NAG Library Routine Document

F08BPF (ZTPQRT)

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

F08BPF (ZTPQRT) computes the QR factorization of a complex $(m+n)$ by n triangular-pentagonal matrix.

2 Specification

```
SUBROUTINE F08BPF (M, N, L, NB, A, LDA, B, LDB, T, LDT, WORK, INFO)
INTEGER                 M, N, L, NB, LDA, LDB, LDT, INFO
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), T(LDT,*), WORK(*)
```

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

3 Description

F08BPF (ZTPQRT) forms the QR factorization of a complex $(m+n)$ by n triangular-pentagonal matrix C,

$$C = \begin{pmatrix} A \\ B \end{pmatrix}$$

where A is an upper triangular n by n matrix and B is an m by n pentagonal matrix consisting of an $(m-l)$ by n rectangular matrix B_1 on top of an l by n upper trapezoidal matrix B_2:

$$B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}.$$

The upper trapezoidal matrix B_2 consists of the first l rows of an n by n upper triangular matrix, where $0 \leq l \leq \min(m,n)$. If $l = 0$, B is m by n rectangular; if $l = n$ and $m = n$, B is upper triangular.

A recursive, explicitly blocked, QR factorization (see F08APF (ZGEQRT)) is performed on the matrix C. The upper triangular matrix R, details of the unitary matrix Q, and further details (the block reflector factors) of Q are returned.

Typically the matrix A or B_2 contains the matrix R from the QR factorization of a subproblem and F08BPF (ZTPQRT) performs the QR update operation from the inclusion of matrix B_1.

For example, consider the QR factorization of an l by n matrix $\hat{B}$ with $l < n$: $\hat{B} = \hat{Q}\hat{R}$, $\hat{R} = \begin{pmatrix} \hat{R}_1 & \hat{R}_2 \end{pmatrix}$, where $\hat{R}_1$ is l by l upper triangular and $\hat{R}_2$ is $(n-l)$ by n rectangular (this can be performed by F08APF (ZGEQRT)). Given an initial least-squares problem $\hat{B}\hat{X} = \hat{Y}$ where X and Y are l by $nrhs$ matrices, we have $\hat{R}\hat{X} = \hat{Q}^{\mathrm{H}}\hat{Y}$.

Now, adding an additional $m-l$ rows to the original system gives the augmented least squares problem

$$BX = Y$$

where B is an m by n matrix formed by adding $m-l$ rows on top of $\hat{R}$ and Y is an m by $nrhs$ matrix formed by adding $m-l$ rows on top of $\hat{Q}^{\mathrm{H}}\hat{Y}$.

F08BPF (ZTPQRT) can then be used to perform the QR factorization of the pentagonal matrix B; the n by n matrix A will be zero on input and contain R on output.

In the case where $\hat{B}$ is r by n, $r \geq n$, $\hat{R}$ is n by n upper triangular (forming A) on top of $r-n$ rows of zeros (forming first $r-n$ rows of B). Augmentation is then performed by adding rows to the bottom of B with $l = 0$.

4 References

Elmroth E and Gustavson F (2000) Applying Recursion to Serial and Parallel QR Factorization Leads to Better Performance *IBM Journal of Research and Development. (Volume 44)* **4** 605–624

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1: M – INTEGER *Input*

On entry: m, the number of rows of the matrix B.

Constraint: M ≥ 0.

2: N – INTEGER *Input*

On entry: n, the number of columns of the matrix B and the order of the upper triangular matrix A.

Constraint: N ≥ 0.

3: L – INTEGER *Input*

On entry: l, the number of rows of the trapezoidal part of B (i.e., B_2).

Constraint: $0 \leq$ L $\leq \min(\mathrm{M}, \mathrm{N})$.

4: NB – INTEGER *Input*

On entry: the explicitly chosen block-size to be used in the algorithm for computing the QR factorization. See Section 9 for details.

Constraints:

NB ≥ 1;
if N > 0, NB $\leq$ N.

5: A(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 n by n upper triangular matrix A.

On exit: the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

6: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08BPF (ZTPQRT) is called.

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

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

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

On entry: the m by n pentagonal matrix B composed of an $(m-l)$ by n rectangular matrix B_1 above an l by n upper trapezoidal matrix B_2.

On exit: details of the unitary matrix Q.

8: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08BPF (ZTPQRT) is called.

Constraint: LDB $\geq \max(1, \mathrm{M})$.

