NAG Toolbox for MATLAB

f08xs

1 Purpose

f08xs implements the QZ method for finding generalized eigenvalues of the complex matrix pair (A, B) of order n, which is in the generalized upper Hessenberg form.

2 Syntax

```
[a, b, alpha, beta, q, z, info] = f08xs(job, compq, compz, ilo, ihi, a,
b, q, z, 'n', n)
```

3 Description

f08xs implements a single-shift version of the QZ method for finding the generalized eigenvalues of the complex matrix pair (A, B) which is in the generalized upper Hessenberg form. If the matrix pair (A, B) is not in the generalized upper Hessenberg form, then the function f08ws should be called before invoking f08xs.

This problem is mathematically equivalent to solving the matrix equation

$$\det(A - \lambda B) = 0.$$

Note that, to avoid underflow, overflow and other arithmetic problems, the generalized eigenvalues λ_j are never computed explicitly by this function but defined as ratios between two computed values, α_j and β_j:

$$\lambda_j = \alpha_j / \beta_j.$$

The parameters α_j, in general, are finite complex values and β_j are finite real nonnegative values.

If desired, the matrix pair (A, B) may be reduced to generalized Schur form. That is, the transformed matrices A and B are upper triangular and the diagonal values of A and B provide α and β.

The parameter **job** specifies two options. If **job** = 'S' then the matrix pair (A, B) is simultaneously reduced to Schur form by applying one unitary transformation (usually called Q) on the left and another (usually called Z) on the right. That is,

$$A \leftarrow Q^{\mathrm{H}} AZ$$
$$B \leftarrow Q^{\mathrm{H}} BZ$$

If **job** = 'E' then at each iteration the same transformations are computed but they are only applied to those parts of A and B which are needed to compute α and β. This option could be used if generalized eigenvalues are required but not generalized eigenvectors.

If **job** = 'S' and **compq** = 'V' or 'I' and **compz** = 'V' or 'I' then the unitary transformations used to reduce the pair (A, B) are accumulated into the input arrays **q** and **z**. If generalized eigenvectors are required then **job** must be set to **job** = 'S' and if left (right) generalized eigenvectors are to be computed then **compq** (**compz**) must be set to **compq** = 'V' or 'I' rather than **compq** = 'N'.

If **compq** = 'I', then eigenvectors are accumulated on the identity matrix and on exit the array **q** contains the left eigenvector matrix Q. However, if **compq** = 'V' then the transformations are accumulated in the user-supplied matrix Q_0 in array **q** on entry and thus on exit **q** contains the matrix product QQ_0. A similar convention is used for **compz**.

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

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

Moler C B and Stewart G W 1973 An algorithm for generalized matrix eigenproblems *SIAM J. Numer. Anal.* **10** 241–256

Stewart G W and Sun J-G 1990 *Matrix Perturbation Theory* Academic Press, London

5 Parameters

5.1 Compulsory Input Parameters

1: **job – string**

Specifies the operations to be performed on (A, B).

job = 'E'

The matrix pair (A, B) on exit might not be in the generalized Schur form.

job = 'S'

The matrix pair (A, B) on exit will be in the generalized Schur form.

Constraint: **job** = 'E' or 'S'.

2: **compq – string**

Specifies the operations to be performed on Q:

compq = 'N'

The array **q** is unchanged.

compq = 'V'

The left transformation Q is accumulated on the array **q**.

compq = 'I'

The array **q** is initialized to the identity matrix before the left transformation Q is accumulated in **q**.

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

3: **compz – string**

Specifies the operations to be performed on Z.

compz = 'N'

The array **z** is unchanged.

compz = 'V'

The right transformation Z is accumulated on the array **z**.

compz = 'I'

The array **z** is initialized to the identity matrix before the right transformation Z is accumulated in **z**.

