# NAG Library Routine Document

# F08XNF (ZGGES)

**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

F08XNF (ZGGES) 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 Specification

```
SUBROUTINE F08XNF (JOBVSL, JOBVSR, SORT, SELCTG, N, A, LDA, B, LDB,         &
                   SDIM, ALPHA, BETA, VSL, LDVSL, VSR, LDVSR, WORK,         &
                   LWORK, RWORK, BWORK, INFO)

INTEGER                 N, LDA, LDB, SDIM, LDVSL, LDVSR, LWORK, INFO
REAL (KIND=nag_wp)      RWORK(max(1,8*N))
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), ALPHA(N), BETA(N),               &
                        VSL(LDVSL,*), VSR(LDVSR,*), WORK(max(1,LWORK))
LOGICAL                 SELCTG, BWORK(*)
CHARACTER(1)            JOBVSL, JOBVSR, SORT
EXTERNAL                SELCTG
```

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

## 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, $\lambda$, of $(A,B)$ are computed from the diagonals of $T$ and $S$ and satisfy

$$Az = \lambda Bz,$$

where $z$ is the corresponding generalized eigenvector. $\lambda$ is actually returned as the pair $(\alpha,\beta)$ such that

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

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

Optionally, F08XNF (ZGGES) 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.

F08XNF (ZGGES) 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

1: JOBVSL – CHARACTER(1) *Input*

*On entry*: 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) *Input*

*On entry*: 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) *Input*

*On entry*: 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. *External Procedure*

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 F08XNF (ZGGES), and may be called with the dummy function F08XNZ.

The specification of SELCTG is:

```
FUNCTION SELCTG (A, B)
LOGICAL SELCTG
COMPLEX (KIND=nag_wp) A, B
```

1: A – COMPLEX (KIND=nag_wp) *Input*
2: B – COMPLEX (KIND=nag_wp) *Input*

*On entry*: an eigenvalue $\mathrm{A}(j)/\mathrm{B}(j)$ is selected if $\mathrm{SELCTG}(\mathrm{A}(j),\mathrm{B}(j))$ is .TRUE..

Note that in the ill-conditioned case, a selected generalized eigenvalue may no longer satisfy $\mathrm{SELCTG}(\mathrm{A}(j),\mathrm{B}(j)) =$ .TRUE. after ordering. INFO $= \mathrm{N}+2$ in this case.

SELCTG must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which F08XNF (ZGGES) is called. Parameters denoted as *Input* must **not** be changed by this procedure.

5: N – INTEGER *Input*

*On entry*: $n$, the order of the matrices $A$ and $B$.

*Constraint*: $\mathrm{N} \geq 0$.

6: $\mathrm{A}(\mathrm{LDA},*)$ – COMPLEX (KIND=nag_wp) array *Input/Output*

**Note**: the second dimension of the array A must be at least $\max(1,\mathrm{N})$.

*On entry*: the first of the pair of matrices, $A$.

*On exit*: A has been overwritten by its generalized Schur form $S$.

7: LDA – INTEGER *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08XNF (ZGGES) is called.

*Constraint*: $\text{LDA} \geq \max(1, \text{N})$.

8: $\text{B}(\text{LDB}, *)$ – COMPLEX (KIND=nag_wp) array *Input/Output*

**Note**: the second dimension of the array B must be at least $\max(1, \text{N})$.

*On entry*: the second of the pair of matrices, $B$.

*On exit*: B has been overwritten by its generalized Schur form $T$.

9: LDB – INTEGER *Input*

*On entry*: the first dimension of the array B as declared in the (sub)program from which F08XNF (ZGGES) is called.

*Constraint*: $\text{LDB} \geq \max(1, \text{N})$.

10: SDIM – INTEGER *Output*

*On exit*: if SORT = 'N', SDIM = 0.

If SORT = 'S', SDIM = number of eigenvalues (after sorting) for which SELCTG is .TRUE..

11: $\text{ALPHA}(\text{N})$ – COMPLEX (KIND=nag_wp) array *Output*

*On exit*: see the description of BETA.

