NAG Library Routine Document

F08XSF (ZHGEQZ)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F08XSF (ZHGEQZ) 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 Specification

```
SUBROUTINE F08XSF (JOB, COMPQ, COMPZ, N, ILO, IHI, A, LDA, B, LDB,          &
                   ALPHA, BETA, Q, LDQ, Z, LDZ, WORK, LWORK, RWORK,         &
                   INFO)

INTEGER                 N, ILO, IHI, LDA, LDB, LDQ, LDZ, LWORK, INFO
REAL (KIND=nag_wp)      RWORK(N)
COMPLEX (KIND=nag_wp)   A(LDA,*), B(LDB,*), ALPHA(N), BETA(N), Q(LDQ,*),    &
                        Z(LDZ,*), WORK(max(1,LWORK))
CHARACTER(1)            JOB, COMPQ, COMPZ
```

The routine may be called by its LAPACK name ***zhgeqz***.

3 Description

F08XSF (ZHGEQZ) 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 routine F08WSF (ZGGHRD) should be called before invoking F08XSF (ZHGEQZ).

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 routine 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 non-negative 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,

$$A \leftarrow Q^{\mathrm{H}} A Z$$
$$B \leftarrow Q^{\mathrm{H}} B Z$$

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

1: JOB – CHARACTER(1) *Input*

On entry: 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 – CHARACTER(1) *Input*

On entry: 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 – CHARACTER(1) *Input*

On entry: 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: N – INTEGER *Input*

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

Constraint: N ≥ 0.

5: ILO – INTEGER *Input*

6: IHI – INTEGER *Input*

On entry: 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 F08WVF (ZGGBAL) if the matrix pair was previously balanced; otherwise, ILO $= 1$ and IHI $=$ N.

Constraints:

if N > 0, $1 \leq$ ILO $\leq$ IHI $\leq$ N;
if N $= 0$, ILO $= 1$ and IHI $= 0$.

7: A(LDA, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least max(1, N).

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

On exit: 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.

8: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08XSF (ZHGEQZ) is called.

Constraint: LDA $\geq$ max(1, N).

9: B(LDB, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B must be at least max(1, N).

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

On exit: 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.

10: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08XSF (ZHGEQZ) is called.

Constraint: LDB $\geq$ max(1, N).

11: ALPHA(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: α_j, for $j = 1, 2, \ldots, n$.

12: BETA(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: β_j, for $j = 1, 2, \ldots, n$.

13: Q(LDQ, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array Q must be at least max(1, N) if COMPQ = 'V' or 'I' and at least 1 if COMPQ = 'N'.

On entry: if COMPQ = 'V', the matrix Q_0. The matrix Q_0 is usually the matrix Q returned by F08WSF (ZGGHRD).

If COMPQ = 'N', Q is not referenced.

On exit: if COMPQ = 'V', Q contains the matrix product QQ_0.

If COMPQ = 'I', Q contains the transformation matrix Q.

14: LDQ – INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which F08XSF (ZHGEQZ) is called.

Constraints:

if COMPQ = 'V' or 'I', LDQ $\geq$ N;
if COMPQ = 'N', LDQ $\geq$ 1.

15: Z(LDZ, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array Z must be at least max(1, N) if COMPZ = 'V' or 'I' and at least 1 if COMPZ = 'N'.

On entry: if COMPZ = 'V', the matrix Z_0. The matrix Z_0 is usually the matrix Z returned by F08WSF (ZGGHRD).

If COMPZ = 'N', Z is not referenced.

On exit: if COMPZ = 'V', Z contains the matrix product ZZ_0.

If COMPZ = 'I', Z contains the transformation matrix Z.

16: LDZ – INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which F08XSF (ZHGEQZ) is called.

Constraints:

if COMPZ = 'V' or 'I', LDZ $\geq$ N;
if COMPZ = 'N', LDZ $\geq$ 1.

17: WORK(max(1, LWORK)) – COMPLEX (KIND=nag_wp) array *Workspace*

On exit: if INFO = 0, the real part of WORK(1) contains the minimum value of LWORK required for optimal performance.

