NAG Library Routine Document

F08YFF (DTGEXC)

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

F08YFF (DTGEXC) reorders the generalized Schur factorization of a matrix pair in real generalized Schur form.

2 Specification

```
SUBROUTINE F08YFF (WANTQ, WANTZ, N, A, LDA, B, LDB, Q, LDQ, Z, LDZ,        &
                   IFST, ILST, WORK, LWORK, INFO)

INTEGER             N, LDA, LDB, LDQ, LDZ, IFST, ILST, LWORK, INFO
REAL (KIND=nag_wp) A(LDA,*), B(LDB,*), Q(LDQ,*), Z(LDZ,*),                &
                   WORK(max(1,LWORK))
LOGICAL             WANTQ, WANTZ
```

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

3 Description

F08YFF (DTGEXC) reorders the generalized real n by n matrix pair (S,T) in real generalized Schur form, so that the diagonal element or block of (S,T) with row index i_1 is moved to row i_2, using an orthogonal equivalence transformation. That is, S and T are factorized as

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

where $\left(\hat{S},\hat{T}\right)$ are also in real generalized Schur form.

The pair (S,T) are in real generalized Schur form if S is block upper triangular with 1 by 1 and 2 by 2 diagonal blocks and T is upper triangular as returned, for example, by F08XAF (DGGES), or F08XEF (DHGEQZ) with JOB = 'S'.

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

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

then, optionally, the matrices Q and Z can be updated as $Q\hat{Q}$ and $Z\hat{Z}$.

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 Arguments

1: WANTQ – LOGICAL *Input*

On entry: if WANTQ = .TRUE., update the left transformation matrix Q.

If WANTQ = .FALSE., do not update Q.

2: WANTZ – LOGICAL *Input*

On entry: if WANTZ = .TRUE., update the right transformation matrix Z.

If WANTZ = .FALSE., do not update Z.

3: N – INTEGER *Input*

On entry: n, the order of the matrices S and T.

Constraint: N ≥ 0.

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

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

On entry: the matrix S in the pair (S, T).

On exit: the updated matrix $\hat{S}$.

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08YFF (DTGEXC) is called.

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

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

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

On entry: the matrix T, in the pair (S, T).

On exit: the updated matrix $\hat{T}$

7: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08YFF (DTGEXC) is called.

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

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

Note: the second dimension of the array Q must be at least max(1, N) if WANTQ = .TRUE., and at least 1 otherwise.

On entry: if WANTQ = .TRUE., the orthogonal matrix Q.

On exit: if WANTQ = .TRUE., the updated matrix $Q\hat{Q}$.

If WANTQ = .FALSE., Q is not referenced.

9: LDQ – INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which F08YFF (DTGEXC) is called.

Constraints:

if WANTQ = .TRUE., LDQ $\geq$ max(1, N);
otherwise LDQ ≥ 1.

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

Note: the second dimension of the array Z must be at least max(1, N) if WANTZ = .TRUE., and at least 1 otherwise.

On entry: if WANTZ = .TRUE., the orthogonal matrix Z.

On exit: if WANTZ = .TRUE., the updated matrix $Z\hat{Z}$.

If WANTZ = .FALSE., Z is not referenced.

11: LDZ – INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which F08YFF (DTGEXC) is called.

Constraints:

if WANTZ = .TRUE., LDZ $\geq \max(1, \mathrm{N})$;
otherwise LDZ ≥ 1.

12: IFST – INTEGER *Input/Output*

13: ILST – INTEGER *Input/Output*

On entry: the indices i_1 and i_2 that specify the reordering of the diagonal blocks of (S, T). The block with row index IFST is moved to row ILST, by a sequence of swapping between adjacent blocks.

On exit: if IFST pointed on entry to the second row of a 2 by 2 block, it is changed to point to the first row; ILST always points to the first row of the block in its final position (which may differ from its input value by $+1$ or -1).

Constraint: $1 \leq \mathrm{IFST} \leq \mathrm{N}$ and $1 \leq \mathrm{ILST} \leq \mathrm{N}$.

14: WORK$(\max(1, \mathrm{LWORK}))$ – REAL (KIND=nag_wp) array *Workspace*

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

15: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08YFF (DTGEXC) 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.

Constraints: if LWORK $\neq -1$,

if N ≤ 1, LWORK ≥ 1;
otherwise LWORK $\geq 4 \times \mathrm{N} + 16$.

16: 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 $= 1$

The transformed matrix pair $\left(\hat{S}, \hat{T}\right)$ would be too far from generalized Schur form; the problem is ill-conditioned. (S, T) may have been partially reordered, and ILST points to the first row of the current position of the block being moved.

