NAG Toolbox

nag_lapack_ztgsen (f08yu)

1 Purpose

nag_lapack_ztgsen (f08yu) reorders the generalized Schur factorization of a complex matrix pair in generalized Schur form, so that a selected cluster of eigenvalues appears in the leading elements on the diagonal of the generalized Schur form. The function also, optionally, computes the reciprocal condition numbers of the cluster of eigenvalues and/or corresponding deflating subspaces.

2 Syntax

```
[a, b, alpha, beta, q, z, m, pl, pr, dif, info] = nag_lapack_ztgsen(ijob, wantq,
wantz, select, a, b, q, z, 'n', n)
```

```
[a, b, alpha, beta, q, z, m, pl, pr, dif, info] = f08yu(ijob, wantq, wantz,
select, a, b, q, z, 'n', n)
```

3 Description

nag_lapack_ztgsen (f08yu) factorizes the generalized complex n by n matrix pair (S,T) in generalized Schur form, using a unitary equivalence transformation as

$$S=\hat{Q}\hat{S}\hat{Z}^{\mathrm{H}},\quad T=\hat{Q}\hat{T}\hat{Z}^{\mathrm{H}},$$

where $\left(\hat{S},\hat{T}\right)$ are also in generalized Schur form and have the selected eigenvalues as the leading diagonal elements. The leading columns of Q and Z are the generalized Schur vectors corresponding to the selected eigenvalues and form orthonormal subspaces for the left and right eigenspaces (deflating subspaces) of the pair (S,T).

The pair (S,T) are in generalized Schur form if S and T are upper triangular as returned, for example, by nag_lapack_zgges (f08xn), or nag_lapack_zhgeqz (f08xs) with **job** $=$ 'S'. The diagonal elements define the generalized eigenvalues (α_i,β_i), for $i=1,2,\ldots,n$, of the pair (S,T). The eigenvalues are given by

$$\lambda_i=\alpha_i/\beta_i,$$

but are returned as the pair (α_i,β_i) in order to avoid possible overflow in computing λ_i. Optionally, the function returns reciprocals of condition number estimates for the selected eigenvalue cluster, p and q, the right and left projection norms, and of deflating subspaces, Dif_u and Dif_l. For more information see Sections 2.4.8 and 4.11 of Anderson *et al.* (1999).

If S and T are the result of a generalized Schur factorization of a matrix pair (A,B)

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

then, optionally, the matrices Q and Z can be updated as $Q\hat{Q}$ and $Z\hat{Z}$. Note that the condition numbers of the pair (S,T) are the same as those of the pair (A,B).

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

5 Parameters

5.1 Compulsory Input Parameters

1: **ijob** – INTEGER

Specifies whether condition numbers are required for the cluster of eigenvalues (p and q) or the deflating subspaces (Dif_u and Dif_l).

ijob $= 0$
: Only reorder with respect to **select**. No extras.

ijob $= 1$
: Reciprocal of norms of 'projections' onto left and right eigenspaces with respect to the selected cluster (p and q).

ijob $= 2$
: The upper bounds on Dif_u and Dif_l. F-norm-based estimate (**dif**$(1:2)$).

ijob $= 3$
: Estimate of Dif_u and Dif_l. 1-norm-based estimate (**dif**$(1:2)$). About five times as expensive as **ijob** $= 2$.

ijob $= 4$
: Compute **pl**, **pr** and **dif** as in **ijob** $= 0$, 1 and 2. Economic version to get it all.

ijob $= 5$
: Compute **pl**, **pr** and **dif** as in **ijob** $= 0$, 1 and 3.

Constraint: $0 \leq$ **ijob** ≤ 5.

2: **wantq** – LOGICAL

If **wantq** $=$ *true*, update the left transformation matrix Q.

If **wantq** $=$ *false*, do not update Q.

3: **wantz** – LOGICAL

If **wantz** $=$ *true*, update the right transformation matrix Z.

If **wantz** $=$ *false*, do not update Z.

4: **select**(**n**) – LOGICAL array

Specifies the eigenvalues in the selected cluster. To select an eigenvalue λ_j, **select**(j) must be set to *true*.

5: **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 matrix S in the pair (S,T).

6: **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 matrix T, in the pair (S,T).

7: **q**(ldq, :) – COMPLEX (KIND=nag_wp) array

The first dimension, ldq, of the array **q** must satisfy

if **wantq** $=$ *true*, $ldq \geq \max(1, \mathbf{n})$;
otherwise $ldq \geq 1$.

The second dimension of the array **q** must be at least $\max(1, \mathbf{n})$ if **wantq** $=$ *true*, and at least 1 otherwise.

If **wantq** $=$ *true*, the n by n matrix Q.

