NAG Library Routine Document

F08KTF (ZUNGBR)

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

F08KTF (ZUNGBR) generates one of the complex unitary matrices Q or P^H which were determined by F08KSF (ZGEBRD) when reducing a complex matrix to bidiagonal form.

2   Specification

```
SUBROUTINE F08KTF (VECT, M, N, K, A, LDA, TAU, WORK, LWORK, INFO)

INTEGER              M, N, K, LDA, LWORK, INFO
COMPLEX (KIND=nag_wp) A(LDA,*), TAU(*), WORK(max(1,LWORK))
CHARACTER(1)         VECT
```

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

3   Description

F08KTF (ZUNGBR) is intended to be used after a call to F08KSF (ZGEBRD), which reduces a complex rectangular matrix A to real bidiagonal form B by a unitary transformation: $A = QBP^H$. F08KSF (ZGEBRD) represents the matrices Q and P^H as products of elementary reflectors.

This routine may be used to generate Q or P^H explicitly as square matrices, or in some cases just the leading columns of Q or the leading rows of P^H.

The various possibilities are specified by the parameters VECT, M, N and K. The appropriate values to cover the most likely cases are as follows (assuming that A was an m by n matrix):

1.   To form the full m by m matrix Q:

        `CALL ZUNGBR('Q',m,m,n,...)`

    (note that the array A must have at least m columns).

2.   If $m > n$, to form the n leading columns of Q:

        `CALL ZUNGBR('Q',m,n,n,...)`

3.   To form the full n by n matrix P^H:

        `CALL ZUNGBR('P',n,n,m,...)`

    (note that the array A must have at least n rows).

4.   If $m < n$, to form the m leading rows of P^H:

        `CALL ZUNGBR('P',m,n,m,...)`

4   References

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

5 Parameters

1: VECT – CHARACTER(1) *Input*

On entry: indicates whether the unitary matrix Q or P^H is generated.

VECT = 'Q'
  Q is generated.

VECT = 'P'
  P^H is generated.

Constraint: VECT = 'Q' or 'P'.

2: M – INTEGER *Input*

On entry: m, the number of rows of the unitary matrix Q or P^H to be returned.

Constraint: $M \geq 0$.

3: N – INTEGER *Input*

On entry: n, the number of columns of the unitary matrix Q or P^H to be returned.

Constraints:

  $N \geq 0$;
  if VECT = 'Q' and $M > K$, $M \geq N \geq K$;
  if VECT = 'Q' and $M \leq K$, $M = N$;
  if VECT = 'P' and $N > K$, $N \geq M \geq K$;
  if VECT = 'P' and $N \leq K$, $N = M$.

4: K – INTEGER *Input*

On entry: if VECT = 'Q', the number of columns in the original matrix A.

If VECT = 'P', the number of rows in the original matrix A.

Constraint: $K \geq 0$.

5: 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: details of the vectors which define the elementary reflectors, as returned by F08KSF (ZGEBRD).

On exit: the unitary matrix Q or P^H, or the leading rows or columns thereof, as specified by VECT, M and N.

6: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08KTF (ZUNGBR) is called.

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

7: TAU(∗) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array TAU must be at least $\max(1, \min(M, K))$ if VECT = 'Q' and at least $\max(1, \min(N, K))$ if VECT = 'P'.

On entry: further details of the elementary reflectors, as returned by F08KSF (ZGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

8:    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.

9:    LWORK – INTEGER   *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08KTF (ZUNGBR) 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 $\geq \min(M, N) \times nb$, where nb is the optimal **block size**.

Constraint: LWORK $\geq \max(1, \min(M, N))$ or LWORK = −1.

10:   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.

7   Accuracy

The computed matrix Q differs from an exactly unitary matrix by a matrix E such that

$$\|E\|_2 = O(\epsilon),$$

where ϵ is the **machine precision**. A similar statement holds for the computed matrix P^{H}.

