1 Purpose

nag_dorgqr (f08afc) generates all or part of the real orthogonal matrix $Q$ from a QR factorization computed by nag_dgeqrf (f08aec) or nag_dgeqpf (f08bec).

2 Specification

```c
void nag_dorgqr (Nag_OrderType order, Integer m, Integer n, Integer k, double a[],
                 Integer pda, const double tau[], NagError *fail)
```

3 Description

nag_dorgqr (f08afc) 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 generate $Q$ explicitly as a square matrix, or to form only its leading columns. Usually $Q$ is determined from the QR factorization of an $m$ by $p$ matrix $A$ with $m \geq p$. The whole of $Q$ may be computed by:

```
nag_dorgqr (order,m,m,p,&a,pda,tau,&fail)
```

(note that the array $a$ must have at least $m$ columns) or its leading $p$ columns by:

```
nag_dorgqr (order,m,p,p,&a,pda,tau,&fail)
```

The columns of $Q$ returned by the last call form an orthonormal basis for the space spanned by the columns of $A$; thus nag_dgeqrf (f08aec) followed by nag_dorgqr (f08afc) can be used to orthogonalise the columns of $A$.

The information returned by the QR factorization functions also yields the QR factorization of the leading $k$ columns of $A$, where $k < p$. The orthogonal matrix arising from this factorization can be computed by:

```
nag_dorgqr (order,m,m,k,&a,pda,tau,&fail)
```

or its leading $k$ columns by:

```
nag_dorgqr (order,m,k,k,&a,pda,tau,&fail)
```

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:  \( m \) – Integer  \( \text{Input} \)
    
    \text{On entry:} m, the order of the orthogonal matrix \( Q \).

    \text{Constraint:} m \geq 0.

3:  \( n \) – Integer  \( \text{Input} \)
    
    \text{On entry:} n, the number of columns of matrix \( Q \) that are required.

    \text{Constraint:} m \geq n \geq 0.

4:  \( k \) – Integer  \( \text{Input} \)
    
    \text{On entry:} k, the number of elementary reflectors whose product defines the matrix \( Q \).

    \text{Constraint:} n \geq k \geq 0.

5:  \( a[\text{dim}] \) – double  \( \text{Input/Output} \)
    
    \text{Note:} the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \max(1, \text{pda} \times n) \) when \( \text{order} = \text{Nag\_ColMajor} \) and at least \( \max(1, \text{pda} \times m) \) when \( \text{order} = \text{Nag\_RowMajor} \).

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

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

    \text{On exit:} the \( m \) by \( n \) matrix \( Q \).

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

    \text{Constraints:}

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

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

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

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

8:  \( \text{fail} \) – NagError *  \( \text{Output} \)

    The NAG error * parameter (see the Essential Introduction).

6  \text{Error Indicators and Warnings}

**NE\_INT**

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

Constraint: \( m \geq 0 \).

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

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

**NE\_INT\_2**

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

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

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

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

**NE_ALLOC_FAIL**
Memory allocation failed.

**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 matrix \( Q \) differs from an exactly orthogonal matrix by a matrix \( E \) such that
\[
\| E \|_2 = O(\epsilon),
\]
where \( \epsilon \) is the *machine precision*.

**8 Further Comments**
The total number of floating-point operations is approximately
\[
4mnk - 2(m + n)k^2 + \frac{4}{3}k^3; \quad \text{when} \quad n = k,
\]
the number is approximately \( \frac{2}{3}n^2(3m - n) \).

The complex analogue of this function is nag_zungqr (f08atc).

**9 Example**
To form the leading 4 columns of the orthogonal matrix \( Q \) from the QR factorization of the matrix \( A \), where
\[
A = \begin{pmatrix}
-0.57 & -1.28 & -0.39 & 0.25 \\
-1.93 & 1.08 & -0.31 & -2.14 \\
2.30 & 0.24 & 0.40 & -0.35 \\
-1.93 & 0.64 & -0.66 & 0.08 \\
0.15 & 0.30 & 0.15 & -2.13 \\
-0.02 & 1.03 & -1.43 & 0.50
\end{pmatrix}.
\]
The columns of \( Q \) form an orthonormal basis for the space spanned by the columns of \( A \).

**9.1 Program Text**
/* nag_dorgqr (f08afc) Example Program.*/
* * Copyright 2001 Numerical Algorithms Group.*
* * Mark 7, 2001.*
*/
#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer i, j, m, n, pda, tau_len;
  Integer exit_status=0;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  char *title=0;
  double *a=0, *tau=0;

  #ifdef NAG_COLUMN_MAJOR
  #define A(I,J) a[(J-1)*pda+I-1]
  order = Nag_ColMajor;
  #else
  #define A(I,J) a[(I-1)*pda+J-1]
  order = Nag_RowMajor;
  #endif

  INIT_FAIL(fail);
  Vprintf("f08afc Example Program Results

");

  /* Skip heading in data file */
  Vscanf("%*[^
] ");
  Vscanf("%ld%ld%*[^
] ", &m, &n);
  #ifdef NAG_COLUMN_MAJOR
  pda = m;
  #else
  pda = n;
  #endif
  tau_len = MIN(m, n);

  /* Allocate memory */
  if ( !(title = NAG_ALLOC(31, char)) ||
      !(a = NAG_ALLOC(m * n, double)) ||
      !(tau = NAG_ALLOC(tau_len, double)) )
  {
    Vprintf("Allocation failure
");
    exit_status = -1;
    goto END;
  }

  /* Read A from data file */
  for (i = 1; i <= m; ++i)
  {
    for (j = 1; j <= n; ++j)
      Vscanf("%lf", &A(i,j));
  }
  Vscanf("%*[^
] ");

  /* Compute the QR factorization of A */
  f08aec(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
  {
    Vprintf("Error from f08aec.
%s
", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Form the leading N columns of Q explicitly */
  f08afc(order, m, n, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
  {
    Vprintf("Error from f08afc.
%s
", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print the leading N columns of Q only */
  Vsprintf(title, "The leading %2ld columns of Q
");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n, a, pda, title, 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (title) NAG_FREE(title);
if (a) NAG_FREE(a);
if (tau) NAG_FREE(tau);

return exit_status;
}

9.2 Program Data

f08afc Example Program Data
6 4 :Values of M and N
-0.57 -1.28 -0.39 0.25
-1.93 1.08 -0.31 -2.14
2.30 0.24 0.40 -0.35
-1.93 0.64 -0.66 0.08
0.15 0.30 0.15 -2.13
-0.02 1.03 -1.43 0.50 :End of matrix A

9.3 Program Results

f08afc Example Program Results
The leading 4 columns of Q

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.1576</td>
<td>0.6744</td>
<td>-0.4571</td>
<td>0.4489</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.5335</td>
<td>-0.3861</td>
<td>0.2583</td>
<td>0.3898</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.6358</td>
<td>-0.2928</td>
<td>0.0165</td>
<td>0.1930</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.5335</td>
<td>-0.1692</td>
<td>-0.0834</td>
<td>-0.2350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.0415</td>
<td>-0.1593</td>
<td>0.1475</td>
<td>0.7436</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.0055</td>
<td>-0.5064</td>
<td>-0.8339</td>
<td>0.0335</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>