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

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

if **wantz** $=$ *true*, $ldz \geq \max(1, \mathbf{n})$;
otherwise $ldz \geq 1$.

The second dimension of the array **z** must be at least $\max(1, \mathbf{n})$ if **wantz** $=$ *true*, and at least 1 otherwise.

If **wantz** $=$ *true*, the n by n matrix Z.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **select** and 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 S and T.

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

The updated matrix $\hat{S}$.

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

The updated matrix $\hat{T}$

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

alpha and **beta** contain diagonal elements of $\hat{S}$ and $\hat{T}$, respectively, when the pair (S, T) has been reduced to generalized Schur form. **alpha**(i)/**beta**(i), for $i = 1, 2, \ldots, \mathbf{n}$, are the eigenvalues.

5: **q**(ldq, :) – COMPLEX (KIND=nag_wp) array

The first dimension, ldq, of the array **q** will be

if **wantq** $=$ *true*, $ldq = \max(1, \mathbf{n})$;
otherwise $ldq = 1$.

The second dimension of the array **q** will be $\max(1, \mathbf{n})$ if **wantq** $=$ *true* and 1 otherwise.

If **wantq** $=$ *true*, the updated matrix $Q\hat{Q}$.

If **wantq** $=$ *false*, **q** is not referenced.

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

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

if **wantz** $=$ *true*, $ldz = \max(1, \mathbf{n})$;
otherwise $ldz = 1$.

The second dimension of the array **z** will be $\max(1, \mathbf{n})$ if **wantz** $=$ *true* and 1 otherwise.

If **wantz** $=$ *true*, the updated matrix $Z\hat{Z}$.

If **wantz** $=$ *false*, **z** is not referenced.

7: **m** – INTEGER

The dimension of the specified pair of left and right eigenspaces (deflating subspaces).

8: **pl** – REAL (KIND=nag_wp)
9: **pr** – REAL (KIND=nag_wp)

If **ijob** $= 1$, 4 or 5, **pl** and **pr** are lower bounds on the reciprocal of the norm of 'projections' p and q onto left and right eigenspace with respect to the selected cluster. $0 <$ **pl**, **pr** ≤ 1.

If **m** $= 0$ or **m** $=$ **n**, **pl** $=$ **pr** $= 1$.

If **ijob** $= 0$, 2 or 3, **pl** and **pr** are not referenced.

10: **dif**$(:)$ – REAL (KIND=nag_wp) array

The dimension of the array **dif** will be 2

If **ijob** ≥ 2, **dif**$(1:2)$ store the estimates of Dif_u and Dif_l.

If **ijob** $= 2$ or 4, **dif**$(1:2)$ are F-norm-based upper bounds on Dif_u and Dif_l.

If **ijob** $= 3$ or 5, **dif**$(1:2)$ are 1-norm-based estimates of Dif_u and Dif_l.

If **m** $= 0$ or n, **dif**$(1:2) = \|(A,B)\|_F$.

If **ijob** $= 0$ or 1, **dif** is not referenced.

11: **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: **ijob**, 2: **wantq**, 3: **wantz**, 4: **select**, 5: **n**, 6: **a**, 7: **lda**, 8: **b**, 9: **ldb**, 10: **alpha**, 11: **beta**, 12: **q**, 13: **ldq**, 14: **z**, 15: **ldz**, 16: **m**, 17: **pl**, 18: **pr**, 19: **dif**, 20: **work**, 21: **lwork**, 22: **iwork**, 23: **liwork**, 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 $= 1$

Reordering of (S, T) failed because the transformed matrix pair $\left(\hat{S}, \hat{T}\right)$ would be too far from generalized Schur form; the problem is very ill-conditioned. (S, T) may have been partially reordered. If requested, 0 is returned in **dif**$(1:2)$, **pl** and **pr**.

7 Accuracy

The computed generalized Schur form is nearly the exact generalized Schur form for nearby matrices $(S + E)$ and $(T + F)$, where

$$\|E\|_2 = O\epsilon\|S\|_2 \quad \text{and} \quad \|F\|_2 = O\epsilon\|T\|_2,$$

and ϵ is the ***machine precision***. See Section 4.11 of Anderson *et al.* (1999) for further details of error bounds for the generalized nonsymmetric eigenproblem, and for information on the condition numbers returned.

8 Further Comments

The real analogue of this function is nag_lapack_dtgsen (f08yg).

