NAG Toolbox

nag_lapack_zsteqr (f08js)

1 Purpose

nag_lapack_zsteqr (f08js) computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian matrix which has been reduced to tridiagonal form.

2 Syntax

```
[d, e, z, info] = nag_lapack_zsteqr(compz, d, e, z, 'n', n)
[d, e, z, info] = f08js(compz, d, e, z, 'n', n)
```

3 Description

nag_lapack_zsteqr (f08js) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric 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

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

The function stores the real orthogonal matrix Z in a complex array, so that it may also be used to compute all the eigenvalues and eigenvectors of a complex Hermitian 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 nag_lapack_zsteqr (f08js), 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'.

nag_lapack_zsteqr (f08js) uses the implicitly shifted QR algorithm, switching between the QR and QL variants in order to handle graded matrices effectively (see Greenbaum and Dongarra (1980)). The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a complex factor of absolute value 1.

If only the eigenvalues of T are required, it is more efficient to call nag_lapack_dsterf (f08jf) instead. If T is positive definite, small eigenvalues can be computed more accurately by nag_lapack_zpteqr (f08ju).

4 References

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

Greenbaum A and Dongarra J J (1980) Experiments with QR/QL methods for the symmetric triangular eigenproblem *LAPACK Working Note No. 17 (Technical Report CS-89-92)* University of Tennessee, Knoxville http://www.netlib.org/lapack/lawnspdf/lawn17.pdf

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

5 Parameters

5.1 Compulsory Input Parameters

1: **compz** – CHARACTER(1)

Indicates whether the eigenvectors are to be computed.

compz = 'N'
Only the eigenvalues are computed (and the array **z** is not referenced).

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

compz = 'I'
The eigenvalues and eigenvectors of T are computed (and the array **z** is initialized by the function).

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

2: **d**(:) – REAL (KIND=nag_wp) array

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

The diagonal elements of the tridiagonal matrix T.

3: **e**(:) – REAL (KIND=nag_wp) array

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 (KIND=nag_wp) 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', $ldz \geq 1$.

The second dimension of the array **z** 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** – INTEGER

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 Output Parameters

1: **d**(:) – REAL (KIND=nag_wp) array

The dimension of the array **d** will be $\max(1, \mathbf{n})$

The n eigenvalues in ascending order, unless **info** > 0 (in which case see Section 6).

2: **e**(:) – REAL (KIND=nag_wp) array

The dimension of the array **e** will be $\max(1, \mathbf{n} - 1)$

3: **z**(ldz, :) – COMPLEX (KIND=nag_wp) array

The first dimension, ldz, of the array **z** will be

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

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

If **compz** = 'V' or 'I', 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** – INTEGER

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

6 Error Indicators and Warnings

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 (*warning*)

The algorithm has failed to find all the eigenvalues after a total of $30 \times$ **n** iterations. In this case, **d** and **e** contain on exit the diagonal and off-diagonal elements, respectively, of a tridiagonal matrix unitarily similar to T. If **info** $= i$, then i off-diagonal elements have not converged to zero.

7 Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrix $(T + E)$, where

$$\|E\|_2 = O(\epsilon)\|T\|_2,$$

and ϵ is the ***machine precision***.

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

$$\left|\tilde{\lambda}_i - \lambda_i\right| \leq c(n)\epsilon\|T\|_2,$$

where $c(n)$ is a modestly increasing function of n.

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:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\|T\|_2}{\min_{i\neq j}|\lambda_i - \lambda_j|}.$$

Thus the accuracy of a computed eigenvector depends on the gap between its eigenvalue and all the other eigenvalues.

8 Further Comments

The total number of real floating-point operations is typically about $24n^2$ if **compz** = 'N' and about $14n^3$ 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 nag_lapack_dsteqr (f08je).

9 Example

See Section 10 in nag_lapack_zungtr (f08ft), nag_lapack_zupgtr (f08gt) or nag_lapack_zhbtrd (f08hs), which illustrate the use of this function to compute the eigenvalues and eigenvectors of a full or band Hermitian matrix.

9.1 Program Text

```
function f08js_example

fprintf('f08js example results\n\n');

% Hermitian matrix A (Lower triangular part stored)
uplo = 'L';
a = [-2.28 + 0.00i,  0.00 + 0i,     0    + 0i,     0    + 0i;
      1.78 + 2.03i, -1.12 + 0i,     0    + 0i,     0    + 0i;
      2.26 - 0.10i,  0.01 - 0.43i, -0.37 + 0i,     0    + 0i;
     -0.12 - 2.53i, -1.07 - 0.86i,  2.31 + 0.92i, -0.73 + 0i];

% Reduce to tridiagonal form
[QT, d, e, tau, info] = f08fs( ...
                                uplo, a);

% Form Q
[Q, info] = f08ft( ...
                  uplo, QT, tau);

% Calculate eigenvalues/vectors of A from Q, d and e.
compz = 'V';
[w, ~, z, info] = f08js( ...
                         compz, d, e, Q);

% Normalize: largest elements are real
for i = 1:4
  [~,k] = max(abs(real(z(:,i)))+abs(imag(z(:,i))));
  z(:,i) = z(:,i)*conj(z(k,i))/abs(z(k,i));
end

disp(' Eigenvalues of A:');
disp(w');
disp(' Corresponding eigenvectors:');
disp(z);
```

9.2 Program Results

```
     f08js example results

 Eigenvalues of A:
   -6.0002   -3.0030    0.5036    3.9996

 Corresponding eigenvectors:
   0.7299 + 0.0000i  -0.2120 + 0.1497i   0.1000 - 0.3570i   0.1991 + 0.4720i
  -0.1663 - 0.2061i   0.7307 + 0.0000i   0.2863 - 0.3353i  -0.2467 + 0.3751i
  -0.4165 - 0.1417i  -0.3291 + 0.0479i   0.6890 + 0.0000i   0.4468 + 0.1466i
   0.1743 + 0.4162i   0.5200 + 0.1329i   0.0662 + 0.4347i   0.5612 + 0.0000i
```