8   Parallelism and Performance

F08KTF (ZUNGBR) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08KTF (ZUNGBR) 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 real floating-point operations for the cases listed in Section 3 are approximately as follows:

1.   To form the whole of Q:

$\frac{16}{3}n(3m^2 - 3mn + n^2)$ if $m > n$,

$\frac{16}{3}m^3$ if $m \leq n$;

2.   To form the n leading columns of Q when $m > n$:

$\frac{8}{3}n^2(3m - n)$;

3. To form the whole of P^{H}:

$\frac{16}{3}n^3$ if $m \geq n$,

$\frac{16}{3}m^3(3n^2 - 3mn + m^2)$ if $m < n$;

4. To form the m leading rows of P^{H} when $m < n$:

$\frac{8}{3}m^2(3n - m)$.

The real analogue of this routine is F08KFF (DORGBR).

10   Example

For this routine two examples are presented, both of which involve computing the singular value decomposition of a matrix A, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}$$

in the first example and

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}$$

in the second. A must first be reduced to tridiagonal form by F08KSF (ZGEBRD). The program then calls F08KTF (ZUNGBR) twice to form Q and P^{H}, and passes these matrices to F08MSF (ZBDSQR), which computes the singular value decomposition of A.

10.1   Program Text

```
      Program f08ktfe

!     F08KTF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f06tff, nag_wp, x04dbf, zbdsqr, zgebrd, zungbr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ic, ifail, info, lda, ldc, ldu,  &
                                          ldvt, lwork, m, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), c(:,:), taup(:), tauq(:),  &
                                          u(:,:), vt(:,:), work(:)
      Real (Kind=nag_wp), Allocatable  :: d(:), e(:), rwork(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F08KTF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Do ic = 1, 2
        Read (nin,*) m, n
        lda = m
        ldc = n
        ldu = m
```

```
           ldvt = n
           lwork = 64*(m+n)
           Allocate (a(lda,n),c(ldc,n),taup(n),tauq(n),u(ldu,n),vt(ldvt,n), &
             work(lwork),d(n),e(n-1),rwork(4*n-4))

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

!          Reduce A to bidiagonal form
!          The NAG name equivalent of zgebrd is f08ksf
           Call zgebrd(m,n,a,lda,d,e,tauq,taup,work,lwork,info)

           If (m>=n) Then

!             Copy A to VT and U
              Call f06tff('Upper',n,n,a,lda,vt,ldvt)
              Call f06tff('Lower',m,n,a,lda,u,ldu)

!             Form P**H explicitly, storing the result in VT
!             The NAG name equivalent of zungbr is f08ktf
              Call zungbr('P',n,n,m,vt,ldvt,taup,work,lwork,info)

!             Form Q explicitly, storing the result in U
              Call zungbr('Q',m,n,n,u,ldu,tauq,work,lwork,info)

!             Compute the SVD of A
!             The NAG name equivalent of zbdsqr is f08msf
              Call zbdsqr('Upper',n,n,m,0,d,e,vt,ldvt,u,ldu,c,ldc,rwork,info)

!             Print singular values, left & right singular vectors

              Write (nout,*)
              Write (nout,*) 'Example 1: singular values'
              Write (nout,99999) d(1:n)
              Write (nout,*)
              Flush (nout)

!             ifail: behaviour on error exit
!                    =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
              ifail = 0
              Call x04dbf('General',' ',n,n,vt,ldvt,'Bracketed','F7.4', &
                'Example 1: right singular vectors, by row','Integer',rlabs, &
                'Integer',clabs,80,0,ifail)

              Write (nout,*)
              Flush (nout)

              Call x04dbf('General',' ',m,n,u,ldu,'Bracketed','F7.4', &
                'Example 1: left singular vectors, by column','Integer',rlabs, &
                'Integer',clabs,80,0,ifail)

           Else

!             Copy A to VT and U
              Call f06tff('Upper',m,n,a,lda,vt,ldvt)
              Call f06tff('Lower',m,m,a,lda,u,ldu)

!             Form P**H explicitly, storing the result in VT
!             The NAG name equivalent of zungbr is f08ktf
              Call zungbr('P',m,n,m,vt,ldvt,taup,work,lwork,info)

!             Form Q explicitly, storing the result in U
              Call zungbr('Q',m,m,n,u,ldu,tauq,work,lwork,info)

!             Compute the SVD of A
!             The NAG name equivalent of zbdsqr is f08msf
              Call zbdsqr('Lower',m,n,m,0,d,e,vt,ldvt,u,ldu,c,ldc,rwork,info)

!             Print singular values, left & right singular vectors

              Write (nout,*)
```

```
        Write (nout,*) 'Example 2: singular values'
        Write (nout,99999) d(1:m)
        Write (nout,*)
        Flush (nout)

        ifail = 0
        Call x04dbf('General',' ',m,n,vt,ldvt,'Bracketed','F7.4', &
          'Example 2: right singular vectors, by row','Integer',rlabs, &
          'Integer',clabs,80,0,ifail)

        Write (nout,*)
        Flush (nout)

        Call x04dbf('General',' ',m,m,u,ldu,'Bracketed','F7.4', &
          'Example 2: left singular vectors, by column','Integer',rlabs, &
          'Integer',clabs,80,0,ifail)

      End If
      Deallocate (a,c,taup,tauq,u,vt,work,d,e,rwork)
    End Do

99999 Format (8X,4(F7.4,11X:))
    End Program f08ktfe
```

