

NAG C Library Function Document

nag_dorgqr (f08afc)

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

```
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

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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 *Input*
On entry: m , the order of the orthogonal matrix Q .
Constraint: $m \geq 0$.
- 3: **n** – Integer *Input*
On entry: n , the number of columns of matrix Q that are required.
Constraint: $m \geq n \geq 0$.
- 4: **k** – Integer *Input*
On entry: k , the number of elementary reflectors whose product defines the matrix Q .
Constraint: $n \geq k \geq 0$.
- 5: **a**[dim] – double *Input/Output*
Note: the dimension, dim , of the array **a** must be at least $\max(1, pda \times n)$ when **order** = **Nag_ColMajor** and at least $\max(1, pda \times m)$ when **order** = **Nag_RowMajor**.
If **order** = **Nag_ColMajor**, the (i, j) th element of the matrix A is stored in **a**[($j - 1$) $\times$ **pda** + $i - 1$] and if **order** = **Nag_RowMajor**, the (i, j) th element of the matrix A is stored in **a**[($i - 1$) $\times$ **pda** + $j - 1$].
On entry: details of the vectors which define the elementary reflectors, as returned by nag_dgeqrf (f08aec) or nag_dgeqpf (f08bec).
On exit: the m by n matrix Q .
- 6: **pda** – Integer *Input*
On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **a**.
Constraints:
if **order** = **Nag_ColMajor**, $pda \geq \max(1, m)$;
if **order** = **Nag_RowMajor**, $pda \geq \max(1, n)$.
- 7: **tau**[dim] – const double *Input*
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_dgeqrf (f08aec) or nag_dgeqpf (f08bec).
- 8: **fail** – NagError * *Output*
The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** = $\langle value \rangle$.

Constraint: $m \geq 0$.

On entry, **pda** = $\langle value \rangle$.

Constraint: $pda > 0$.

NE_INT_2

On entry, **m** = $\langle value \rangle$, **n** = $\langle value \rangle$.

Constraint: $m \geq n \geq 0$.

On entry, **n** = $\langle value \rangle$, **k** = $\langle value \rangle$.

Constraint: **n** $\geq$ **k** $\geq$ 0.

On entry, **pda** = $\langle value \rangle$, **m** = $\langle value \rangle$.

Constraint: **pda** $\geq$ max(1, **m**).

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.

Constraint: **pda** $\geq$ max(1, **n**).

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ 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 ϵ is the *machine precision*.

8 Further Comments

The total number of floating-point operations is approximately $4mnk - 2(m+n)k^2 + \frac{4}{3}k^3$; when $n = k$, 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\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    Vscanf("%ld%ld%*[\n] ", &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\n");
        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("%*[\n] ");

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

    /* Print the leading N columns of Q only */
    Vsprintf(title, "The leading %2ld columns of Q\n", n);

```

```

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

	1	2	3	4
1	-0.1576	0.6744	-0.4571	0.4489
2	-0.5335	-0.3861	0.2583	0.3898
3	0.6358	-0.2928	0.0165	0.1930
4	-0.5335	-0.1692	-0.0834	-0.2350
5	0.0415	-0.1593	0.1475	0.7436
6	-0.0055	-0.5064	-0.8339	0.0335
