NAG Library Function Document

nag_dormlq (f08akc)

1 Purpose

nag_dormlq (f08akc) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from an LQ factorization computed by nag_dgelqf (f08ahc).

2 Specification

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

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

3 Description

nag_dormlq (f08akc) is intended to be used after a call to nag_dgelqf (f08ahc), which performs an LQ factorization of a real matrix A. The orthogonal matrix Q is represented as a product of elementary reflectors.

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

\[ QC, Q^T C, CQ \text{ or } CQ^T, \]

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

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

*Input*

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

*Constraint:* \text{order} = Nag_RowMajor or Nag_ColMajor.

2: \( \text{side} \) – Nag_SideType

*Input*

*On entry:* indicates how Q or Q\(^T\) is to be applied to C.

\text{side} = Nag_LeftSide
\quad Q \text{ or } Q^T \text{ is applied to } C \text{ from the left.}

\text{side} = Nag_RightSide
\quad Q \text{ or } Q^T \text{ is applied to } C \text{ from the right.}

*Constraint:* \text{side} = Nag_LeftSide or Nag_RightSide.
trans – Nag_TransType

*On entry:* indicates whether \( Q \) or \( Q^T \) is to be applied to \( C \).

\[
\text{trans} = \text{Nag\_NoTrans} \quad Q \text{ is applied to } C.
\]

\[
\text{trans} = \text{Nag\_Trans} \quad Q^T \text{ is applied to } C.
\]

*Constraint:* \( \text{trans} = \text{Nag\_NoTrans} \) or \( \text{Nag\_Trans} \).

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:*

\[
\begin{align*}
\text{if } & \text{side} = \text{Nag\_LeftSide}, \quad m \geq k \geq 0; \\
\text{if } & \text{side} = \text{Nag\_RightSide}, \quad n \geq k \geq 0.
\end{align*}
\]

7: \( \text{a}[\text{dim}] \) – const double

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

\[
\begin{align*}
\max(1, pda \times m) & \text{ when } \text{side} = \text{Nag\_LeftSide} \text{ and } \text{order} = \text{Nag\_ColMajor}; \\
\max(1, k \times pda) & \text{ when } \text{side} = \text{Nag\_LeftSide} \text{ and } \text{order} = \text{Nag\_RowMajor}; \\
\max(1, pda \times n) & \text{ when } \text{side} = \text{Nag\_RightSide} \text{ and } \text{order} = \text{Nag\_ColMajor}; \\
\max(1, k \times pda) & \text{ when } \text{side} = \text{Nag\_RightSide} \text{ and } \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

*On entry:* details of the vectors which define the elementary reflectors, as returned by nag_dgelqf (f08ahc).

8: \( pda \) – Integer

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

*Constraints:*

\[
\begin{align*}
\text{if } & \text{order} = \text{Nag\_ColMajor}, \quad pda \geq \max(1, k); \\
\text{if } & \text{order} = \text{Nag\_RowMajor}, \\
\text{if } & \text{side} = \text{Nag\_LeftSide}, \quad pda \geq \max(1, m); \\
\text{if } & \text{side} = \text{Nag\_RightSide}, \quad pda \geq \max(1, n).
\end{align*}
\]

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

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

*On entry:* further details of the elementary reflectors, as returned by nag_dgelqf (f08ahc).
10: \( \mathbf{c}[\text{dim}] \) – double

**Input/Output**

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

The \((i,j)\)th element of the matrix \( \mathbf{C} \) is stored in
\[
\begin{align*}
\mathbf{c}[j-1 \times pdc + i - 1] & \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor}; \\
\mathbf{c}[(i-1) \times pdc + j - 1] & \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

*On entry:* the \( m \) by \( n \) matrix \( \mathbf{C} \).

*On exit:* \( \mathbf{c} \) is overwritten by \( \mathbf{Q} \mathbf{C} \) or \( \mathbf{Q}^\top \mathbf{C} \) or \( \mathbf{C} \mathbf{Q} \) or \( \mathbf{C} \mathbf{Q}^\top \) as specified by \( \text{side} \) and \( \text{trans} \).

11: \( \text{pdc} \) – Integer

**Input**

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

**Constraints:**
\[
\begin{align*}
\text{if} \quad \text{order} = \text{Nag\_ColMajor}, \quad \text{pdc} & \geq \max(1, m) \\
\text{if} \quad \text{order} = \text{Nag\_RowMajor}, \quad \text{pdc} & \geq \max(1, n).
\end{align*}
\]

12: \( \text{fail} \) – NagError*

**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, \( \text{side} = \langle \text{value} \rangle \), \( m = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \) and \( k = \langle \text{value} \rangle \).

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

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

On entry, \( \text{side} = \langle \text{value} \rangle \), \( \text{pda} = \langle \text{value} \rangle \), \( m = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

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

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

#### NE_INT

On entry, \( m = \langle \text{value} \rangle \).

Constraint: \( m \geq 0 \).

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).

Constraint: \( \text{pda} > 0 \).

On entry, \( \text{pdc} = \langle \text{value} \rangle \).

Constraint: \( \text{pdc} > 0 \).
NE_INT_2
On entry, pda = \langle value \rangle and k = \langle value \rangle.
Constraint: pda \geq \max(1, k).

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

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

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_dormlq (f08akc) is not threaded by NAG in any implementation.

nag_dormlq (f08akc) 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
\[ 2nk(2m - k) \text{ if side} = \text{Nag}_\text{LeftSide} \]
and
\[ 2mk(2n - k) \text{ if side} = \text{Nag}_\text{RightSide}. \]
The complex analogue of this function is nag_zunmlq (f08axc).

10 Example
See Section 10 in nag_dgelqf (f08ahc).