12: $\text{BETA}(\text{N})$ – COMPLEX (KIND=nag_wp) array *Output*

*On exit*: $\text{ALPHA}(j)/\text{BETA}(j)$, for $j = 1, 2, \ldots, \text{N}$, will be the generalized eigenvalues. $\text{ALPHA}(j)$, for $j = 1, 2, \ldots, \text{N}$ and $\text{BETA}(j)$, for $j = 1, 2, \ldots, \text{N}$, are the diagonals of the complex Schur form $(A, B)$ output by F08XNF (ZGGES). The $\text{BETA}(j)$ will be non-negative real.

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

13: $\text{VSL}(\text{LDVSL}, *)$ – COMPLEX (KIND=nag_wp) array *Output*

**Note**: the second dimension of the array VSL must be at least $\max(1, \text{N})$ if JOBVSL = 'V', and at least 1 otherwise.

*On exit*: if JOBVSL = 'V', VSL will contain the left Schur vectors, $Q$.

If JOBVSL = 'N', VSL is not referenced.

14: LDVSL – INTEGER *Input*

*On entry*: the first dimension of the array VSL as declared in the (sub)program from which F08XNF (ZGGES) is called.

*Constraints*:

if JOBVSL = 'V', $\text{LDVSL} \geq \max(1, \text{N})$;
otherwise $\text{LDVSL} \geq 1$.

15: VSR(LDVSR, $*$) – COMPLEX (KIND=nag_wp) array *Output*

**Note**: the second dimension of the array VSR must be at least $\max(1, \mathrm{N})$ if JOBVSR = 'V', and at least 1 otherwise.

*On exit*: if JOBVSR = 'V', VSR will contain the right Schur vectors, $Z$.

If JOBVSR = 'N', VSR is not referenced.

16: LDVSR – INTEGER *Input*

*On entry*: the first dimension of the array VSR as declared in the (sub)program from which F08XNF (ZGGES) is called.

*Constraints*:

if JOBVSR = 'V', LDVSR $\geq \max(1, \mathrm{N})$;
otherwise LDVSR $\geq 1$.

17: WORK($\max(1, \mathrm{LWORK})$) – 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 F08XNF (ZGGES) is called.

If LWORK $= -1$, a workspace query is assumed; the routine only calculates the optimal 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.

*Suggested value*: for optimal performance, LWORK must generally be larger than the minimum, say $2 \times \mathrm{N} + nb \times \mathrm{N}$, where $nb$ is the optimal ***block size*** for F08NSF (ZGEHRD).

*Constraint*: LWORK $\geq \max(1, 2 \times \mathrm{N})$.

19: RWORK($\max(1, 8 \times \mathrm{N})$) – REAL (KIND=nag_wp) array *Workspace*

20: BWORK($*$) – LOGICAL array *Workspace*

**Note**: the dimension of the array BWORK must be at least 1 if SORT = 'N', and at least $\max(1, \mathrm{N})$ otherwise.

If SORT = 'N', BWORK is not referenced.

21: 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 $= 1$ to N

The $QZ$ iteration failed. $(A, B)$ are not in Schur form, but ALPHA($j$) and BETA($j$) should be correct for $j = \mathrm{INFO} + 1, \ldots, \mathrm{N}$.

INFO $= \mathrm{N} + 1$

Unexpected error returned from F08XSF (ZHGEQZ).

INFO $= \mathrm{N} + 2$

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 $= \mathrm{N} + 3$

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 $\epsilon$ is the ***machine precision***. See Section 4.11 of Anderson *et al.* (1999) for further details.

# 8 Parallelism and Performance

F08XNF (ZGGES) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08XNF (ZGGES) 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

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

The real analogue of this routine is F08XAF (DGGES).