18: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08XSF (ZHGEQZ) is called.

If LWORK = -1, a workspace query is assumed; the routine only calculates the minimum size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

Constraint: LWORK $\geq$ max(1, N) or LWORK = -1.

19: RWORK(N) – REAL (KIND=nag_wp) array *Workspace*

20: INFO – INTEGER *Output*

On exit: INFO $= 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

INFO < 0

If INFO $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

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 Parallelism and Performance

F08XSF (ZHGEQZ) is not threaded by NAG in any implementation.

F08XSF (ZHGEQZ) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

F08XSF (ZHGEQZ) is the fifth step in the solution of the complex generalized eigenvalue problem and is called after F08WSF (ZGGHRD).

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

The real analogue of this routine is F08XEF (DHGEQZ).

10 Example

This example computes the α and β parameters, which defines the generalized eigenvalues, of the matrix pair (A, B) given by

$$A = \begin{pmatrix} 1.0 + 3.0i & 1.0 + 4.0i & 1.0 + 5.0i & 1.0 + 6.0i \\ 2.0 + 2.0i & 4.0 + 3.0i & 8.0 + 4.0i & 16.0 + 5.0i \\ 3.0 + 1.0i & 9.0 + 2.0i & 27.0 + 3.0i & 81.0 + 4.0i \\ 4.0 + 0.0i & 16.0 + 1.0i & 64.0 + 2.0i & 256.0 + 3.0i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1.0+0.0i & 2.0+1.0i & 3.0+2.0i & 4.0+3.0i \\ 1.0+1.0i & 4.0+2.0i & 9.0+3.0i & 16.0+4.0i \\ 1.0+2.0i & 8.0+3.0i & 27.0+4.0i & 64.0+5.0i \\ 1.0+3.0i & 16.0+4.0i & 81.0+5.0i & 256.0+6.0i \end{pmatrix}.$$

This requires calls to five routines: F08WVF (ZGGBAL) to balance the matrix, F08ASF (ZGEQRF) to perform the QR factorization of B, F08AUF (ZUNMQR) to apply Q to A, F08WSF (ZGGHRD) to reduce the matrix pair to the generalized Hessenberg form and F08XSF (ZHGEQZ) to compute the eigenvalues using the QZ algorithm.