9: T(LDT, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array T must be at least N.

On exit: further details of the unitary matrix Q. The number of blocks is $b = \left\lceil \frac{k}{\mathrm{NB}} \right\rceil$, where $k = \min(m, n)$ and each block is of order NB except for the last block, which is of order $k - (b-1) \times \mathrm{NB}$. For each of the blocks, an upper triangular block reflector factor is computed: $\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_b$. These are stored in the NB by n matrix T as $\boldsymbol{T} = [\boldsymbol{T}_1 | \boldsymbol{T}_2 | \ldots | \boldsymbol{T}_b]$.

10: LDT – INTEGER *Input*

On entry: the first dimension of the array T as declared in the (sub)program from which F08BPF (ZTPQRT) is called.

Constraint: LDT $\geq$ NB.

11: WORK($*$) – COMPLEX (KIND=nag_wp) array *Workspace*

Note: the dimension of the array WORK must be at least NB $\times$ N.

12: 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 factorization is the exact factorization of a nearby matrix $(A + E)$, where

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

and ϵ is the ***machine precision***.

8 Parallelism and Performance

F08BPF (ZTPQRT) is not threaded by NAG in any implementation.

F08BPF (ZTPQRT) 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 approximately $\frac{2}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n-m)$ if $m < n$.

The block size, NB, used by F08BPF (ZTPQRT) is supplied explicitly through the interface. For moderate and large sizes of matrix, the block size can have a marked effect on the efficiency of the algorithm with the optimal value being dependent on problem size and platform. A value of $\mathrm{NB} = 64 \ll \min(m,n)$ is likely to achieve good efficiency and it is unlikely that an optimal value would exceed 340.

To apply Q to an arbitrary complex rectangular matrix C, F08BPF (ZTPQRT) may be followed by a call to F08BQF (ZTPMQRT). For example,

```
CALL ZTPMQRT('Left','Transpose',M,P,N,L,NB,B,LDB, &
             T,LDT,C,LDC,C(n+1,1),LDC,WORK,INFO)
```

forms $C = Q^{\mathrm{H}}C$, where C is $(m+n)$ by p.

To form the unitary matrix Q explicitly set $p = m+n$, initialize C to the identity matrix and make a call to F08BQF (ZTPMQRT) as above.

10 Example

This example finds the basic solutions for the linear least squares problems

$$\text{minimize } \|Ax_i - b_i\|_2, \quad i = 1,2$$

where b_1 and b_2 are the columns of the matrix B,

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

$$B = \begin{pmatrix} -2.09+1.93i & 3.26-2.70i \\ 3.34-3.53i & -6.22+1.16i \\ -4.94-2.04i & 7.94-3.13i \\ 0.17+4.23i & 1.04-4.26i \\ -5.19+3.63i & -2.31-2.12i \\ 0.98+2.53i & -1.39-4.05i \end{pmatrix}.$$

A QR factorization is performed on the first 4 rows of A using F08APF (ZGEQRT) after which the first 4 rows of B are updated by applying Q^T using F08AQF (ZGEMQRT). The remaining row is added by performing a QR update using F08BPF (ZTPQRT); B is updated by applying the new Q^T using F08BQF (ZTPMQRT); the solution is finally obtained by triangular solve using R from the updated QR.