# 10 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.

## 10.1 Program Text

```
    Program f08xnfe

!     F08XNF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f08xnz, nag_wp, x02ajf, x04dbf, zgemm, zgges,     &
                             zlange => f06uaf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nb = 64, nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)           :: alph, bet
      Real (Kind=nag_wp)              :: normd, norme
      Integer                         :: i, ifail, info, lda, ldb, ldc, ldd,  &
                                         lde, ldvsl, ldvsr, lwork, n, sdim
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), alpha(:), b(:,:), beta(:), &
                                           c(:,:), d(:,:), e(:,:), vsl(:,:),  &
                                           vsr(:,:), work(:)
      Complex (Kind=nag_wp)           :: wdum(1)
      Real (Kind=nag_wp), Allocatable  :: rwork(:)
      Logical, Allocatable            :: bwork(:)
      Character (1)                   :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: cmplx, max, nint, real
!     .. Executable Statements ..
      Write (nout,*) 'F08XNF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      lda = n
      ldb = n
      ldc = n
      ldd = n
      lde = n
      ldvsl = n
      ldvsr = n
      Allocate (a(lda,n),alpha(n),b(ldb,n),beta(n),c(ldc,n),d(ldd,n),e(lde,n), &
        vsl(ldvsl,n),vsr(ldvsr,n),rwork(8*n),bwork(n))

!     Use routine workspace query to get optimal workspace.
      lwork = -1
!     The NAG name equivalent of zgges is f08xnf
      Call zgges('Vectors (left)','Vectors (right)','No sort',f08xnz,n,a,lda, &
        b,ldb,sdim,alpha,beta,vsl,ldvsl,vsr,ldvsr,wdum,lwork,rwork,bwork,info)

!     Make sure that there is enough workspace for blocksize nb.
      lwork = max((nb+1)*n,nint(real(wdum(1))))
      Allocate (work(lwork))

!     Read in the matrices A and B
      Read (nin,*)(a(i,1:n),i=1,n)
      Read (nin,*)(b(i,1:n),i=1,n)

!     Copy A and B into D and E respectively
      d(1:n,1:n) = a(1:n,1:n)
      e(1:n,1:n) = b(1:n,1:n)

!     Print matrices A and B
!     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','F8.4','Matrix A', &
        'Integer',rlabs,'Integer',clabs,80,0,ifail)
```

```
      Write (nout,*)
      Flush (nout)

      ifail = 0
      Call x04dbf('General',' ',n,n,b,ldb,'Bracketed','F8.4','Matrix B', &
        'Integer',rlabs,'Integer',clabs,80,0,ifail)
      Write (nout,*)
      Flush (nout)

!     Find the generalized Schur form
!     The NAG name equivalent of zgges is f08xnf
      Call zgges('Vectors (left)','Vectors (right)','No sort',f08xnz,n,a,lda, &
        b,ldb,sdim,alpha,beta,vsl,ldvsl,vsr,ldvsr,work,lwork,rwork,bwork,info)

      If (info>0) Then
        Write (nout,99999) 'Failure in ZGGES. INFO =', info
      Else

!       Compute A - Q*S*Z^H from the factorization of (A,B) and store in
!       matrix D
!       The NAG name equivelent of zgemm is f06zaf
        alph = cmplx(1,kind=nag_wp)
        bet = cmplx(0,kind=nag_wp)
        Call zgemm('N','N',n,n,n,alph,vsl,ldvsl,a,lda,bet,c,ldc)
        alph = cmplx(-1,kind=nag_wp)
        bet = cmplx(1,kind=nag_wp)
        Call zgemm('N','C',n,n,n,alph,c,ldc,vsr,ldvsr,bet,d,ldd)

!       Compute B - Q*T*Z^H from the factorization of (A,B) and store in
!       matrix E
        alph = cmplx(1,kind=nag_wp)
        bet = cmplx(0,kind=nag_wp)
        Call zgemm('N','N',n,n,n,alph,vsl,ldvsl,b,ldb,bet,c,ldc)
        alph = cmplx(-1,kind=nag_wp)
        bet = cmplx(1,kind=nag_wp)
        Call zgemm('N','C',n,n,n,alph,c,ldc,vsr,ldvsr,bet,e,lde)

!       Find norms of matrices D and E and warn if either is too large
!       f06uaf is the NAG name equivalent of the LAPACK auxiliary zlange
        normd = zlange('O',ldd,n,d,ldd,rwork)
        norme = zlange('O',lde,n,e,lde,rwork)
        If (normd>x02ajf()**0.75_nag_wp .Or. norme>x02ajf()**0.75_nag_wp) Then
          Write (nout,*) 'Norm of A-(Q*S*Z^H) or norm of B-(Q*T*Z^H) &
            &is much greater than 0.'
          Write (nout,*) 'Schur factorization has failed.'
        Else
!         Print generalized eigenvalues
          Write (nout,*) 'Generalized Eigenvalues'

          Do i = 1, n
            If (beta(i)/=0.0_nag_wp) Then
              Write (nout,99998) i, alpha(i)/beta(i)
            Else
              Write (nout,99997) i
            End If
          End Do
        End If
      End If

99999 Format (1X,A,I4)
99998 Format (1X,I2,1X,'(',1P,E11.4,',',E11.4,')')
99997 Format (1X,I4,'Eigenvalue is infinite')
    End Program f08xnfe
```

