

# NAG Library Function Document

## nag\_ztpmqrt (f08bqc)

### 1 Purpose

nag\_ztpmqrt (f08bqc) multiplies an arbitrary complex matrix  $C$  by the complex unitary matrix  $Q$  from a  $QR$  factorization computed by nag\_ztpqrt (f08bpc).

### 2 Specification

```
#include <nag.h>
#include <nagf08.h>

void nag_ztpmqrt (Nag_OrderType order, Nag_SideType side,
                  Nag_TransType trans, Integer m, Integer n, Integer k, Integer l,
                  Integer nb, const Complex v[], Integer pdv, const Complex t[],
                  Integer pdt, Complex c1[], Integer pdc1, Complex c2[], Integer pdc2,
                  NagError *fail)
```

### 3 Description

nag\_ztpmqrt (f08bqc) is intended to be used after a call to nag\_ztpqrt (f08bpc) which performs a  $QR$  factorization of a triangular-pentagonal matrix containing an upper triangular matrix  $A$  over a pentagonal matrix  $B$ . The unitary matrix  $Q$  is represented as a product of elementary reflectors.

This function may be used to form the matrix products

$$QC, Q^H C, CQ \text{ or } CQ^H,$$

where the complex rectangular  $m_c$  by  $n_c$  matrix  $C$  is split into component matrices  $C_1$  and  $C_2$ .

If  $Q$  is being applied from the left ( $QC$  or  $Q^H C$ ) then

$$C = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}$$

where  $C_1$  is  $k$  by  $n_c$ ,  $C_2$  is  $m_v$  by  $n_c$ ,  $m_c = k + m_v$  is fixed and  $m_v$  is the number of rows of the matrix  $V$  containing the elementary reflectors (i.e.,  $\mathbf{m}$  as passed to nag\_ztpqrt (f08bpc)); the number of columns of  $V$  is  $n_v$  (i.e.,  $\mathbf{n}$  as passed to nag\_ztpqrt (f08bpc)).

If  $Q$  is being applied from the right ( $CQ$  or  $CQ^H$ ) then

$$C = \begin{pmatrix} C_1 & C_2 \end{pmatrix}$$

where  $C_1$  is  $m_c$  by  $k$ , and  $C_2$  is  $m_c$  by  $m_v$  and  $n_c = k + m_v$  is fixed.

The matrices  $C_1$  and  $C_2$  are overwritten by the result of the matrix product.

A common application of this routine is in updating the solution of a linear least squares problem as illustrated in Section 10 in nag\_ztpqrt (f08bpc).

### 4 References

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

## 5 Arguments

- 1: **order** – Nag\_OrderType *Input*  
*On entry:* the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag\_RowMajor. See Section 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.  
*Constraint:* **order** = Nag\_RowMajor or Nag\_ColMajor.
- 2: **side** – Nag\_SideType *Input*  
*On entry:* indicates how  $Q$  or  $Q^H$  is to be applied to  $C$ .  
**side** = Nag\_LeftSide  
 $Q$  or  $Q^H$  is applied to  $C$  from the left.  
**side** = Nag\_RightSide  
 $Q$  or  $Q^H$  is applied to  $C$  from the right.  
*Constraint:* **side** = Nag\_LeftSide or Nag\_RightSide.
- 3: **trans** – Nag\_TransType *Input*  
*On entry:* indicates whether  $Q$  or  $Q^H$  is to be applied to  $C$ .  
**trans** = Nag\_NoTrans  
 $Q$  is applied to  $C$ .  
**trans** = Nag\_ConjTrans  
 $Q^H$  is applied to  $C$ .  
*Constraint:* **trans** = Nag\_NoTrans or Nag\_ConjTrans.
- 4: **m** – Integer *Input*  
*On entry:* the number of rows of the matrix  $C_2$ , that is,  
if **side** = Nag\_LeftSide  
then  $m_v$ , the number of rows of the matrix  $V$ ;  
if **side** = Nag\_RightSide  
then  $m_c$ , the number of rows of the matrix  $C$ .  
*Constraint:* **m**  $\geq 0$ .
- 5: **n** – Integer *Input*  
*On entry:* the number of columns of the matrix  $C_2$ , that is,  
if **side** = Nag\_LeftSide  
then  $n_c$ , the number of columns of the matrix  $C$ ;  
if **side** = Nag\_RightSide  
then  $n_v$ , the number of columns of the matrix  $V$ .  
*Constraint:* **n**  $\geq 0$ .
- 6: **k** – Integer *Input*  
*On entry:*  $k$ , the number of elementary reflectors whose product defines the matrix  $Q$ .  
*Constraint:* **k**  $\geq 0$ .