10.1 Program Text

```
    Program f08xsfe

!     F08XSF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: m01daf, m01edf, nag_wp, x04dbf, zgeqrf, zggbal,   &
                             zgghrd, zhgeqz, zunmqr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, ihi, ilo, info, irows,     &
                                          jwork, lda, ldb, ldq, ldz, lwork, n, &
                                          ni
      Character (1)                    :: compq, compz, job
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), alpha(:), b(:,:), beta(:), &
                                            e(:), q(:,:), tau(:), work(:),     &
                                            z(:,:)
      Real (Kind=nag_wp), Allocatable  :: emod(:), lscale(:), rscale(:),       &
                                          rwork(:)
      Integer, Allocatable             :: irank(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, aimag, nint, real
!     .. Executable Statements ..
      Write (nout,*) 'F08XSF Example Program Results'
      Flush (nout)

!     Skip heading in data file

      Read (nin,*)
      Read (nin,*) n
      ldq = 1
      ldz = 1
      lda = n
      ldb = n
      lwork = 6*n
      Allocate (a(lda,n),alpha(n),b(ldb,n),beta(n),q(ldq,ldq),tau(n), &
        work(lwork),z(ldz,ldz),lscale(n),rscale(n),rwork(6*n))

!     READ matrix A from data file
      Read (nin,*)(a(i,1:n),i=1,n)

!     READ matrix B from data file
      Read (nin,*)(b(i,1:n),i=1,n)

!     Balance matrix pair (A,B)
      job = 'B'
      Call zggbal(job,n,a,lda,b,ldb,ilo,ihi,lscale,rscale,rwork,info)

!     Matrix A after balancing
```

```
!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call x04dbf('General',' ',n,n,a,lda,'Bracketed','F7.4', &
        'Matrix A after balancing','Integer',rlabs,'Integer',clabs,80,0,ifail)

      Write (nout,*)
      Flush (nout)

!     Matrix B after balancing

      ifail = 0
      Call x04dbf('General',' ',n,n,b,ldb,'Bracketed','F7.4', &
        'Matrix B after balancing','Integer',rlabs,'Integer',clabs,80,0,ifail)

      Write (nout,*)
      Flush (nout)

!     Reduce B to triangular form using QR
      irows = ihi + 1 - ilo

!     The NAG name equivalent of zgeqrf is f08asf
      Call zgeqrf(irows,irows,b(ilo,ilo),ldb,tau,work,lwork,info)

!     Apply the orthogonal transformation to A
!     The NAG name equivalent of zunmqr is f08auf
      Call zunmqr('L','C',irows,irows,irows,b(ilo,ilo),ldb,tau,a(ilo,ilo),lda, &
        work,lwork,info)

!     Compute the generalized Hessenberg form of (A,B) -> (H,T)
      compq = 'N'
      compz = 'N'

!     The NAG name equivalent of zgghrd is f08wsf
      Call zgghrd(compq,compz,irows,1,irows,a(ilo,ilo),lda,b(ilo,ilo),ldb,q, &
        ldq,z,ldz,info)

!     Matrix A (H) in generalized Hessenberg form
      ifail = 0
      Call x04dbf('General',' ',n,n,a,lda,'Bracketed','F7.3', &
        'Matrix A in Hessenberg form','Integer',rlabs,'Integer',clabs,80,0, &
        ifail)

      Write (nout,*)
      Flush (nout)

!     Matrix B (T) in generalized Hessenberg form
      ifail = 0
      Call x04dbf('General',' ',n,n,b,ldb,'Bracketed','F7.3', &
        'Matrix B is triangular','Integer',rlabs,'Integer',clabs,80,0,ifail)

!     Routine ZHGEQZ
!     Workspace query: jwork = -1

      jwork = -1
      job = 'E'
!     The NAG name equivalent of zhgeqz is f08xsf
      Call zhgeqz(job,compq,compz,n,ilo,ihi,a,lda,b,ldb,alpha,beta,q,ldq,z, &
        ldz,work,jwork,rwork,info)
      Write (nout,*)
      Write (nout,99999) nint(real(work(1)))
      Write (nout,99998) lwork
      Write (nout,*)
      Write (nout,99997)
      Write (nout,*)
      Flush (nout)

!     Compute the generalized Schur form
!     if the workspace lwork is adequate
```

```
      If (nint(real(work(1)))<=lwork) Then

!       The NAG name equivalent of zhgeqz is f08xsf
        Call zhgeqz(job,compq,compz,n,ilo,ihi,a,lda,b,ldb,alpha,beta,q,ldq,z, &
          ldz,work,lwork,rwork,info)

!       Print the generalized eigenvalues in descending size order
!       Note: the actual values of beta are real and non-negative

!       Calculate the moduli of the finite eigenvalues.
        Allocate (e(n),emod(n),irank(n))
        ni = 0
        Do i = 1, n
          If (real(beta(i))/=0.0_nag_wp) Then
            ni = ni + 1
            e(ni) = alpha(i)/beta(i)
            emod(ni) = abs(e(ni))
          Else
            Write (nout,99996) i
          End If
        End Do

!       Rearrange the finite eigenvalues in descending order of modulus.
        ifail = 0
        Call m01daf(emod,1,ni,'Descending',irank,ifail)
        ifail = 0
        Call m01edf(e,1,ni,irank,ifail)

        Write (nout,99995)(i,'(',real(e(i)),',',aimag(e(i)),')',i=1,ni)
      Else
        Write (nout,99994)
      End If

99999 Format (1X,'Minimal required LWORK = ',I6)
99998 Format (1X,'Actual value of  LWORK = ',I6)
99997 Format (1X,'Generalized eigenvalues')
99996 Format (1X,I4,5X,'Infinite eigenvalue')
99995 Format (1X,I4,5X,A,F7.3,A,F7.3,A)
99994 Format (1X,'Insufficient workspace allocated for call to F08XSF/ZHGEQZ')
    End Program f08xsfe
```