## 10.2 Program Data

```
F08XNF Example Program Data
  4                                                             : Value of N
  (-21.10,-22.50) ( 53.50,-50.50) (-34.50,127.50) (  7.50,  0.50)
  ( -0.46, -7.78) ( -3.50,-37.50) (-15.50, 58.50) (-10.50, -1.50)
  (  4.30, -5.50) ( 39.70,-17.10) (-68.50, 12.50) ( -7.50, -3.50)
  (  5.50,  4.40) ( 14.40, 43.30) (-32.50,-46.00) (-19.00,-32.50) : End of A
  (  1.00, -5.00) (  1.60,  1.20) ( -3.00,  0.00) (  0.00, -1.00)
  (  0.80, -0.60) (  3.00, -5.00) ( -4.00,  3.00) ( -2.40, -3.20)
  (  1.00,  0.00) (  2.40,  1.80) ( -4.00, -5.00) (  0.00, -3.00)
  (  0.00,  1.00) ( -1.80,  2.40) (  0.00, -4.00) (  4.00, -5.00) : End of B
```

## 10.3 Program Results

```
 F08XNF Example Program Results

 Matrix A
                     1                    2                    3
 1  (-21.1000,-22.5000) ( 53.5000,-50.5000) (-34.5000,127.5000)
 2  ( -0.4600, -7.7800) ( -3.5000,-37.5000) (-15.5000, 58.5000)
 3  (  4.3000, -5.5000) ( 39.7000,-17.1000) (-68.5000, 12.5000)
 4  (  5.5000,  4.4000) ( 14.4000, 43.3000) (-32.5000,-46.0000)

                     4
 1  (  7.5000,  0.5000)
 2  (-10.5000, -1.5000)
 3  ( -7.5000, -3.5000)
 4  (-19.0000,-32.5000)

 Matrix B
                     1                    2                    3
 1  (  1.0000, -5.0000) (  1.6000,  1.2000) ( -3.0000,  0.0000)
 2  (  0.8000, -0.6000) (  3.0000, -5.0000) ( -4.0000,  3.0000)
 3  (  1.0000,  0.0000) (  2.4000,  1.8000) ( -4.0000, -5.0000)
 4  (  0.0000,  1.0000) ( -1.8000,  2.4000) (  0.0000, -4.0000)

                     4
 1  (  0.0000, -1.0000)
 2  ( -2.4000, -3.2000)
 3  (  0.0000, -3.0000)
 4  (  4.0000, -5.0000)

 Generalized Eigenvalues
  1 ( 3.0000E+00,-9.0000E+00)
  2 ( 2.0000E+00,-5.0000E+00)
  3 ( 3.0000E+00,-1.0000E+00)
  4 ( 4.0000E+00,-5.0000E+00)
```