10.1 Program Text

```
    Program f08bpfe

!     F08BPF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: dznrm2, nag_wp, x04dbf, zgemqrt, zgeqrt, ztpmqrt, &
                             ztpqrt, ztrtrs
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                :: nbmax = 64, nin = 5, nout = 6
```

```
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, j, lda, ldb, ldt,    &
                                          lwork, m, n, nb, nrhs
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), c(:,:), t(:,:),    &
                                            work(:)
      Real (Kind=nag_wp), Allocatable  :: rnorm(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, min
!     .. Executable Statements ..
      Write (nout,*) 'F08BPF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, nrhs
      lda = m
      ldb = m
      nb = min(m,n,nbmax)
      ldt = nb
      lwork = nb*max(n,m)
      Allocate (a(lda,n),b(ldb,nrhs),c(ldb,nrhs),rnorm(nrhs),t(ldt,min(m, &
        n)),work(lwork))

!     Read A and B from data file

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

      c(1:m,1:nrhs) = b(1:m,1:nrhs)
!     Compute the QR factorization of first n rows of A
!     The NAG name equivalent of zgeqrt is f08apf
      Call zgeqrt(n,n,nb,a,lda,t,ldt,work,info)

!     Compute C = (C1) = (Q**H)*B, storing the result in C
!                 (C2)
!     The NAG name equivalent of zgemqrt is f08aqf
      Call zgemqrt('Left','Conjugate Transpose',n,nrhs,n,nb,a,lda,t,ldt,c,ldb, &
        work,info)

      b(1:n,1:nrhs) = c(1:n,1:nrhs)
!     Compute least-squares solutions for first n rows by backsubstitution in
!     R*X = C1
!     The NAG name equivalent of ztrtrs is f07tsf
      Call ztrtrs('Upper','No transpose','Non-Unit',n,nrhs,a,lda,c,ldb,info)

      If (info>0) Then
        Write (nout,*) 'The upper triangular factor, R, of A is singular, '
        Write (nout,*) 'the least squares solution could not be computed'
      Else

!       Print solution using first n rows

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call x04dbf('General',' ',n,nrhs,c,ldb,'Bracketed','F7.4', &
          'solution(s) for n rows','Integer',rlabs,'Integer',clabs,80,0,ifail)

      End If

!     Now add the remaining rows and perform QR update
!     The NAG name equivalent of ztpqrt is f08bpf
      Call ztpqrt(m-n,n,0,nb,a,lda,a(n+1,1),lda,t,ldt,work,info)

!     Apply orthogonal transformations to C
!     The NAG name equivalent of ztpmqrt is f08bqf
      Call ztpmqrt('Left','Conjugate Transpose',m-n,nrhs,n,0,nb,a(n+1,1),lda, &
        t,ldt,b,ldb,b(5,1),ldb,work,info)
!     Compute least-squares solutions for first n rows by backsubstitution in
```

```
!     R*X = C1
!     The NAG name equivalent of ztrtrs is f07tsf
      Call ztrtrs('Upper','No transpose','Non-Unit',n,nrhs,a,lda,b,ldb,info)

      If (info>0) Then
        Write (nout,*) 'The upper triangular factor, R, of A is singular, '
        Write (nout,*) 'the least squares solution could not be computed'
      Else

!       Print least-squares solutions
        Write (nout,*)
        ifail = 0
        Call x04dbf('G',' ',n,nrhs,b,ldb,'Bracketed','F7.4', &
          'Least-squares solution(s) for all rows','Integer',rlabs,'Integer', &
          clabs,80,0,ifail)

!       Compute and print estimates of the square roots of the residual
!       sums of squares

!       The NAG name equivalent of dznrm2 is f06jjf
        Do j = 1, nrhs
          rnorm(j) = dznrm2(m-n,b(n+1,j),1)
        End Do

        Write (nout,*)
        Write (nout,*) 'Square root(s) of the residual sum(s) of squares'
        Write (nout,99999) rnorm(1:nrhs)
      End If


99999 Format (5X,1P,7E11.2)
    End Program f08bpfe
```

10.2 Program Data

```
F08BPF Example Program Data

  6      4      2                                          : m, n and nrhs

 ( 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) : matrix A

 (-2.09, 1.93) ( 3.26,-2.70)
 ( 3.34,-3.53) (-6.22, 1.16)
 (-4.94,-2.04) ( 7.94,-3.13)
 ( 0.17, 4.23) ( 1.04,-4.26)
 (-5.19, 3.63) (-2.31,-2.12)
 ( 0.98, 2.53) (-1.39,-4.05)                               : matrix B
```

10.3 Program Results

```
 F08BPF Example Program Results

 solution(s) for n rows
                     1                  2
 1  (-0.5091,-1.2428) ( 0.7569, 1.4384)
 2  (-2.3789, 2.8651) ( 5.1727,-3.6193)
 3  ( 1.4634,-2.2064) (-2.6613, 2.1339)
 4  ( 0.4701, 2.6964) (-2.6933, 0.2724)

 Least-squares solution(s) for all rows
                     1                  2
 1  (-0.5044,-1.2179) ( 0.7629, 1.4529)
 2  (-2.4281, 2.8574) ( 5.1570,-3.6089)
```

```
 3  ( 1.4872,-2.1955) (-2.6518, 2.1203)
 4  ( 0.4537, 2.6904) (-2.7606, 0.3318)

 Square root(s) of the residual sum(s) of squares
        6.88E-02   1.87E-01
```