- 7: **l** – Integer *Input*  
*On entry:*  $l$ , the number of rows of the upper trapezoidal part of the pentagonal composite matrix  $V$ , passed (as **b**) in a previous call to nag\_ztpqrt (f08bpc). This must be the same value used in the previous call to nag\_ztpqrt (f08bpc) (see **l** in nag\_ztpqrt (f08bpc)).  
*Constraint:*  $0 \leq l \leq k$ .
- 8: **nb** – Integer *Input*  
*On entry:*  $nb$ , the blocking factor used in a previous call to nag\_ztpqrt (f08bpc) to compute the  $QR$  factorization of a triangular-pentagonal matrix containing composite matrices  $A$  and  $B$ .  
*Constraints:*  
 $nb \geq 1$ ;  
if  $k > 0$ ,  $nb \leq k$ .
- 9: **v**[ $dim$ ] – const Complex *Input*  
**Note:** the dimension,  $dim$ , of the array **v** must be at least  
 $\max(1, p_{dv} \times k)$  when **order** = Nag\_ColMajor;  
 $\max(1, m \times p_{dv})$  when **order** = Nag\_RowMajor and **side** = Nag\_LeftSide;  
 $\max(1, n \times p_{dv})$  when **order** = Nag\_RowMajor and **side** = Nag\_RightSide.  
The  $(i, j)$ th element of the matrix  $V$  is stored in  
 $v[(j-1) \times p_{dv} + i - 1]$  when **order** = Nag\_ColMajor;  
 $v[(i-1) \times p_{dv} + j - 1]$  when **order** = Nag\_RowMajor.  
*On entry:* the  $m_v$  by  $n_v$  matrix  $V$ ; this should remain unchanged from the array **b** returned by a previous call to nag\_ztpqrt (f08bpc).
- 10: **p<sub>dv</sub>** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **v**.  
*Constraints:*  
if **order** = Nag\_ColMajor,  
if **side** = Nag\_LeftSide,  $p_{dv} \geq \max(1, m)$ ;  
if **side** = Nag\_RightSide,  $p_{dv} \geq \max(1, n)$ .;  
if **order** = Nag\_RowMajor,  $p_{dv} \geq \max(1, k)$ .
- 11: **t**[ $dim$ ] – const Complex *Input*  
**Note:** the dimension,  $dim$ , of the array **t** must be at least  
 $\max(1, p_{dt} \times k)$  when **order** = Nag\_ColMajor;  
 $\max(1, nb \times p_{dt})$  when **order** = Nag\_RowMajor.  
The  $(i, j)$ th element of the matrix  $T$  is stored in  
 $t[(j-1) \times p_{dt} + i - 1]$  when **order** = Nag\_ColMajor;  
 $t[(i-1) \times p_{dt} + j - 1]$  when **order** = Nag\_RowMajor.  
*On entry:* this must remain unchanged from a previous call to nag\_ztpqrt (f08bpc) (see **t** in nag\_ztpqrt (f08bpc)).
- 12: **p<sub>dt</sub>** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **t**.

*Constraints:*

if **order** = Nag\_ColMajor, **pdt**  $\geq$  **nb**;  
 if **order** = Nag\_RowMajor, **pdt**  $\geq$  max(1, **k**).

13: **c1**[*dim*] – Complex

*Input/Output*

**Note:** the dimension, *dim*, of the array **c1** must be at least

max(1, **pdc1**  $\times$  **n**) when **side** = Nag\_LeftSide and **order** = Nag\_ColMajor;  
 max(1, **k**  $\times$  **pdc1**) when **side** = Nag\_LeftSide and **order** = Nag\_RowMajor;  
 max(1, **pdc1**  $\times$  **k**) when **side** = Nag\_RightSide and **order** = Nag\_ColMajor;  
 max(1, **m**  $\times$  **pdc1**) when **side** = Nag\_RightSide and **order** = Nag\_RowMajor.

*On entry:*  $C_1$ , the first part of the composite matrix  $C$ :

if **side** = Nag\_LeftSide  
 then **c1** contains the first  $k$  rows of  $C$ ;

if **side** = Nag\_RightSide  
 then **c1** contains the first  $k$  columns of  $C$ .

*On exit:* **c1** is overwritten by the corresponding block of  $QC$  or  $Q^H C$  or  $CQ$  or  $CQ^H$ .

14: **pdc1** – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **c1**.