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.

8 Parallelism and Performance

F08YFF (DTGEXC) 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 complex analogue of this routine is F08YTF (ZTGEXC).

10 Example

This example exchanges blocks 2 and 1 of the matrix pair (S,T), where

$$S = \begin{pmatrix} 4.0 & 1.0 & 1.0 & 2.0 \\ 0 & 3.0 & 4.0 & 1.0 \\ 0 & 1.0 & 3.0 & 1.0 \\ 0 & 0 & 0 & 6.0 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 2.0 & 1.0 & 1.0 & 3.0 \\ 0 & 1.0 & 2.0 & 1.0 \\ 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 2.0 \end{pmatrix}.$$

10.1 Program Text

```
    Program f08yffe

!     F08YFF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: dtgexc, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                :: nin = 5, nout = 6
      Logical, Parameter                :: wantq = .False., wantz = .False.
!     .. Local Scalars ..
      Integer                           :: i, ifail, ifst, ilst, info, lda,     &
                                           ldb, ldq, ldz, lwork, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable   :: a(:,:), b(:,:), q(:,:), work(:),     &
                                           z(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F08YFF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      ldq = 1
      ldz = 1
      lda = n
      ldb = n
```

```
      lwork = 4*n + 16
      Allocate (a(lda,n),b(ldb,n),q(ldq,1),work(lwork),z(ldz,1))

!     Read A and B from data file

      Read (nin,*)(a(i,1:n),i=1,n)
      Read (nin,*)(b(i,1:n),i=1,n)

!     Read the row indices

      Read (nin,*) ifst, ilst

!     Reorder A and B

!     The NAG name equivalent of dtgexc is f08yff
      Call dtgexc(wantq,wantz,n,a,lda,b,ldb,q,ldq,z,ldz,ifst,ilst,work,lwork,  &
        info)

      If (info/=0) Then
        Write (nout,99999) info, ilst
        Write (nout,*)
        Flush (nout)
      End If

!     The resulting reordered Schur matrices can differ by +- signs by
!     multiplying rows and columns of Q and Z by -1. We will normalize here by
!     making the diagonals and last column of B positive.
      Call normalize(a,b)

!     Print reordered generalized Schur form

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call x04caf('General',' ',n,n,a,lda,'Reordered Schur matrix A',ifail)

      Write (nout,*)
      Flush (nout)

      ifail = 0
      Call x04caf('General',' ',n,n,b,ldb,'Reordered Schur matrix B',ifail)

99999 Format (' Reordering could not be completed. INFO = ',I3,' ILST = ',I5)

    Contains
      Subroutine normalize(a,b)

!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: a(lda,n), b(ldb,n)
!       .. Local Scalars ..
        Integer                         :: i, j
!       .. Intrinsic Procedures ..
        Intrinsic                       :: max
!       .. Executable Statements ..

!       Last column of B positive
        Do i = 1, n
          j = max(1,i-1)
          If (b(i,n)<0.0_nag_wp) Then
            a(i,j:n) = -a(i,j:n)
            b(i,i:n) = -b(i,i:n)
          End If
        End Do

!       Diagonals of B positive
        Do i = 1, n - 1
          If (b(i,i)<0.0_nag_wp) Then
            a(1:i+1,i) = -a(1:i+1,i)
```

```
            b(1:i,i) = -b(1:i,i)
          End If
        End Do
      End Subroutine normalize
    End Program f08yffe
```

10.2 Program Data

```
F08YFF Example Program Data
  4                   :Value of N
  4.0  1.0  1.0  2.0
  0.0  3.0  4.0  1.0
  0.0  1.0  3.0  1.0
  0.0  0.0  0.0  6.0   :End of matrix A
  2.0  1.0  1.0  3.0
  0.0  1.0  2.0  1.0
  0.0  0.0  1.0  1.0
  0.0  0.0  0.0  2.0   :End of matrix B
  2  1                :Values of IFST and ILST
```

10.3 Program Results

```
 F08YFF Example Program Results

 Reordered Schur matrix A
             1          2          3          4
 1      4.1926     1.2591     2.5578     0.4520
 2     -0.8712     0.8627     2.7912     1.1383
 3      0.0000     0.0000     4.2426     2.1213
 4      0.0000     0.0000     0.0000     6.0000

 Reordered Schur matrix B
             1          2          3          4
 1      1.7439     0.0000     0.7533     0.0661
 2      0.0000     0.5406     1.8972     1.7308
 3      0.0000     0.0000     2.1213     2.8284
 4      0.0000     0.0000     0.0000     2.0000
```