NAG C 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
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

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

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

4 References


5 Parameters

1: order – Nag_OrderType  
   *Input*

   *On entry:* the `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 `order = Nag_RowMajor`. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

   *Constraint:* \( order = \text{Nag\_RowMajor} \) or \( \text{Nag\_ColMajor} \).

2: side – Nag_SideType  
   *Input*

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

   - if `side = Nag_LeftSide`, \( Q \) or \( Q^T \) is applied to \( C \) from the left;
   - if `side = Nag_RightSide`, \( Q \) or \( Q^T \) is applied to \( C \) from the right.

   *Constraint:* \( side = \text{Nag\_LeftSide} \) or \( \text{Nag\_RightSide} \).

3: trans – Nag_TransType  
   *Input*

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

   - if `trans = Nag_NoTrans`, \( Q \) is applied to \( C \);
   - if `trans = Nag_Trans`, \( Q^T \) is applied to \( C \).

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

Input

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

Constraint: \( m \geq 0 \).

5: \( n \) – Integer 

Input

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

Constraint: \( n \geq 0 \).

6: \( k \) – Integer 

Input

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

Constraints:

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

7: \( a[\text{dim}] \) – double 

Input/Output

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

- \( \max(1, \text{pda} \times m) \) when \( \text{side} = \text{Nag\_LeftSide} \) and \( \text{order} = \text{Nag\_ColMajor} \);
- \( \max(1, \text{pda} \times k) \) when \( \text{side} = \text{Nag\_LeftSide} \) and \( \text{order} = \text{Nag\_RowMajor} \);
- \( \max(1, \text{pda} \times n) \) when \( \text{side} = \text{Nag\_RightSide} \) and \( \text{order} = \text{Nag\_ColMajor} \);
- \( \max(1, \text{pda} \times k) \) when \( \text{side} = \text{Nag\_RightSide} \) and \( \text{order} = \text{Nag\_RowMajor} \).

If \( \text{order} = \text{Nag\_ColMajor} \), the \((i,j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times \text{pda} + i - 1] \). 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 nag_dgelqf (f08ahc).

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

8: \( \text{pda} \) – Integer 

Input

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

Constraints:

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

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

Input

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

Input/Output

Note: the dimension, \( \text{dim} \), of the array \( \text{c} \) 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 \( c[(j-1) \times \text{pdc} + i - 1] \). 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 \( Q^T C \) or \( CQ \) or \( CQ^T \) as specified by \( \text{side} \) and \( \text{trans} \).


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 ≥ max(1, m);
if order = Nag_RowMajor, pdc ≥ 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 = value.
Constraint: m ≥ 0.

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

On entry, pda = value.
Constraint: pda > 0.

On entry, pdc = value.
Constraint: pdc > 0.

**NE_INT_2**

On entry, pda = value, k = value.
Constraint: pda ≥ max(1, k).

On entry, pdc = value, m = value.
Constraint: pdc ≥ max(1, m).

On entry, pdc = value, n = value.
Constraint: pdc ≥ max(1, n).

**NE_ENUM_INT_3**

On entry, side = value, m = value, n = value, k = value.
Constraint: if side = Nag_LeftSide, m ≥ k ≥ 0;
if side = Nag_RightSide, n ≥ k ≥ 0.

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

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter value 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 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_zunmlq (f08axc).

9 Example

See Section 9 of the document for nag_dgelqf (f08ahc).