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

4: **ilo – int32 scalar**

5: **ihi – int32 scalar**

The indices i_{lo} and i_{hi}, respectively which define the upper triangular parts of A. The submatrices $A(1 : i_{\mathrm{lo}} - 1, 1 : i_{\mathrm{lo}} - 1)$ and $A(i_{\mathrm{hi}} + 1 : n, i_{\mathrm{hi}} + 1 : n)$ are then upper triangular. These parameters are provided by f08wv if the matrix pair was previously balanced; otherwise, **ilo** $= 1$ and **ihi** $=$ **n**.

Constraints:

if $\mathbf{n} > 0$, $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{n}$;
if $\mathbf{n} = 0$, $\mathbf{ilo} = 1$ and $\mathbf{ihi} = 0$.

6: **a(lda,$*$) – complex array**

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

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

The n by n upper Hessenberg matrix A. The elements below the first subdiagonal must be set to zero.

7: **b(ldb,$*$) – complex array**

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

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

The n by n upper triangular matrix B. The elements below the diagonal must be zero.

8: **q(ldq,$*$) – complex array**

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

if $\mathbf{compq} =$ 'V' or 'I', $\mathbf{ldq} \geq \mathbf{n}$;
if $\mathbf{compq} =$ 'N', $\mathbf{ldq} \geq 1$.

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

If $\mathbf{compq} =$ 'V', the matrix Q_0 is usually the matrix Q returned by f08ns.

If $\mathbf{compq} =$ 'N', **q** is not referenced.

9: **z(ldz,$*$) – complex array**

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

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

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

If $\mathbf{compz} =$ 'V', the matrix Z_0. Usually, Z_0 is the matrix Z returned by f08ws.

If $\mathbf{compz} =$ 'N', **z** is not referenced.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The second dimension of the arrays **a**, **b**, **q**, **z** and the first dimension of the arrays **a**, **b**, **q**, **z**. (An error is raised if these dimensions are not equal.)

n, the order of the matrices A, B, Q and Z.

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

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, ldq, ldz, work, lwork, rwork

5.4 Output Parameters

1: **a(lda,$*$) – complex array**

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

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

If **job** $=$ 'S', the matrix pair (A, B) will be simultaneously reduced to generalized Schur form.

If **job** $=$ 'E', the 1 by 1 and 2 by 2 diagonal blocks of the matrix pair (A, B) will give generalized eigenvalues but the remaining elements will be irrelevant.

2: **b(ldb,$*$) – complex array**

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

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

If **job** $=$ 'S', the matrix pair (A, B) will be simultaneously reduced to generalized Schur form.

If **job** $=$ 'E', the 1 by 1 and 2 by 2 diagonal blocks of the matrix pair (A, B) will give generalized eigenvalues but the remaining elements will be irrelevant.

3: **alpha($*$) – complex array**

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

α_j, for $j = 1, \ldots, n$.

4: **beta($*$) – complex array**

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

β_j, for $j = 1, \ldots, n$.

5: **q(ldq,$*$) – complex array**

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

if **compq** $=$ 'V' or 'I', **ldq** $\geq$ **n**;
if **compq** $=$ 'N', **ldq** ≥ 1.

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

If **compq** $=$ 'V', **q** contains the matrix product QQ_0.

If **compq** $=$ 'I', **q** contains the transformation matrix Q.

6: **z(ldz,$*$) – complex array**

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

if **compz** $=$ 'V' or 'I', **ldz** $\geq$ **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** contains the matrix product ZZ_0.

If **compz** $=$ 'I', **z** contains the transformation matrix Z.

7: **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: **job**, 2: **compq**, 3: **compz**, 4: **n**, 5: **ilo**, 6: **ihi**, 7: **a**, 8: **lda**, 9: **b**, 10: **ldb**, 11: **alpha**, 12: **beta**, 13: **q**, 14: **ldq**, 15: **z**, 16: **ldz**, 17: **work**, 18: **lwork**, 19: **rwork**, 20: **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 $1 \leq$ **info** $\leq$ **n**, the QZ iteration did not converge and the matrix pair (A, B) is not in the generalized Schur form at exit. However, if **info** $<$ **n**, then the computed α_i and β_i should be correct for $i =$ **info** $+ 1, \ldots,$ **n**.