*Constraints:*

if **order** = Nag\_ColMajor,  
     if **side** = Nag\_LeftSide, **pdc1**  $\geq$  max(1, **k**);  
     if **side** = Nag\_RightSide, **pdc1**  $\geq$  max(1, **m**);  
 if **order** = Nag\_RowMajor,  
     if **side** = Nag\_LeftSide, **pdc1**  $\geq$  max(1, **n**);  
     if **side** = Nag\_RightSide, **pdc1**  $\geq$  max(1, **k**).

15: **c2**[*dim*] – Complex

*Input/Output*

**Note:** the dimension, *dim*, of the array **c2** must be at least

max(1, **pdc2**  $\times$  **n**) when **order** = Nag\_ColMajor;  
 max(1, **m**  $\times$  **pdc2**) when **order** = Nag\_RowMajor.

*On entry:*  $C_2$ , the second part of the composite matrix  $C$ .

if **side** = Nag\_LeftSide  
 then **c2** contains the remaining  $m_v$  rows of  $C$ ;

if **side** = Nag\_RightSide  
 then **c2** contains the remaining  $m_v$  columns of  $C$ ;

*On exit:* **c2** is overwritten by the corresponding block of  $QC$  or  $Q^H C$  or  $CQ$  or  $CQ^H$ .

16: **pdc2** – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **c2**.

*Constraints:*

if **order** = Nag\_ColMajor, **pdc2**  $\geq$  max(1, **m**);  
 if **order** = Nag\_RowMajor, **pdc2**  $\geq$  max(1, **n**).

17: **fail** – NagError \*

Input/Output

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_ENUM\_INT\_3

On entry, **side** =  $\langle value \rangle$ , **k** =  $\langle value \rangle$ , **m** =  $\langle value \rangle$  and **pdc1** =  $\langle value \rangle$ .

Constraint: if **side** = Nag\_LeftSide, **pdc1**  $\geq \max(1, \mathbf{k})$ ;

if **side** = Nag\_RightSide, **pdc1**  $\geq \max(1, \mathbf{m})$ .

On entry, **side** =  $\langle value \rangle$ , **m** =  $\langle value \rangle$ , **n** =  $\langle value \rangle$  and **pdv** =  $\langle value \rangle$ .

Constraint: if **side** = Nag\_LeftSide, **pdv**  $\geq \max(1, \mathbf{m})$ ;

if **side** = Nag\_RightSide, **pdv**  $\geq \max(1, \mathbf{n})$ .

On entry, **side** =  $\langle value \rangle$ , **pdc1** =  $\langle value \rangle$ , **n** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .

Constraint: if **side** = Nag\_LeftSide, **pdc1**  $\geq \max(1, \mathbf{n})$ ;

if **side** = Nag\_RightSide, **pdc1**  $\geq \max(1, \mathbf{k})$ .

### NE\_INT

On entry, **k** =  $\langle value \rangle$ .

Constraint: **k**  $\geq 0$ .

On entry, **m** =  $\langle value \rangle$ .

Constraint: **m**  $\geq 0$ .

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 0$ .

### NE\_INT\_2

On entry, **l** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .

Constraint:  $0 \leq \mathbf{l} \leq \mathbf{k}$ .

On entry, **m** =  $\langle value \rangle$  and **pdc2** =  $\langle value \rangle$ .

Constraint: **pdc2**  $\geq \max(1, \mathbf{m})$ .

On entry, **nb** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .

Constraint: **nb**  $\geq 1$  and

if **k**  $> 0$ , **nb**  $\leq \mathbf{k}$ .

On entry, **pdc2** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: **pdc2**  $\geq \max(1, \mathbf{n})$ .

On entry, **pdt** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .

Constraint: **pdt**  $\geq \max(1, \mathbf{k})$ .

On entry, **pdt** =  $\langle value \rangle$  and **nb** =  $\langle value \rangle$ .

Constraint: **pdt**  $\geq \mathbf{nb}$ .

On entry, **pdv** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .

Constraint: **pdv**  $\geq \max(1, \mathbf{k})$ .

**NE\_INTERNAL\_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**7 Accuracy**

The computed result differs from the exact result by a matrix  $E$  such that

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

where  $\epsilon$  is the *machine precision*.

**8 Parallelism and Performance**

nag\_ztpmqrt (f08bqc) 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 function. 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  $2nk(2m - k)$  if **side** = Nag\_LeftSide and  $2mk(2n - k)$  if **side** = Nag\_RightSide.

The real analogue of this function is nag\_dtpmqrt (f08bcc).

**10 Example**

See Section 10 in nag\_ztpqrt (f08bpc).

---