NAG Library Function Document

nag_dtpmqrt (f08bcc)

1 Purpose

nag_dtpmqrt (f08bcc) multiplies an arbitrary real matrix $C$ by the real orthogonal matrix $Q$ from a $QR$ factorization computed by nag_dtpqrt (f08bbc).

2 Specification

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

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

3 Description

nag_dtpmqrt (f08bcc) is intended to be used after a call to nag_dtpqrt (f08bbc) which performs a $QR$ factorization of a triangular-pentagonal matrix containing an upper triangular matrix $A$ over a pentagonal matrix $B$. The orthogonal matrix $Q$ is represented as a product of elementary reflectors.

This function may be used to form the matrix products

$$QC, Q^TC, CQ 	ext{ or } CQ^T,$$

where the real 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 $Q^TC$) then

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

where $C_1$ is $k$ by $n_c$, $C_2$ is $m_c$ 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., $m$ as passed to nag_dtpqrt (f08bbc)); the number of columns of $V$ is $n_v$ (i.e., $n$ as passed to nag_dtpqrt (f08bbc)).

If $Q$ is being applied from the right ($CQ$ or $CQ^T$) 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_dtpqrt (f08bbc).

4 References

5 Arguments

1: \texttt{order} – Nag_OrderType \hspace{1cm} \textit{Input}

\textit{On entry:} the \texttt{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 \texttt{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \texttt{order} = Nag_RowMajor or Nag_ColMajor.

2: \texttt{side} – Nag_SideType \hspace{1cm} \textit{Input}

\textit{On entry:} indicates how $Q$ or $Q^T$ is to be applied to $C$.

\texttt{side} = Nag_LeftSide

$Q$ or $Q^T$ is applied to $C$ from the left.

\texttt{side} = Nag_RightSide

$Q$ or $Q^T$ is applied to $C$ from the right.

\textit{Constraint:} \texttt{side} = Nag_LeftSide or Nag_RightSide.

3: \texttt{trans} – Nag_TransType \hspace{1cm} \textit{Input}

\textit{On entry:} indicates whether $Q$ or $Q^T$ is to be applied to $C$.

\texttt{trans} = Nag_NoTrans

$Q$ is applied to $C$.

\texttt{trans} = Nag_Trans

$Q^T$ is applied to $C$.

\textit{Constraint:} \texttt{trans} = Nag_NoTrans or Nag_Trans.

4: \texttt{m} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of rows of the matrix $C_2$, that is,

\texttt{if} \texttt{side} = Nag_LeftSide

then $m_v$, the number of rows of the matrix $V$;

\texttt{if} \texttt{side} = Nag_RightSide

then $m_v$, the number of rows of the matrix $C$.

\textit{Constraint:} \texttt{m} $\geq$ 0.

5: \texttt{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of columns of the matrix $C_2$, that is,

\texttt{if} \texttt{side} = Nag_LeftSide

then $n_v$, the number of columns of the matrix $C$;

\texttt{if} \texttt{side} = Nag_RightSide

then $n_v$, the number of columns of the matrix $V$.

\textit{Constraint:} \texttt{n} $\geq$ 0.

6: \texttt{k} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} $k$, the number of elementary reflectors whose product defines the matrix $Q$.

\textit{Constraint:} \texttt{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_dtpqrt (f08bbc). This must be the same value used in the previous call to nag_dtpqrt (f08bbc) (see l in nag_dtpqrt (f08bbc)).

Constraint: 0 ≤ l ≤ k.

8: nb – Integer  
Input

On entry: nb, the blocking factor used in a previous call to nag_dtpqrt (f08bbc) to compute the QR factorization of a triangular-pentagonal matrix containing composite matrices A and B.

Constraints:

nb ≥ 1;
if k > 0, nb ≤ k.

9: v[dim] – const double  
Input

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

max(1, pdv × k) when order = Nag_ColMajor;
max(1, m × pdv) when order = Nag_RowMajor and side = Nag_LeftSide;
max(1, n × pdv) when order = Nag_RowMajor and side = Nag_RightSide.

The (i, j)th element of the matrix V is stored in

v[(j - 1) × pdv + i - 1] when order = Nag_ColMajor;
v[(i - 1) × pdv + 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_dtpqrt (f08bbc).

10: pdv – 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, pdv ≥ max(1, m);
    if side = Nag_RightSide, pdv ≥ max(1, n);

if order = Nag_RowMajor, pdv ≥ max(1, k).

11: t[dim] – const double  
Input

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

max(1, pdt × k) when order = Nag_ColMajor;
max(1, nb × pdt) when order = Nag_RowMajor.

The (i, j)th element of the matrix T is stored in

t[(j - 1) × pdt + i - 1] when order = Nag_ColMajor;
t[(i - 1) × pdt + j - 1] when order = Nag_RowMajor.

On entry: this must remain unchanged from a previous call to nag_dtpqrt (f08bbc) (see t in nag_dtpqrt (f08bbc)).

12: pdt – Integer  
Input

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

if \( \text{order} = \text{Nag\_ColMajor}, \text{pdt} \geq nb \);
if \( \text{order} = \text{Nag\_RowMajor}, \text{pdt} \geq \max(1, k) \).

13: \( \text{c1}[\text{dim}] \) – double

\text{Input/Output}

\text{Note:} \ the \ dimension, \ \text{dim}, \ of \ the \ array \ \text{c1} \ must \ be \ at \ least
\( \max(1, \text{pdc1} \times n) \) \ when \ \text{side} = \text{Nag\_LeftSide} \ and \ \text{order} = \text{Nag\_ColMajor};
\( \max(1, k \times \text{pdc1}) \) \ when \ \text{side} = \text{Nag\_LeftSide} \ and \ \text{order} = \text{Nag\_RowMajor};
\( \max(1, \text{pdc1} \times k) \) \ when \ \text{side} = \text{Nag\_RightSide} \ and \ \text{order} = \text{Nag\_ColMajor};
\( \max(1, m \times \text{pdc1}) \) \ when \ \text{side} = \text{Nag\_RightSide} \ and \ \text{order} = \text{Nag\_RowMajor}.