If **n** $+ 1 \leq$ **info** $\leq 2 \times$ **n**, the computation of shifts failed and the matrix pair (A, B) is not in the generalized Schur form at exit. However, if **info** $< 2 \times$ **n**, then the computed α_i and β_i should be correct for $i =$ **info** $-$ **n** $+ 1, \ldots,$ **n**.

If **info** $> 2 \times$ **n**, then an unexpected Library error has occurred. Please contact NAG with details of your program.

7 Accuracy

Please consult Section 4.11 of the LAPACK Users' Guide (see Anderson *et al.* 1999) and Chapter 6 of Stewart and Sun 1990, for more information.

8 Further Comments

f08xs is the fifth step in the solution of the complex generalized eigenvalue problem and is called after f08ws.

The number of floating-point operations taken by this function is proportional to n^3.

The real analogue of this function is f08xe.

9 Example

```
job = 'E';
compq = 'N';
compz = 'N';
ilo = int32(1);
ihi = int32(4);
a    =    [complex(-2.867890104378767,    -1.594524360587801),    complex(-
0.809337098604556, ...
                 -0.3276607283529636),    complex(-4.900373446187322,    -
0.9865105961392747), ...
     complex(-0.04834623303205134, +1.162636735910666);
            complex(-2.671939461604618,  +0.5945064559939077),  complex(-
0.7895240421486864, ...
                +0.04903482075256903),    complex(-4.954929775736532,    -
0.1634387045312732), ...
     complex(-0.4386325532444865, -0.5739313215365673);
     complex(0, +0), complex(-0.09825782595897961, -0.01149417965898412),
...
           complex(-1.167669110453865,   -0.1365936851015428),   complex(-
1.756232676852781, -0.2054437263770907);
```

```
          complex(0,  +0),  complex(0,  +0),  complex(0.08729329881919053,
+0.003819531014388017), ...
     complex(0.03170217359735389, +0.001387133226909417)];
b   =   [complex(-1.774823934929885,   +0),   complex(-0.7210490086119171,
+0.04290055077107185), ...
            complex(-5.020721861715561,   +1.189845102979979),   complex(-
0.1450254390211963, +0.7257437885879333);
      complex(0, +0), complex(-0.2176281526219798, +0.03516041416104906),
...
           complex(-2.541102900685018,   -0.1458063680541886),   complex(-
0.8228500482725508, -0.4184333588843852);
         complex(0,  +0),  complex(0,  +0),  complex(-1.395782135347712,  -
0.1632782984144689), ...
     complex(-1.747484189780436, -0.2044203302132494);
          complex(0,  +0),  complex(0,  +0),  complex(0,  +0),  complex(-
0.09963146114886638, -0.004359389105629694)];
q = [complex(0, +0)];
z = [complex(0, +0)];
[aOut, bOut, alpha, beta, qOut, zOut, info] = ...
    f08xs(job, compq, compz, ilo, ihi, a, b, q, z)
```

```
aOut =
   -0.1438 + 0.3741i  -1.7000 + 0.1473i  -1.4170 - 1.0916i  -6.6330 -
0.8784i
         0             0.2824 + 0.5209i  -1.5812 + 0.5647i   1.9829 -
3.4956i
         0                  0             0.1522 - 0.2800i   1.1701 -
1.7439i
         0                  0                  0             0.8591 -
0.0636i
bOut =
    0.2263             0.2959 + 0.6808i  -0.9706 + 0.3572i  -3.4142 +
1.1586i
         0            0.5723             -1.7815 + 0.5004i   1.7329 -
3.4482i
         0                  0            0.3323              1.7834 -
2.2524i
        0                 0                 0            1.2738
alpha =
  -0.1438 + 0.3741i
   0.2824 + 0.5209i
   0.1522 - 0.2800i
   0.8591 - 0.0636i
beta =
    0.2263
    0.5723
    0.3323
    1.2738
qOut =
     0
zOut =
     0
info =
           0
```