

NAG Toolbox

nag_lapack_zggev (f08wn)

1 Purpose

`nag_lapack_zggev (f08wn)` computes for a pair of n by n complex nonsymmetric matrices (A, B) the generalized eigenvalues and, optionally, the left and/or right generalized eigenvectors using the QZ algorithm.

2 Syntax

```
[a, b, alpha, beta, vl, vr, info] = nag_lapack_zggev(jobvl, jobvr, a, b, 'n', n)
[a, b, alpha, beta, vl, vr, info] = f08wn(jobvl, jobvr, a, b, 'n', n)
```

3 Description

A generalized eigenvalue for a pair of matrices (A, B) is a scalar λ or a ratio $\alpha/\beta = \lambda$, such that $A - \lambda B$ is singular. It is usually represented as the pair (α, β) , as there is a reasonable interpretation for $\beta = 0$, and even for both being zero.

The right generalized eigenvector v_j corresponding to the generalized eigenvalue λ_j of (A, B) satisfies

$$Av_j = \lambda_j Bv_j.$$

The left generalized eigenvector u_j corresponding to the generalized eigenvalue λ_j of (A, B) satisfies

$$u_j^H A = \lambda_j u_j^H B,$$

where u_j^H is the conjugate-transpose of u_j .

All the eigenvalues and, if required, all the eigenvectors of the complex generalized eigenproblem $Ax = \lambda Bx$, where A and B are complex, square matrices, are determined using the QZ algorithm. The complex QZ algorithm consists of three stages:

1. A is reduced to upper Hessenberg form (with real, non-negative subdiagonal elements) and at the same time B is reduced to upper triangular form.
2. A is further reduced to triangular form while the triangular form of B is maintained and the diagonal elements of B are made real and non-negative. This is the generalized Schur form of the pair (A, B).

This function does not actually produce the eigenvalues λ_j , but instead returns α_j and β_j such that

$$\lambda_j = \alpha_j / \beta_j, \quad j = 1, 2, \dots, n.$$

The division by β_j becomes your responsibility, since β_j may be zero, indicating an infinite eigenvalue.

3. If the eigenvectors are required they are obtained from the triangular matrices and then transformed back into the original coordinate system.

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

Wilkinson J H (1979) Kronecker's canonical form and the *QZ* algorithm *Linear Algebra Appl.* **28** 285–303

5 Parameters

5.1 Compulsory Input Parameters

1: **jobvl** – CHARACTER(1)

If **jobvl** = 'N', do not compute the left generalized eigenvectors.

If **jobvl** = 'V', compute the left generalized eigenvectors.

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

2: **jobvr** – CHARACTER(1)

If **jobvr** = 'N', do not compute the right generalized eigenvectors.

If **jobvr** = 'V', compute the right generalized eigenvectors.

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

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

The first dimension of the array **a** must be at least max(1, **n**).

The second dimension of the array **a** must be at least max(1, **n**).

The matrix *A* in the pair (*A*, *B*).

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

The first dimension of the array **b** must be at least max(1, **n**).

The second dimension of the array **b** must be at least max(1, **n**).

The matrix *B* in the pair (*A*, *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**. (An error is raised if these dimensions are not equal.)

n, the order of the matrices *A* and *B*.

Constraint: **n** ≥ 0 .

5.3 Output Parameters

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

The first dimension of the array **a** will be max(1, **n**).

The second dimension of the array **a** will be max(1, **n**).

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

The first dimension of the array **b** will be max(1, **n**).

The second dimension of the array **b** will be max(1, **n**).

b has been overwritten.

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

See the description of **beta**.

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

alpha(*j*)/**beta**(*j*), for $j = 1, 2, \dots, n$, will be the generalized eigenvalues.

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_j/β_j . However, $\max|\alpha_j|$ will always be less than and usually comparable with $\|\mathbf{a}\|_2$ in magnitude, and $\max|\beta_j|$ will always be less than and usually comparable with $\|\mathbf{b}\|_2$.

5: **vl**(*ldvl*,:) – COMPLEX (KIND=nag_wp) array

The first dimension, *ldvl*, of the array **vl** will be

if **jobvl** = 'V', *ldvl* = max(1, **n**);
otherwise *ldvl* = 1.

The second dimension of the array **vl** will be max(1, **n**) if **jobvl** = 'V' and 1 otherwise.

If **jobvl** = 'V', the left generalized eigenvectors u_j are stored one after another in the columns of **vl**, in the same order as the corresponding eigenvalues. Each eigenvector will be scaled so the largest component will have |real part| + |imag. part| = 1.