\text{On entry:} \ \text{c1}, \ the \ first \ part \ of \ the \ composite \ matrix \ \text{C}.
if \ \text{side} = \text{Nag\_LeftSide}
then \ \text{c1} \ contains \ the \ first \ \text{k} \ rows \ of \ \text{C};
if \ \text{side} = \text{Nag\_RightSide}
then \ \text{c1} \ contains \ the \ first \ \text{k} \ columns \ of \ \text{C}.

\text{On exit:} \ \text{c1} \ is \ overwritten \ by \ the \ corresponding \ block \ of \ \text{QC} \ or \ \text{Q}^T\text{C} \ or \ \text{CQ} \ or \ \text{CQ}^T.

14: \( \text{pdc1} \) – Integer

\text{Input}

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

\text{Constraints:}

if \ \text{order} = \text{Nag\_ColMajor},
\quad \text{if} \ \text{side} = \text{Nag\_LeftSide}, \ \text{pdc1} \geq \max(1, k);
\quad \text{if} \ \text{side} = \text{Nag\_RightSide}, \ \text{pdc1} \geq \max(1, m);
if \ \text{order} = \text{Nag\_RowMajor},
\quad \text{if} \ \text{side} = \text{Nag\_LeftSide}, \ \text{pdc1} \geq \max(1, n);
\quad \text{if} \ \text{side} = \text{Nag\_RightSide}, \ \text{pdc1} \geq \max(1, k).

15: \( \text{c2}[\text{dim}] \) – double

\text{Input/Output}

\text{Note:} \ the \ dimension, \ \text{dim}, \ of \ the \ array \ \text{c2} \ must \ be \ at \ least
\( \max(1, \text{pdc2} \times n) \) \ when \ \text{order} = \text{Nag\_ColMajor};
\( \max(1, m \times \text{pdc2}) \) \ when \ \text{order} = \text{Nag\_RowMajor}.

\text{On entry:} \ \text{C2}, \ the \ second \ part \ of \ the \ composite \ matrix \ \text{C}.
if \ \text{side} = \text{Nag\_LeftSide}
then \ \text{c2} \ contains \ the \ remaining \ \text{m}_e \ rows \ of \ \text{C};
if \ \text{side} = \text{Nag\_RightSide}
then \ \text{c2} \ contains \ the \ remaining \ \text{m}_e \ columns \ of \ \text{C};

\text{On exit:} \ \text{c2} \ is \ overwritten \ by \ the \ corresponding \ block \ of \ \text{QC} \ or \ \text{Q}^T\text{C} \ or \ \text{CQ} \ or \ \text{CQ}^T.

16: \( \text{pdc2} \) – Integer

\text{Input}

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

\text{Constraints:}

if \ \text{order} = \text{Nag\_ColMajor}, \ \text{pdc2} \geq \max(1, m);
if \ \text{order} = \text{Nag\_RowMajor}, \ \text{pdc2} \geq \max(1, n).
6 Error Indicators and Warnings

**NE_ALLOC_FAIL**
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**
On entry, argument ⟨value⟩ had an illegal value.

**NE_ENUM_INT_3**
On entry, side = ⟨value⟩, k = ⟨value⟩, m = ⟨value⟩ and pdc1 = ⟨value⟩.
Constraint: if side = Nag_LeftSide, pdc1 ≥ max(1, k);
if side = Nag_RightSide, pdc1 ≥ max(1, m).

On entry, side = ⟨value⟩, m = ⟨value⟩, n = ⟨value⟩ and pdv = ⟨value⟩.
Constraint: if side = Nag_LeftSide, pdv ≥ max(1, m);
if side = Nag_RightSide, pdv ≥ max(1, n).

On entry, side = ⟨value⟩, pdc1 = ⟨value⟩, n = ⟨value⟩ and k = ⟨value⟩.
Constraint: if side = Nag_LeftSide, pdc1 ≥ max(1, n);
if side = Nag_RightSide, pdc1 ≥ max(1, k).

**NE_INT**
On entry, k = ⟨value⟩.
Constraint: k ≥ 0.

On entry, m = ⟨value⟩.
Constraint: m ≥ 0.

On entry, n = ⟨value⟩.
Constraint: n ≥ 0.

**NE_INT_2**
On entry, l = ⟨value⟩ and k = ⟨value⟩.
Constraint: 0 ≤ l ≤ k.

On entry, m = ⟨value⟩ and pdc2 = ⟨value⟩.
Constraint: pdc2 ≥ max(1, m).

On entry, nb = ⟨value⟩ and k = ⟨value⟩.
Constraint: nb ≥ 1 and if k > 0, nb ≤ k.

On entry, pdc2 = ⟨value⟩ and n = ⟨value⟩.
Constraint: pdc2 ≥ max(1, n).

On entry, pdt = ⟨value⟩ and k = ⟨value⟩.
Constraint: pdt ≥ max(1, k).

On entry, pdt = ⟨value⟩ and nb = ⟨value⟩.
Constraint: pdt ≥ nb.

On entry, pdv = ⟨value⟩ and k = ⟨value⟩.
Constraint: pdv ≥ max(1, 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 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction 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_dtpmqrt (f08bcc) is not threaded by NAG in any implementation.

nag_dtpmqrt (f08bcc) 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 complex analogue of this function is nag_ztpmqrt (f08bqc).

10 Example
See Section 10 in nag_dtpqrt (f08bbc).