NAG Toolbox

nag_lapack_zgges (f08xn)

1 Purpose

nag_lapack_zgges (f08xn) computes the generalized eigenvalues, the generalized Schur form (S, T) and, optionally, the left and/or right generalized Schur vectors for a pair of n by n complex nonsymmetric matrices (A, B).

2 Syntax

```
[a, b, sdim, alpha, beta, vsl, vsr, info] = nag_lapack_zgges(jobvsl, jobvsr, sort, selctg, a, b, 'n', n)
[a, b, sdim, alpha, beta, vsl, vsr, info] = f08xn(jobvsl, jobvsr, sort, selctg, a, b, 'n', n)
```

3 Description

The generalized Schur factorization for a pair of complex matrices (A, B) is given by

$$A = QSZ^{\mathrm{H}}, \quad B = QTZ^{\mathrm{H}},$$

where Q and Z are unitary, T and S are upper triangular. The generalized eigenvalues, λ, of (A, B) are computed from the diagonals of T and S and satisfy

$$Az = \lambda Bz,$$

where z is the corresponding generalized eigenvector. λ is actually returned as the pair (α, β) such that

$$\lambda = \alpha/\beta$$

since β, or even both α and β can be zero. The columns of Q and Z are the left and right generalized Schur vectors of (A, B).

Optionally, nag_lapack_zgges (f08xn) can order the generalized eigenvalues on the diagonals of (S, T) so that selected eigenvalues are at the top left. The leading columns of Q and Z then form an orthonormal basis for the corresponding eigenspaces, the deflating subspaces.

nag_lapack_zgges (f08xn) computes T to have real non-negative diagonal entries. The generalized Schur factorization, before reordering, is computed by the QZ algorithm.

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 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: **jobvsl** – CHARACTER(1)

If **jobvsl** = 'N', do not compute the left Schur vectors.

If **jobvsl** = 'V', compute the left Schur vectors.

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

2: **jobvsr** – CHARACTER(1)

If **jobvsr** = 'N', do not compute the right Schur vectors.

If **jobvsr** = 'V', compute the right Schur vectors.

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

3: **sort** – CHARACTER(1)

Specifies whether or not to order the eigenvalues on the diagonal of the generalized Schur form.

sort = 'N'
Eigenvalues are not ordered.

sort = 'S'
Eigenvalues are ordered (see **selctg**).

Constraint: **sort** = 'N' or 'S'.

4: **selctg** – LOGICAL FUNCTION, supplied by the user.

If **sort** = 'S', **selctg** is used to select generalized eigenvalues to the top left of the generalized Schur form.

If **sort** = 'N', **selctg** is not referenced by nag_lapack_zgges (f08xn), and may be called with the string `'f08xnz'`.

```
[result] = selctg(a, b)
```

Input Parameters

1: **a** – COMPLEX (KIND=nag_wp)
2: **b** – COMPLEX (KIND=nag_wp)

An eigenvalue $\mathbf{a}(j)/\mathbf{b}(j)$ is selected if $\mathbf{selctg}(\mathbf{a}(j), \mathbf{b}(j))$ is *true*.

Note that in the ill-conditioned case, a selected generalized eigenvalue may no longer satisfy $\mathbf{selctg}(\mathbf{a}(j), \mathbf{b}(j)) = true$ after ordering. $\mathbf{info} = \mathbf{n} + 2$ in this case.

Output Parameters

1: **result**

$result = true$ for selected eigenvalues.

5: $\mathbf{a}(lda, :)$ – COMPLEX (KIND=nag_wp) array

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

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

The first of the pair of matrices, A.

6: $\mathbf{b}(ldb, :)$ – COMPLEX (KIND=nag_wp) array

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

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

The second of the pair of matrices, B.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **a**, **b** and the second dimension of the arrays **a**, **b**.

n, the order of the matrices A and B.

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

5.3 Output Parameters

1: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

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

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

a stores its generalized Schur form S.

2: **b**$(ldb,:)$ – COMPLEX (KIND=nag_wp) array

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

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

b stores its generalized Schur form T.

3: **sdim** – INTEGER

If **sort** = 'N', **sdim** = 0.

If **sort** = 'S', **sdim** = number of eigenvalues (after sorting) for which **selctg** is *true*.

4: **alpha**(**n**) – COMPLEX (KIND=nag_wp) array

See the description of **beta**.

5: **beta**(**n**) – COMPLEX (KIND=nag_wp) array

alpha(j)/**beta**(j), for $j = 1,2,\ldots,\mathbf{n}$, will be the generalized eigenvalues. **alpha**(j), for $j = 1,2,\ldots,\mathbf{n}$ and **beta**(j), for $j = 1,2,\ldots,\mathbf{n}$, are the diagonals of the complex Schur form (A,B) output by nag_lapack_zgges (f08xn). The **beta**(j) will be non-negative real.

Note: the quotients **alpha**(j)/**beta**(j) may easily overflow or underflow, and **beta**(j) may even be zero. Thus, you should avoid naively computing the ratio α/β. However, **alpha** will always be less than and usually comparable with $\|\mathbf{a}\|$ in magnitude, and **beta** will always be less than and usually comparable with $\|\mathbf{b}\|$.

6: **vsl**$(ldvsl,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, $ldvsl$, of the array **vsl** will be

if **jobvsl** = 'V', $ldvsl = \max(1,\mathbf{n})$;
otherwise $ldvsl = 1$.

