NAG Library Function Document

nag_zunmql (f08cuc)

1 Purpose

nag_zunmql (f08cuc) multiplies a general complex \( m \) by \( n \) matrix \( C \) by the complex unitary matrix \( Q \) from a \( QL \) factorization computed by nag_zgeqlf (f08csc).

2 Specification

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

void nag_zunmql (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_zunmql (f08cuc) is intended to be used following a call to nag_zgeqlf (f08csc), which performs a \( QL \) factorization of a complex matrix \( A \) and represents the unitary matrix \( Q \) as a product of elementary reflectors.

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

\[
QC, \quad Q^H C, \quad CQ, \quad CQ^H ,
\]

overwriting the result on \( C \), which may be any complex rectangular \( m \) by \( n \) 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 10 in nag_zgeqlf (f08csc).

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\textit{Input}

\textit{On entry:} the \textit{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 \textit{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:} \textit{order} \( = \) Nag_RowMajor or Nag_ColMajor.

2: \( \text{side} \) – Nag_SideType

\textit{Input}

\textit{On entry:} indicates how \( Q \) or \( Q^H \) is to be applied to \( C \).

\textit{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

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

On entry: $m$, the number of rows of the matrix $C$.

Constraint: $m \geq 0$.

5: $n$ – Integer

On entry: $n$, the number of columns of the matrix $C$.

Constraint: $n \geq 0$.

6: $k$ – Integer

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

Constraints:

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

7: $a[dim]$ – const Complex

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

max$(1, pda \times k)$ when order = Nag_ColMajor;
max$(1, m \times pda)$ when order = Nag_RowMajor and side = Nag_LeftSide;
max$(1, n \times pda)$ when order = Nag_RowMajor and side = Nag_RightSide.

On entry: details of the vectors which define the elementary reflectors, as returned by nag_zgeqlf (f08ecf).

On exit: is modified by nag_zunmql (f08ecf) but restored on exit.

8: $pda$ – Integer

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

Constraints:

if order = Nag_ColMajor,
    if side = Nag_LeftSide, pda $\geq$ max$(1, m)$;
    if side = Nag_RightSide, pda $\geq$ max$(1, n)$;
if order = Nag_RowMajor, pda $\geq$ max$(1, k)$.

9: $\tau[dim]$ – const Complex

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

On entry: further details of the elementary reflectors, as returned by nag_zgeqlf (f08ecf).
10:  \( \mathbf{c}[\text{dim}] \) – Complex \hspace{1cm} \text{Input/Output}

Note: the dimension, \( \text{dim} \), of the array \( \mathbf{c} \) must be at least
\[ \max(1, \text{pdc} \times \text{n}) \] when \( \text{order} = \text{Nag}_\text{ColMajor} \);
\[ \max(1, \text{m} \times \text{pdc}) \] when \( \text{order} = \text{Nag}_\text{RowMajor} \).

The \((i,j)\)th element of the matrix \( \mathbf{C} \) is stored in
\[ \mathbf{c}[(j-1) \times \text{pdc} + i - 1] \] when \( \text{order} = \text{Nag}_\text{ColMajor} \);
\[ \mathbf{c}[(i-1) \times \text{pdc} + j - 1] \] when \( \text{order} = \text{Nag}_\text{RowMajor} \).

On entry: the \( \text{m} \) by \( \text{n} \) matrix \( \mathbf{C} \).
On exit: \( \mathbf{c} \) is overwritten by \( \mathbf{QC} \) or \( \mathbf{Q}^\mathbf{H} \mathbf{C} \) or \( \mathbf{CQ} \) or \( \mathbf{CQ}^\mathbf{H} \) as specified by \( \text{side} \) and \( \text{trans} \).

11:  \( \text{pdc} \) – Integer \hspace{1cm} \text{Input}

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

Constraints:
\[ \text{if } \text{order} = \text{Nag}_\text{ColMajor}, \text{pdc} \geq \max(1, \text{m}); \]
\[ \text{if } \text{order} = \text{Nag}_\text{RowMajor}, \text{pdc} \geq \max(1, \text{n}). \]

12:  \( \text{fail} \) – NagError* \hspace{1cm} \text{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

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 \( \langle \text{value} \rangle \) had an illegal value.

**NE_ENUM_INT_3**

On entry, \( \text{side} = \langle \text{value} \rangle \), \( \text{m} = \langle \text{value} \rangle \), \( \text{n} = \langle \text{value} \rangle \) and \( \text{k} = \langle \text{value} \rangle \).
Constraint: if \( \text{side} = \text{Nag}_\text{LeftSide}, \text{m} \geq \text{k} \geq 0; \)
if \( \text{side} = \text{Nag}_\text{RightSide}, \text{n} \geq \text{k} \geq 0. \)

On entry, \( \text{side} = \langle \text{value} \rangle \), \( \text{m} = \langle \text{value} \rangle \), \( \text{pda} = \langle \text{value} \rangle \) and \( \text{n} = \langle \text{value} \rangle \).
Constraint: if \( \text{side} = \text{Nag}_\text{LeftSide}, \text{pda} \geq \max(1, \text{m}); \)
if \( \text{side} = \text{Nag}_\text{RightSide}, \text{pda} \geq \max(1, \text{n}). \)

**NE_INT**

On entry, \( \text{m} = \langle \text{value} \rangle \).
Constraint: \( \text{m} \geq 0. \)

On entry, \( \text{n} = \langle \text{value} \rangle \).
Constraint: \( \text{n} \geq 0. \)

On entry, \( \text{pda} = \langle \text{value} \rangle \).
Constraint: \( \text{pda} > 0. \)

On entry, \( \text{pdc} = \langle \text{value} \rangle \).
Constraint: \( \text{pdc} > 0. \)
On entry, \( pda = \langle \text{value} \rangle \) and \( k = \langle \text{value} \rangle \).
Constraint: \( pda \geq \max(1, k) \).

On entry, \( pdc = \langle \text{value} \rangle \) and \( m = \langle \text{value} \rangle \).
Constraint: \( pdc \geq \max(1, m) \).

On entry, \( pdc = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pdc \geq \max(1, n) \).

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.

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.

The computed result differs from the exact result by a matrix \( E \) such that
\[
\| E \|_2 = O \epsilon \| C \|_2
\]
where \( \epsilon \) is the \textit{machine precision}.

nag_zunmql (f08cuc) 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.

The total number of 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_dormql (f08cgc).

See Section 10 in nag_zgeqlf (f08csc).