NAG Toolbox for MATLAB

f08ju

1 Purpose

f08ju computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian positive-definite matrix which has been reduced to tridiagonal form.

2 Syntax

```
[d, e, z, info] = f08ju(compz, d, e, z, 'n', n)
```

3 Description

f08ju computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric positive-definite tridiagonal matrix T. In other words, it can compute the spectral factorization of T as

$$T = Z\Lambda Z^{\mathrm{T}},$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

$$Tz_i = \lambda_i z_i, \qquad i = 1, 2, \ldots, n.$$

The function stores the real orthogonal matrix Z in a ***complex*** array, so that it may be used to compute all the eigenvalues and eigenvectors of a complex Hermitian positive-definite matrix A which has been reduced to tridiagonal form T:

$$\begin{aligned} A &= QTQ^{\mathrm{H}}, \text{ where } Q \text{ is unitary} \\ &= (QZ)\Lambda(QZ)^{\mathrm{H}}. \end{aligned}$$

In this case, the matrix Q must be formed explicitly and passed to f08ju, which must be called with **compz** = 'V'. The functions which must be called to perform the reduction to tridiagonal form and form Q are:

full matrix	f08fs and f08ft
full matrix, packed storage	f08gs and f08gt
band matrix	f08hs with **vect** = 'V'.

f08ju first factorizes T as LDL^{H} where L is unit lower bidiagonal and D is diagonal. It forms the bidiagonal matrix $B = LD^{\frac{1}{2}}$, and then calls f08ms to compute the singular values of B which are the same as the eigenvalues of T. The method used by the function allows high relative accuracy to be achieved in the small eigenvalues of T. The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a complex factor of absolute value 1.

4 References

Barlow J and Demmel J W 1990 Computing accurate eigensystems of scaled diagonally dominant matrices *SIAM J. Numer. Anal.* **27** 762–791

5 Parameters

5.1 Compulsory Input Parameters

1: **compz – string**

Indicates whether the eigenvectors are to be computed.

compz = 'N'

Only the eigenvalues are computed (and the array **z** is not referenced).

compz = 'I'

The eigenvalues and eigenvectors of T are computed (and the array **z** is initialized by the function).

compz = 'V'

The eigenvalues and eigenvectors of A are computed (and the array **z** must contain the matrix Q on entry).

Constraint: **compz** = 'N', 'V' or 'I'.

2: **d**($*$) – **double array**

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

The diagonal elements of the tridiagonal matrix T.

3: **e**($*$) – **double array**

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

The off-diagonal elements of the tridiagonal matrix T.

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

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

if **compz** = 'V' or 'I', **ldz** $\geq \max(1, \mathbf{n})$;
if **compz** = 'N', **ldz** ≥ 1.

The second dimension of the array must be at least $\max(1, \mathbf{n})$ if **compz** = 'V' or 'I' and at least 1 if **compz** = 'N'

If **compz** = 'V', **z** must contain the unitary matrix Q from the reduction to tridiagonal form.

If **compz** = 'I', **z** need not be set.

5.2 Optional Input Parameters

1: **n** – **int32 scalar**

Default: The first dimension of the array **d** and the second dimension of the array **d**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix T.

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

5.3 Input Parameters Omitted from the MATLAB Interface

ldz, work

5.4 Output Parameters

1: **d**($*$) – **double array**

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

The n eigenvalues in descending order, unless **info** > 0, in which case **d** is overwritten.

2: **e($*$) – double array**

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

e is overwritten.

3: **z(ldz,$*$) – complex array**

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

if **compz** = 'V' or 'I', **ldz** $\geq \max(1, \mathbf{n})$;
if **compz** = 'N', **ldz** ≥ 1.

The second dimension of the array must be at least $\max(1, \mathbf{n})$ if **compz** = 'V' or 'I' and at least 1 if **compz** = 'N'

If **compz** = 'I' or 'V', the n required orthonormal eigenvectors stored as columns of Z; the ith column corresponds to the ith eigenvalue, where $i = 1, 2, \ldots, n$, unless **info** > 0.

