NAG C Library Function Document

nag_zunmqr (f08auc)

1 Purpose

nag_zunmqr (f08auc) multiplies an arbitrary complex matrix $C$ by the complex unitary matrix $Q$ from a $QR$ factorization computed by nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc).

2 Specification

```c
void nag_zunmqr (Nag_OrderType order, Nag_SideType side, Nag_TransType trans, 
    Integer m, Integer n, Integer k, const Complex a[], Integer pda, 
    const Complex tau[], Complex c[], Integer pdc, NagError *fail)
```

3 Description

nag_zunmqr (f08auc) is intended to be used after a call to nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc), which perform a $QR$ factorization of a complex matrix $A$. The unitary matrix $Q$ is represented as a product of elementary reflectors.

This function may be used to form one of the matrix products

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

overwriting the result on $C$ (which may be any complex rectangular matrix).

A common application of this function is in solving linear least-squares problems, as described in the f08 Chapter Introduction, and illustrated in Section 9 of the document for nag_zgeqrf (f08asc).

4 References


5 Parameters

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

\textit{On entry:} the \texttt{order} parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

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

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

\textit{On entry:} indicates how $Q$ or $Q^H$ is to be applied to $C$ as follows:

- if \texttt{side = Nag_LeftSide}, $Q$ or $Q^H$ is applied to $C$ from the left;
- if \texttt{side = Nag_RightSide}, $Q$ or $Q^H$ is applied to $C$ from the right.

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

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

\textit{On entry:} indicates whether $Q$ or $Q^H$ is to be applied to $C$ as follows:
if \( \text{trans} = \text{Nag\_NoTrans} \), \( Q \) is applied to \( C \);

if \( \text{trans} = \text{Nag\_ConjTrans} \), \( Q^H \) is applied to \( C \).

\[
\text{Constraint: } \text{trans} = \text{Nag\_NoTrans} \text{ or } \text{Nag\_ConjTrans}.
\]

4: \( m \) – Integer \( \text{Input} \)

On entry: \( m \), the number of rows of the matrix \( C \).

\[
\text{Constraint: } m \geq 0.
\]

5: \( n \) – Integer \( \text{Input} \)

On entry: \( n \), the number of columns of the matrix \( C \).

\[
\text{Constraint: } n \geq 0.
\]

6: \( k \) – Integer \( \text{Input} \)

On entry: \( k \), the number of elementary reflectors whose product defines the matrix \( Q \).

\[
\text{Constraints:}
\]

if \( \text{side} = \text{Nag\_LeftSide} \), \( m \geq k \geq 0 \);

if \( \text{side} = \text{Nag\_RightSide} \), \( n \geq k \geq 0 \).

7: \( a[\text{dim}] \) – Complex \( \text{Input/Output} \)

Note: the dimension, \( \text{dim} \), of the array \( a \) must be at least

\[
\max(1, \text{pda} \times k) \text{ when } \text{order} = \text{Nag\_ColMajor};
\]

\[
\max(1, \text{pda} \times m) \text{ when } \text{order} = \text{Nag\_RowMajor} \text{ and } \text{side} = \text{Nag\_LeftSide};
\]

\[
\max(1, \text{pda} \times n) \text{ when } \text{order} = \text{Nag\_RowMajor} \text{ and } \text{side} = \text{Nag\_RightSide}.
\]

If \( \text{order} = \text{Nag\_ColMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times \text{pda} + i - 1] \) and

if \( \text{order} = \text{Nag\_RowMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(i-1) \times \text{pda} + j - 1] \).

On entry: details of the vectors which define the elementary reflectors, as returned by \( \text{nag\_zgeqrf} \) (f08asc) or \( \text{nag\_zgeqpf} \) (f08bsc).

On exit: used as internal workspace prior to being restored and hence is unchanged.

8: \( \text{pda} \) – Integer \( \text{Input} \)

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

\[
\text{Constraints:}
\]

if \( \text{order} = \text{Nag\_ColMajor} \),

\[
\text{if } \text{side} = \text{Nag\_LeftSide}, \text{pda} \geq \max(1, m);
\]

\[
\text{if } \text{side} = \text{Nag\_RightSide}, \text{pda} \geq \max(1, n);
\]

if \( \text{order} = \text{Nag\_RowMajor} \), \( \text{pda} \geq \max(1, k) \).

9: \( \tau[\text{dim}] \) – const Complex \( \text{Input} \)

Note: the dimension, \( \text{dim} \), of the array \( \tau \) must be at least \( \max(1, k) \).

On entry: further details of the elementary reflectors, as returned by \( \text{nag\_zgeqrf} \) (f08asc) or \( \text{nag\_zgeqpf} \) (f08bsc).

10: \( c[\text{dim}] \) – Complex \( \text{Input/Output} \)

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

If \( \text{order} = \text{Nag\_ColMajor} \), the \((i, j)\)th element of the matrix \( C \) is stored in \( c[(j-1) \times \text{pdc} + i - 1] \) and

if \( \text{order} = \text{Nag\_RowMajor} \), the \((i, j)\)th element of the matrix \( C \) is stored in \( c[(i-1) \times \text{pdc} + j - 1] \).
On entry: the $m$ by $n$ matrix $C$.

On exit: $c$ is overwritten by $QC$ or $Q^H C$ or $CQ$ or $CQ^H$ as specified by side and trans.

11: pdc – Integer

Input

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

Constraints:

if order = Nag_ColMajor, pdc $\geq \max(1, m)$;
if order = Nag_RowMajor, pdc $\geq \max(1, n)$.

12: fail – NagError *

Output

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

Constraint: $m \geq 0$.

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

Constraint: $n \geq 0$.

On entry, $pda = \langle value \rangle$.

Constraint: $pda > 0$.

On entry, $pdc = \langle value \rangle$.

Constraint: $pdc > 0$.

NE_INT_2

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

Constraint: $pda \geq \max(1, k)$.

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

Constraint: $pdc \geq \max(1, m)$.

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

Constraint: $pdc \geq \max(1, n)$.

NE_ENUM_INT_3

On entry, side = $\langle value \rangle$, $m = \langle value \rangle$, $n = \langle value \rangle$, $k = \langle value \rangle$.

Constraint: if side = Nag_LeftSide, $m \geq k \geq 0$;
if side = Nag_RightSide, $n \geq k \geq 0$.

On entry, side = $\langle value \rangle$, $m = \langle value \rangle$, $n = \langle value \rangle$, $pda = \langle value \rangle$.

Constraint: if side = Nag_LeftSide, $pda \geq \max(1, m)$;
if side = Nag_RightSide, $pda \geq \max(1, n)$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal value.
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 consult NAG for assistance.

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 Further Comments
The total number of real floating-point operations is approximately $8nk(2m - k)$ if $\text{side} = \text{Nag_LeftSide}$ and $8mk(2n - k)$ if $\text{side} = \text{Nag_RightSide}$.

The real analogue of this function is nag_dormqr (f08agc).

9 Example
See Section 9 of the document for nag_zgeqrf (f08asc).