues and, optionally, eigenvectors of a complex generalized Hermitian-definite eigenproblem, of the form

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

where A and B are Hermitian and B is also positive-definite. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

2 Syntax

```
[a, b, m, w, z, jfail, info] = f08sp(itype, jobz, range, uplo, a, b, vl,
vu, il, iu, abstol, 'n', n)
```

3 Description

f08sp first performs a Cholesky factorization of the matrix B as $B = U^{\mathrm{H}}U$, when **uplo** = 'U' or $B = LL^{\mathrm{H}}$, when **uplo** = 'L'. The generalized problem is then reduced to a standard symmetric eigenvalue problem

$$Cx = \lambda x,$$

which is solved for the desired eigenvalues and eigenvectors. The eigenvectors of C are then backtransformed to give the eigenvectors of the original problem.

For the problem $Az = \lambda Bz$ and $ABz = \lambda z$, the eigenvectors are normalized so that

$$z^{\mathrm{H}}Bz = I.$$

For the problem $BAz = \lambda z$ we correspondingly have

$$z^{\mathrm{H}}B^{-1}z = I.$$

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

Demmel J W and Kahan W 1990 Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

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: **range – string**

If **range** = 'A', all eigenvalues will be found.

If **range** = 'V', all eigenvalues in the half-open interval $(\mathbf{vl}, \mathbf{vu}]$ will be found.

If **range** = 'I', the **il**th to **iu**th eigenvalues will be found.

Constraint: **range** = 'A', 'V' or 'I'.

4: **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'.

5: **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 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.

6: **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 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.

7: **vl – double scalar**
8: **vu – double scalar**

If **range** = 'V', the lower and upper bounds of the interval to be searched for eigenvalues.

If **range** $=$ 'A' or 'I', **vl** and **vu** are not referenced.

Constraint: if **range** $=$ 'V', **vl** $<$ **vu**.

9: **il – int32 scalar**

10: **iu – int32 scalar**

If **range** $=$ 'I', the indices (in ascending order) of the smallest and largest eigenvalues to be returned.

If **range** $=$ 'A' or 'V', **il** and **iu** are not referenced.

Constraints:

if **n** $= 0$, **il** $= 1$ and **iu** $= 0$;
if **n** > 0, $1 \leq$ **il** $\leq$ **iu** $\leq$ **n**.

11: **abstol – double scalar**

The absolute error tolerance for the eigenvalues. An approximate eigenvalue is accepted as converged when it is determined to lie in an interval $[a,b]$ of width less than or equal to

$$\mathbf{abstol} + \epsilon \max(|a|,|b|),$$

where ϵ is the ***machine precision***. If **abstol** is less than or equal to zero, then $\epsilon\|T\|_1$ will be used in its place, where T is the tridiagonal matrix obtained by reducing C to tridiagonal form. Eigenvalues will be computed most accurately when **abstol** is set to twice the underflow threshold $2 \times$ x02am( ), not zero. If this function returns with **info** > 0, indicating that some eigenvectors did not converge, try setting **abstol** to $2 \times$ x02am( ). See Demmel and Kahan 1990.

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** ≥ 0.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, ldz, work, lwork, rwork, iwork

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

The lower triangle (if **uplo** $=$ 'L') or the upper triangle (if **uplo** $=$ 'U') 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** $\leq$ **n**, the part of **b** containing the matrix contains the triangular factor U or L from the Cholesky factorization **b** $= U^{\mathrm{H}}U$ or **b** $= LL^{\mathrm{H}}$.

3: **m – int32 scalar**

The total number of eigenvalues found.

If **range** $=$ 'A', $\mathbf{m} = \mathbf{n}$.

If **range** $=$ 'V', the exact value of **m** is not known in advance, but will satisfy $0 \leq \mathbf{m} \leq \mathbf{n}$.

If **range** $=$ 'I', $\mathbf{m} = \mathbf{iu} - \mathbf{il} + 1$.

4: **w**$(*)$ **– double array**

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

The first **m** elements contain the selected eigenvalues in ascending order.

5: **z**(**ldz**,$*$) **– complex array**

The first dimension, **ldz**, of the array **z** must satisfy

if **jobz** $=$ 'V', $\mathbf{ldz} \geq \max(1, \mathbf{n})$;
$\mathbf{ldz} \geq 1$ otherwise.

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

If **jobz** $=$ 'V', then if **info** $= 0$, the first m columns of Z contain the orthonormal eigenvectors of the matrix A corresponding to the selected eigenvalues, with the ith column of Z holding the eigenvector associated with $\mathbf{w}(i)$. The eigenvectors are normalised as follows:

if **itype** $= 1$ or 2, $Z^{\mathrm{T}}BZ = I$;
if **itype** $= 3$, $Z^{\mathrm{T}}B^{-1}Z = I$.

If **jobz** $=$ 'N', **z** is not referenced.

If an eigenvector fails to converge, then that column of Z contains the latest approximation to the eigenvector, and the index of the eigenvector is returned in **jfail**.

Note: you must ensure that at least $\max(1, \mathbf{m})$ columns are supplied in the array **z**; if **range** $=$ 'V', the exact value of M is not known in advance and an upper bound must be used.

6: **jfail**$(*)$ **– int32 array**

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

If **jobz** $=$ 'V', then if **info** $= 0$, the first **m** elements of **jfail** are zero.

If **info** > 0, **jfail** contains the indices of the eigenvectors that failed to converge.

If **jobz** $=$ 'E', **jfail** is not referenced.

7: **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: **range**, 4: **uplo**, 5: **n**, 6: **a**, 7: **lda**, 8: **b**, 9: **ldb**, 10: **vl**, 11: **vu**, 12: **il**, 13: **iu**, 14: **abstol**, 15: **m**, 16: **w**, 17: **z**, 18: **ldz**, 19: **work**, 20: **lwork**, 21: **rwork**, 22: **iwork**, 23: **jfail**, 24: **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 f08fp returned an error code:

$\leq$ **n** if **info** $= i$, f08fp failed to converge; i eigenvectors failed to converge. Their indices are stored in array **jfail**;

$>$ **n** if **info** $=$ **n** $+ i$, for $1 \leq i \leq$ **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.

8 Further Comments

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

The real analogue of this function is f08sb.

9 Example

```
itype = int32(1);
jobz = 'Vectors';
range = 'Values in range';
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)];
vl = -3;
vu = 3;
il = int32(0);
iu = int32(8185080);
abstol = 0;
[aOut, bOut, m, w, z, jfail, info] = ...
    f08sp(itype, jobz, range, uplo, a, b, vl, vu, il, iu, abstol)
```

```
aOut =
  -1.2636             -2.3214             -0.5211 - 0.0656i   -0.0802 +
0.4016i
        0             -1.8095             -2.7959             -0.1903 +
0.1121i
        0             0             -0.7025             -3.8021
        0             0             0             -0.7133
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
m =
           2
w =
   -2.9936
    0.5047
         0
         0
z =
  -0.3504 + 0.6060i   0.2835 - 0.5806i
  -0.0993 + 0.0631i  -0.3769 - 0.3194i
   0.6851 - 0.5987i  -0.3338 - 0.0134i
  -0.8127             0.6663
jfail =
           0
           0
           0
           0
info =
           0
```