10.2 Program Data

```
F08KTF Example Program Data
  6  4                                        :Values of M and N, Example 1
 ( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
 (-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
 ( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
 ( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)   :End of matrix A
  3  4                                        :Values of M and N, Example 2
 ( 0.28,-0.36) ( 0.50,-0.86) (-0.77,-0.48) ( 1.58, 0.66)
 (-0.50,-1.10) (-1.21, 0.76) (-0.32,-0.24) (-0.27,-1.15)
 ( 0.36,-0.51) (-0.07, 1.33) (-0.75, 0.47) (-0.08, 1.01)   :End of matrix A
```

10.3 Program Results

```
F08KTF Example Program Results

Example 1: singular values
        3.9994          3.0003          1.9944          0.9995

Example 1: right singular vectors, by row
                  1               2               3               4
1 (-0.6971,-0.0000) (-0.0867,-0.3548) ( 0.0560,-0.5400) (-0.1878,-0.2253)
2 ( 0.2403, 0.0000) ( 0.0725,-0.2336) (-0.2477,-0.5291) ( 0.7026, 0.2177)
3 (-0.5123, 0.0000) (-0.3030,-0.1735) ( 0.0678, 0.5162) ( 0.4418, 0.3864)
4 (-0.4403, 0.0000) ( 0.5294, 0.6361) (-0.3027,-0.0346) ( 0.1667, 0.0258)

Example 1: left singular vectors, by column
                  1               2               3               4
1 (-0.5634, 0.0016) (-0.2687,-0.2749) ( 0.2451, 0.4657) ( 0.3787, 0.2987)
2 ( 0.1205,-0.6108) (-0.2909, 0.1085) ( 0.4329,-0.1758) (-0.0182,-0.0437)
3 (-0.0816, 0.1613) (-0.1660, 0.3885) (-0.4667, 0.3821) (-0.0800,-0.2276)
4 ( 0.1441,-0.1532) ( 0.1984,-0.1737) (-0.0034, 0.1555) ( 0.2608,-0.5382)
5 (-0.2487,-0.0926) ( 0.6253, 0.3304) ( 0.2643,-0.0194) ( 0.1002, 0.0140)
6 (-0.3758, 0.0793) (-0.0307,-0.0816) ( 0.1266, 0.1747) (-0.4175,-0.4058)

Example 2: singular values
        3.0004          1.9967          0.9973

Example 2: right singular vectors, by row
                  1               2               3               4
1 ( 0.2454,-0.0001) ( 0.2942,-0.5843) ( 0.0162,-0.0810) ( 0.6794, 0.2083)
2 (-0.1692, 0.5194) ( 0.1915,-0.4374) ( 0.5205,-0.0244) (-0.3149,-0.3208)
3 (-0.5553, 0.1403) ( 0.1438,-0.1507) (-0.5684,-0.5505) (-0.0318,-0.0378)
```

```
Example 2: left singular vectors, by column
                  1                 2                 3
1 ( 0.6518, 0.0000) (-0.4312, 0.0000) ( 0.6239, 0.0000)
2 (-0.4437,-0.5027) (-0.3794, 0.1026) ( 0.2014, 0.5961)
3 (-0.2012, 0.2916) (-0.8122, 0.0030) (-0.3511,-0.3026)
```