9 Example

This example reorders the generalized Schur factors S and T and update the matrices Q and Z given by

$$S = \begin{pmatrix} 4.0 + 4.0i & 1.0 + 1.0i & 1.0 + 1.0i & 2.0 - 1.0i \\ 0 & 2.0 + 1.0i & 1.0 + 1.0i & 1.0 + 1.0i \\ 0 & 0 & 2.0 - 1.0i & 1.0 + 1.0i \\ 0 & 0 & 0 & 6.0 - 2.0i \end{pmatrix},$$

$$T = \begin{pmatrix} 2.0 & 1.0 + 1.0i & 1.0 + 1.0i & 3.0 - 1.0i \\ 0 & 1.0 & 2.0 + 1.0i & 1.0 + 1.0i \\ 0 & 0 & 1.0 & 1.0 + 1.0i \\ 0 & 0 & 0 & 2.0 \end{pmatrix},$$

$$Q = \begin{pmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix},$$

selecting the second and third generalized eigenvalues to be moved to the leading positions. Bases for the left and right deflating subspaces, and estimates of the condition numbers for the eigenvalues and Frobenius norm based bounds on the condition numbers for the deflating subspaces are also output.

9.1 Program Text

```
     function f08yu_example

fprintf('f08yu example results\n\n');

% Generalized Schur form matrix pair
n = 4;
S = [ 4 + 4i,  1 + 1i,  1 + 1i,  2 - 1i;
      0 + 0i,  2 + 1i,  1 + 1i,  1 + 1i;
      0 + 0i,  0 + 0i,  2 - 1i,  1 + 1i;
      0 + 0i,  0 + 0i,  0 + 0i,  6 - 2i];
T = [ 2,       1 + 1i,  1 + 1i,  3 - 1i;
      0 + 0i,  1 + 0i,  2 + 1i,  1 + 1i;
      0 + 0i,  0 + 0i,  1 + 0i,  1 + 1i;
      0 + 0i,  0 + 0i,  0 + 0i,  2 + 0i];

% Want equivalence transformation matrices Q and Z
wantq = true;
```

```
wantz = true;
Q = complex(eye(n));
Z = Q;

% reorder 2nd and 3rd eigenvalues, and
% get projection norms and upper bound estimates
select = [false;     true;     true;     false];
ijob = nag_int(4);

% Reorder the Schur factors S and T and update the matrices Q and Z
[S, T, alpha, beta, Q, Z, m, pl, pr, dif, info] = ...
  f08yu( ...
         ijob, wantq, wantz, select, S, T, Q, Z);

fprintf('Number of selected eigenvalues  = %4d\n\n', m);
eigs = alpha./beta;
disp('Selected Generalized Eigenvalues')
disp(eigs(1:m));
mtitle = 'Basis of left deflating invariant subspace';
[ifail] = x04db( ...
                 'Gen', ' ', Q(:,1:m), 'B', ' ', mtitle, ...
                 'Int', 'Int', nag_int(80), nag_int(0));
fprintf('\n');
mtitle = 'Basis of right deflating invariant subspace';
[ifail] = x04db( ...
                 'Gen', ' ', Z(:,1:m), 'B', ' ', mtitle, ...
                 'Int', 'Int', nag_int(80), nag_int(0));
fprintf('\n%s%s\n%10.2e\n', ...
        'Norm estimate of projection onto  left eigenspace ', ...
        'for selected cluster', 1/pl);
fprintf('\n%s%s\n%10.2e\n', ...
        'Norm estimate of projection onto right eigenspace ', ...
        'for selected cluster', 1/pr);
fprintf('\nF-norm based upper bound on Difu\n%10.2e\n', dif(1));
fprintf('\nF-norm based upper bound on Difl\n%10.2e\n', dif(2));
```

9.2 Program Results

```
     f08yu example results

Number of selected eigenvalues  =    2

Selected Generalized Eigenvalues
   2.0000 + 1.0000i
   2.0000 - 1.0000i

 Basis of left deflating invariant subspace
                     1                   2
 1  (  0.9045,  0.3015) ( -0.0033, -0.2397)
 2  (  0.3015,  0.0000) (  0.2497,  0.7157)
 3  (  0.0000,  0.0000) (  0.0549,  0.6042)
 4  (  0.0000,  0.0000) (  0.0000,  0.0000)

 Basis of right deflating invariant subspace
                     1                   2
 1  (  0.7071,  0.0000) ( -0.5607,  0.0000)
 2  (  0.7071,  0.0000) (  0.5607,  0.0000)
 3  (  0.0000,  0.0000) (  0.0552,  0.6067)
 4  (  0.0000,  0.0000) (  0.0000,  0.0000)

Norm estimate of projection onto  left eigenspace for selected cluster
  8.90e+00

Norm estimate of projection onto right eigenspace for selected cluster
  7.02e+00
```

```
F-norm based upper bound on Difu
  2.18e-01

F-norm based upper bound on Difl
  2.62e-01
```