If **compz** = 'N', **z** is not referenced.

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: **compz**, 2: **n**, 3: **d**, 4: **e**, 5: **z**, 6: **ldz**, 7: **work**, 8: **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

If **info** $= i$, the leading minor of order i is not positive-definite and the Cholesky factorization of T could not be completed. Hence T itself is not positive-definite.

If **info** $= \mathbf{n} + i$, the algorithm to compute the singular values of the Cholesky factor B failed to converge; i off-diagonal elements did not converge to zero.

7 Accuracy

The eigenvalues and eigenvectors of T are computed to high relative accuracy which means that if they vary widely in magnitude, then any small eigenvalues (and corresponding eigenvectors) will be computed more accurately than, for example, with the standard QR method. However, the reduction to tridiagonal form (prior to calling the function) may exclude the possibility of obtaining high relative accuracy in the small eigenvalues of the original matrix if its eigenvalues vary widely in magnitude.

To be more precise, let H be the tridiagonal matrix defined by $H = DTD$, where D is diagonal with $d_{ii} = t_{ii}^{-\frac{1}{2}}$, and $h_{ii} = 1$ for all i. If λ_i is an exact eigenvalue of T and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$\left|\tilde{\lambda}_i - \lambda_i\right| \leq c(n)\epsilon\kappa_2(H)\lambda_i$$

where $c(n)$ is a modestly increasing function of n, ϵ is the ***machine precision***, and $\kappa_2(H)$ is the condition number of H with respect to inversion defined by: $\kappa_2(H) = \|H\| \cdot \left\|H^{-1}\right\|$.

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

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\kappa_2(H)}{relgap_i}$$

where $relgap_i$ is the relative gap between λ_i and the other eigenvalues, defined by

$$relgap_i = \min_{i \neq j} \frac{|\lambda_i - \lambda_j|}{(\lambda_i + \lambda_j)}.$$

8 Further Comments

The total number of real floating-point operations is typically about $30n^2$ if **compz** = 'N' and about $12n^3$ if **compz** = 'V' or 'I', but depends on how rapidly the algorithm converges. When **compz** = 'N', the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when **compz** = 'V' or 'I' can be vectorized and on some machines may be performed much faster.

The real analogue of this function is f08jg.

9 Example

```
compz = 'V';
d = [6.02;
     2.738844788384059;
     5.173556804164482;
     2.467598407451455];
e = [2.74238946905796;
     1.835961995070032;
     1.695211553772095];
z = [complex(1, +0), complex(0, +0), complex(0, +0), complex(0, +0);
      complex(0, +0), complex(-0.1640904784230299, -0.09116137690168336),
...
          complex(0.04492226830902458,  -0.1991468061366732),  complex(-
0.7606249187911637, -0.5869720526411456);
       complex(0, +0), complex(-0.4740391598887533, -0.6344831832357161),
...
                    complex(-0.4067593168412005,     +0.4544041694574636),
complex(0.02193769252276673, +0.01733238795915084);
        complex(0,  +0),  complex(0.5287359860297633,  +0.240666035020444),
complex(-0.1787167294506699, ...
                 +0.7446116967739244),    complex(-0.2225496702687938,    -
0.1631058324212738)];
[dOut, eOut, zOut, info] = f08ju(compz, d, e, z)
```

```
dOut =
    7.9995
    5.9976
    2.0003
    0.4026
eOut =
     0
     0
     0
zOut =
   0.7289            -0.5130            0.2606           -0.3709
  -0.1651 - 0.2067i  -0.2486 - 0.3726i  -0.5981 - 0.4200i  -0.4009 -
0.1860i
  -0.4170 - 0.1413i  -0.3086 + 0.3554i   0.2957 + 0.1501i  -0.1848 -
0.6637i
   0.1748 + 0.4175i  -0.2188 + 0.5166i  -0.3501 - 0.4068i   0.4001 -
0.1798i
info =
           0
```