The second dimension of the array **vsl** will be $\max(1,\mathbf{n})$ if **jobvsl** = 'V' and 1 otherwise.

If **jobvsl** = 'V', **vsl** will contain the left Schur vectors, Q.

If **jobvsl** = 'N', **vsl** is not referenced.

7: **vsr**$(ldvsr,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, $ldvsr$, of the array **vsr** will be

if **jobvsr** = 'V', $ldvsr = \max(1,\mathbf{n})$;
otherwise $ldvsr = 1$.

The second dimension of the array **vsr** will be $\max(1, \mathbf{n})$ if **jobvsr** = 'V' and 1 otherwise.

If **jobvsr** = 'V', **vsr** will contain the right Schur vectors, Z.

If **jobvsr** = 'N', **vsr** is not referenced.

8: **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: **jobvsl**, 2: **jobvsr**, 3: **sort**, 4: **selctg**, 5: **n**, 6: **a**, 7: **lda**, 8: **b**, 9: **ldb**, 10: **sdim**, 11: **alpha**, 12: **beta**, 13: **vsl**, 14: **ldvsl**, 15: **vsr**, 16: **ldvsr**, 17: **work**, 18: **lwork**, 19: **rwork**, 20: **bwork**, 21: **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 = 1 to **n** (*warning*)

The QZ iteration failed. (A, B) are not in Schur form, but **alpha**(j) and **beta**(j) should be correct for $j =$ **info** $+ 1, \ldots,$ **n**.

info = **n** + 1

Unexpected error returned from nag_lapack_zhgeqz (f08xs).

info = **n** + 2 (*warning*)

After reordering, roundoff changed values of some complex eigenvalues so that leading eigenvalues in the generalized Schur form no longer satisfy **selctg** = *true*. This could also be caused by underflow due to scaling.

info = **n** + 3 (*warning*)

The eigenvalues could not be reordered because some eigenvalues were too close to separate (the problem is very ill-conditioned).

7 Accuracy

The computed generalized Schur factorization satisfies

$$A + E = QSZ^{\mathrm{H}}, \quad B + F = QTZ^{\mathrm{H}},$$

where

$$\|(E, F)\|_F = O(\epsilon)\|(A, B)\|_F$$

and ϵ is the ***machine precision***. See Section 4.11 of Anderson *et al.* (1999) for further details.

8 Further Comments

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

The real analogue of this function is nag_lapack_dgges (f08xa).

9 Example

This example finds the generalized Schur factorization of the matrix pair (A, B), where

$$A = \begin{pmatrix} -21.10 - 22.50i & 53.50 - 50.50i & -34.50 + 127.50i & 7.50 + 0.50i \\ -0.46 - 7.78i & -3.50 - 37.50i & -15.50 + 58.50i & -10.50 - 1.50i \\ 4.30 - 5.50i & 39.70 - 17.10i & -68.50 + 12.50i & -7.50 - 3.50i \\ 5.50 + 4.40i & 14.40 + 43.30i & -32.50 - 46.00i & -19.00 - 32.50i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1.00 - 5.00i & 1.60 + 1.20i & -3.00 + 0.00i & 0.00 - 1.00i \\ 0.80 - 0.60i & 3.00 - 5.00i & -4.00 + 3.00i & -2.40 - 3.20i \\ 1.00 + 0.00i & 2.40 + 1.80i & -4.00 - 5.00i & 0.00 - 3.00i \\ 0.00 + 1.00i & -1.80 + 2.40i & 0.00 - 4.00i & 4.00 - 5.00i \end{pmatrix}.$$

Note that the block size (NB) of 64 assumed in this example is not realistic for such a small problem, but should be suitable for large problems.

9.1 Program Text

```
     function f08xn_example

fprintf('f08xn example results\n\n');

% Matrix pair (A,B)
A = [ -21.10 - 22.50i,  53.5 - 50.5i,  -34.5 + 127.5i,   7.5 +  0.5i;
       -0.46 -  7.78i,  -3.5 - 37.5i,  -15.5 +  58.5i, -10.5 -  1.5i;
        4.30 -  5.50i,  39.7 - 17.1i,  -68.5 +  12.5i,  -7.5 -  3.5i;
        5.50 +  4.40i,  14.4 + 43.3i,  -32.5 -  46i,   -19.0 - 32.5i];
B = [ 1   - 5i,    1.6 + 1.2i, -3 + 0i,  0   - 1i;
      0.8 - 0.6i,  3.0 - 5.0i, -4 + 3i, -2.4 - 3.2i;
      1   + 0i,    2.4 + 1.8i, -4 - 5i,  0   - 3i;
      0   + 1i,   -1.8 + 2.4i,  0 - 4i,  4   - 5i];

% Generalized Schur form (S,T) of (A,B), generalized eigenvalues
% and Schur vectors Q and Z with no sorting
jobvsl = 'Vectors (left)';
jobvsr = 'Vectors (right)';
sortp = 'No sort';

[S, T, sdim, alpha, beta, VSL, VSR, info] = ...
  f08xn( ...
         jobvsl, jobvsr, sortp, @selctg, A, B);

% Display eigenvalues in order of magnitude
disp('Generalized eigenvalues')
eigs = alpha./beta;
disp(sort(eigs));
```

9.2 Program Results

```
     f08xn example results

Generalized eigenvalues
   3.0000 - 1.0000i
   2.0000 - 5.0000i
   4.0000 - 5.0000i
   3.0000 - 9.0000i
```