NAG C Library Function Document

nag_dormqr (f08agc)

1 Purpose

nag_dormqr (f08agc) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from a QR factorization computed by nag_dgeqrf (f08aec) or nag_dgeqpf (f08bec).

2 Specification

```c
void nag_dormqr (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_dormqr (f08agc) is intended to be used after a call to nag_dgeqrf (f08aec) or nag_dgeqpf (f08bec), which perform a QR 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

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

overwriting the result on c (which may be any real 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_dgeqrf (f08aec).

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 = Nag_RowMajor or 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 = Nag_LeftSide or Nag_RightSide.

3: trans – Nag_TransType

*Input*

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

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

4: \( m \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} \( m \), the number of rows of the matrix \( C \).
\textit{Constraint:} \( m \geq 0 \).

5: \( n \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} \( n \), the number of columns of the matrix \( C \).
\textit{Constraint:} \( n \geq 0 \).

6: \( k \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} \( k \), the number of elementary reflectors whose product defines the matrix \( Q \).
\textit{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 \hspace{1cm} \text{Input/Output}

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

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

\textit{On entry:} details of the vectors which define the elementary reflectors, as returned by \text{nag\_dgeqrf} (\text{f08aec}) or \text{nag\_dgeqpf} (\text{f08bec}).

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

8: \( pda \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( a \).
\textit{Constraints:}
- if \( \text{order} = \text{Nag\_ColMajor} \),
  - if \( \text{side} = \text{Nag\_LeftSide} \), \( pda \geq \max(1, m) \);
  - if \( \text{side} = \text{Nag\_RightSide} \), \( pda \geq \max(1, n) \);
- if \( \text{order} = \text{Nag\_RowMajor} \), \( pda \geq \max(1, k) \).

9: \( tau[\text{dim}] \) – const double \hspace{1cm} \text{Input}

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

\textit{On entry:} further details of the elementary reflectors, as returned by \text{nag\_dgeqrf} (\text{f08aec}) or \text{nag\_dgeqpf} (\text{f08bec}).

10: \( c[\text{dim}] \) – double \hspace{1cm} \text{Input/Output}

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

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

11: \textbf{pdc} – Integer

\textit{Input}

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

Constraints:

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

12: \textbf{fail} – NagError *

\textit{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{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: \( \text{pdc} > 0 \).

\textbf{NE\_INT\_2}

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

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

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

\textbf{NE\_ENUM\_INT\_3}

On entry, \( \text{side} = \langle \text{value} \rangle, \text{ m} = \langle \text{value} \rangle, \text{ n} = \langle \text{value} \rangle, \text{ k} = \langle \text{value} \rangle \).
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, \( \text{side} = \langle \text{value} \rangle, \text{ m} = \langle \text{value} \rangle, \text{ n} = \langle \text{value} \rangle, \text{ pda} = \langle \text{value} \rangle \).
Constraint: if \( \text{side} = \text{Nag\_LeftSide}, \text{ pda} \geq \max(1, m) \); if \( \text{side} = \text{Nag\_RightSide}, \text{ pda} \geq \max(1, n) \).

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_BAD\_PARAM}

On entry, parameter \( \langle \text{value} \rangle \) 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 floating-point operations is approximately $2nk(2m - k)$ if $\text{side} = \text{Nag\_LeftSide}$ and $2mk(2n - k)$ if $\text{side} = \text{Nag\_RightSide}$.

The complex analogue of this function is nag_zunmqr (f08auc).

9 Example

See Section 9 of the document for nag_dgeqrf (f08aec).