10.2 Program Data

```
F08XSF Example Program Data
   4                                                    : n

(  1.0, 3.0)  (  1.0, 4.0)  (  1.0, 5.0)  (  1.0, 6.0)
(  2.0, 2.0)  (  4.0, 3.0)  (  8.0, 4.0)  ( 16.0, 5.0)
(  3.0, 1.0)  (  9.0, 2.0)  ( 27.0, 3.0)  ( 81.0, 4.0)
(  4.0, 0.0)  ( 16.0, 1.0)  ( 64.0, 2.0)  (256.0, 3.0)  : A

(  1.0, 0.0)  (  2.0, 1.0)  (  3.0, 2.0)  (  4.0, 3.0)
(  1.0, 1.0)  (  4.0, 2.0)  (  9.0, 3.0)  ( 16.0, 4.0)
(  1.0, 2.0)  (  8.0, 3.0)  ( 27.0, 4.0)  ( 64.0, 5.0)
(  1.0, 3.0)  ( 16.0, 4.0)  ( 81.0, 5.0)  (256.0, 6.0)  : B
```

10.3 Program Results

```
 F08XSF Example Program Results
 Matrix A after balancing
                     1                  2                  3                  4
 1  ( 1.0000, 3.0000) ( 1.0000, 4.0000) ( 0.1000, 0.5000) ( 0.1000, 0.6000)
 2  ( 2.0000, 2.0000) ( 4.0000, 3.0000) ( 0.8000, 0.4000) ( 1.6000, 0.5000)
 3  ( 0.3000, 0.1000) ( 0.9000, 0.2000) ( 0.2700, 0.0300) ( 0.8100, 0.0400)
 4  ( 0.4000, 0.0000) ( 1.6000, 0.1000) ( 0.6400, 0.0200) ( 2.5600, 0.0300)

 Matrix B after balancing
                     1                  2                  3                  4
 1  ( 1.0000, 0.0000) ( 2.0000, 1.0000) ( 0.3000, 0.2000) ( 0.4000, 0.3000)
```

```
2  ( 1.0000, 1.0000) ( 4.0000, 2.0000) ( 0.9000, 0.3000) ( 1.6000, 0.4000)
3  ( 0.1000, 0.2000) ( 0.8000, 0.3000) ( 0.2700, 0.0400) ( 0.6400, 0.0500)
4  ( 0.1000, 0.3000) ( 1.6000, 0.4000) ( 0.8100, 0.0500) ( 2.5600, 0.0600)

Matrix A in Hessenberg form
                    1                 2                 3                 4
1  ( -2.868, -1.595) ( -0.809, -0.328) ( -4.900, -0.987) ( -0.048,  1.163)
2  ( -2.672,  0.595) ( -0.790,  0.049) ( -4.955, -0.163) ( -0.439, -0.574)
3  (  0.000,  0.000) ( -0.098, -0.011) ( -1.168, -0.137) ( -1.756, -0.205)
4  (  0.000,  0.000) (  0.000,  0.000) (  0.087,  0.004) (  0.032,  0.001)

Matrix B is triangular
                    1                 2                 3                 4
1  ( -1.775,  0.000) ( -0.721,  0.043) ( -5.021,  1.190) ( -0.145,  0.726)
2  (  0.000,  0.000) ( -0.218,  0.035) ( -2.541, -0.146) ( -0.823, -0.418)
3  (  0.000,  0.000) (  0.000,  0.000) ( -1.396, -0.163) ( -1.747, -0.204)
4  (  0.000,  0.000) (  0.000,  0.000) (  0.000,  0.000) ( -0.100, -0.004)

Minimal required LWORK =      4
Actual value of  LWORK =     24

Generalized eigenvalues

   1     ( -0.635,  1.653)
   2     (  0.493,  0.910)
   3     (  0.458, -0.843)
   4     (  0.674, -0.050)
```