If **jobvl** = 'N', **vl** is not referenced.

6: **vr**(*ldvr*,:) – COMPLEX (KIND=nag_wp) array

The first dimension, *ldvr*, of the array **vr** will be

if **jobvr** = 'V', *ldvr* = max(1, **n**);
otherwise *ldvr* = 1.

The second dimension of the array **vr** will be max(1, **n**) if **jobvr** = 'V' and 1 otherwise.

If **jobvr** = 'V', the right generalized eigenvectors v_j are stored one after another in the columns of **vr**, in the same order as the corresponding eigenvalues. Each eigenvector will be scaled so the largest component will have |real part| + |imag. part| = 1.

If **jobvr** = 'N', **vr** is not referenced.

7: **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: **jobvl**, 2: **jobvr**, 3: **n**, 4: **a**, 5: **lda**, 6: **b**, 7: **ldb**, 8: **alpha**, 9: **beta**, 10: **vl**, 11: **ldvl**, 12: **vr**, 13: **ldvr**, 14: **work**, 15: **lwork**, 16: **rwork**, 17: **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. No eigenvectors have been calculated, but **alpha(j)** and **beta(j)** should be correct for $j = \text{info} + 1, \dots, \text{n}$.

info = n + 1

Unexpected error returned from nag_lapack_zhgeqz (f08xs).

info = n + 2

Error returned from nag_lapack_ztgevc (f08yx).

7 Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrices $(A + E)$ and $(B + F)$, 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.

Note: interpretation of results obtained with the *QZ* algorithm often requires a clear understanding of the effects of small changes in the original data. These effects are reviewed in Wilkinson (1979), in relation to the significance of small values of α_j and β_j . It should be noted that if α_j and β_j are **both** small for any j , it may be that no reliance can be placed on **any** of the computed eigenvalues $\lambda_i = \alpha_i/\beta_i$. You are recommended to study Wilkinson (1979) and, if in difficulty, to seek expert advice on determining the sensitivity of the eigenvalues to perturbations in the data.

8 Further Comments

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

The real analogue of this function is nag_lapack_dggev (f08wa).

9 Example

This example finds all the eigenvalues and right eigenvectors 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 f08wn_example

fprintf('f08wn example results\n\n');

% Generalized eigenvalues and right eigenvectors of (A, B):
n = 4;
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 - 46.00i, -19.0 - 32.5i];
b = [ 1     - 5i,      1.6 + 1.2i,   -3    + 0i,      0     - 1i;
      0.8 - 0.6i,      3     - 5i,      -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];

jobvl = 'No left vectors';
jobvr = 'Vectors (right)';
[~, ~, alpha, beta, ~, VR, info] = ...
f08wn( ...
jobvl, jobvr, a, b);

small = x02am;

[eigs, pos] = sort(alpha./beta);

for j=1:n
  if (abs(alpha(pos(j)))*small >= abs(beta(pos(j))))
    fprintf('\n%d: Eigenvalue is numerically infinite or undetermined\n',j);
    fprintf('%4d: alpha = (%11.4e,%11.4e), beta = (%11.4e,%11.4e)\n', ...
    j, real(alpha(j)), imag(alpha(j)), real(beta(j)), imag(beta(j)));
  else
    fprintf('Eigenvalue (%d):\n', j);
    disp(eigs(j));
  end

  fprintf('Eigenvector (%d):\n', j);
  disp(VR(:, pos(j))/VR(1, pos(j)));
end

```

9.2 Program Results

f08wn example results

Eigenvalue (1):
3.0000 - 1.0000i

Eigenvector (1):
1.0000 + 0.0000i
0.1600 - 0.1200i
0.1200 - 0.1600i
0.1600 + 0.1200i

Eigenvalue (2):
2.0000 - 5.0000i

Eigenvector (2):
1.0000 - 0.0000i
0.0046 - 0.0034i
0.0629 + 0.0000i
-0.0000 + 0.0629i

Eigenvalue (3):
4.0000 - 5.0000i

Eigenvector (3):
1.0000 + 0.0000i
0.0089 - 0.0067i
-0.0333 + 0.0000i
-0.0000 + 0.1556i

Eigenvalue (4):
3.0000 - 9.0000i

```
Eigenvector (4):  
 1.0000 - 0.0000i  
 0.1600 - 0.1200i  
 0.1200 + 0.1600i  
-0.1600 + 0.1200i
```
