

## NAG Library Function Document

### nag\_dorgql (f08cfc)

#### 1 Purpose

nag\_dorgql (f08cfc) generates all or part of the real  $m$  by  $m$  orthogonal matrix  $Q$  from a  $QL$  factorization computed by nag\_dgeqlf (f08cec).

#### 2 Specification

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

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

#### 3 Description

nag\_dorgql (f08cfc) is intended to be used after a call to nag\_dgeqlf (f08cec), which performs a  $QL$  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 trailing columns.

Usually  $Q$  is determined from the  $QL$  factorization of an  $m$  by  $p$  matrix  $A$  with  $m \geq p$ . The whole of  $Q$  may be computed by:

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

(note that the array  $\mathbf{a}$  must have at least  $m$  columns) or its trailing  $p$  columns by:

```
nag_dorgql(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\_dgeqlf (f08cec) followed by nag\_dorgql (f08cfc) can be used to orthogonalize the columns of  $A$ .

The information returned by nag\_dgeqlf (f08cec) also yields the  $QL$  factorization of the trailing  $k$  columns of  $A$ , where  $k < p$ . The orthogonal matrix arising from this factorization can be computed by:

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

or its trailing  $k$  columns by:

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

#### 4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia <http://www.netlib.org/lapack/lug>

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

#### 5 Arguments

1: **order** – Nag\_OrderType *Input*

*On entry:* the **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

**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:* **order** = Nag\_RowMajor or Nag\_ColMajor.

- 2: **m** – Integer *Input*  
*On entry:*  $m$ , the number of rows of the matrix  $Q$ .  
*Constraint:*  $m \geq 0$ .
- 3: **n** – Integer *Input*  
*On entry:*  $n$ , the number of columns of the matrix  $Q$ .  
*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  
 $\max(1, \mathbf{pda} \times \mathbf{n})$  when **order** = Nag\_ColMajor;  
 $\max(1, \mathbf{m} \times \mathbf{pda})$  when **order** = Nag\_RowMajor.  
*On entry:* details of the vectors which define the elementary reflectors, as returned by nag\_dgeqlf (f08cec).  
*On exit:* the  $m$  by  $n$  matrix  $Q$ .  
If **order** = 'Nag\_ColMajor', the  $(i, j)$ th element of the matrix is stored in  $\mathbf{a}[(j - 1) \times \mathbf{pda} + i - 1]$ .  
If **order** = 'Nag\_RowMajor', the  $(i, j)$ th element of the matrix is stored in  $\mathbf{a}[(i - 1) \times \mathbf{pda} + j - 1]$ .
- 6: **pda** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **a**.  
*Constraints:*  
if **order** = Nag\_ColMajor,  $\mathbf{pda} \geq \max(1, \mathbf{m})$ ;  
if **order** = Nag\_RowMajor,  $\mathbf{pda} \geq \max(1, \mathbf{n})$ .
- 7: **tau**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **tau** must be at least  $\max(1, \mathbf{k})$ .  
*On entry:* further details of the elementary reflectors, as returned by nag\_dgeqlf (f08cec).
- 8: **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.

**NE\_BAD\_PARAM**

On entry, argument  $\langle value \rangle$  had an illegal value.

**NE\_INT**

On entry,  $\mathbf{m} = \langle value \rangle$ .

Constraint:  $\mathbf{m} \geq 0$ .

On entry,  $\mathbf{pda} = \langle value \rangle$ .

Constraint:  $\mathbf{pda} > 0$ .

**NE\_INT\_2**

On entry,  $\mathbf{m} = \langle value \rangle$  and  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{m} \geq \mathbf{n} \geq 0$ .

On entry,  $\mathbf{n} = \langle value \rangle$  and  $\mathbf{k} = \langle value \rangle$ .

Constraint:  $\mathbf{n} \geq \mathbf{k} \geq 0$ .

On entry,  $\mathbf{pda} = \langle value \rangle$  and  $\mathbf{m} = \langle value \rangle$ .

Constraint:  $\mathbf{pda} \geq \max(1, \mathbf{m})$ .

On entry,  $\mathbf{pda} = \langle value \rangle$  and  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{pda} \geq \max(1, \mathbf{n})$ .

**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.

**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 Parallelism and Performance**

nag\_dorgql (f08cfc) is not threaded by NAG in any implementation.

nag\_dorgql (f08cfc) 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 Users' Note for your implementation for any additional implementation-specific information.

**9 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\_zungql (f08ctc).

## 10 Example

This example generates the first four columns of the matrix  $Q$  of the  $QL$  factorization of  $A$  as returned by nag\_dgeqlf (f08cec), where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

### 10.1 Program Text

```

/* nag_dorgql (f08cfc) Example Program.
 *
 * Copyright 2011 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

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

#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);

    printf("nag_dorgql (f08cfc) Example Program Results\n\n");

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

#ifdef NAG_COLUMN_MAJOR
    pda = m;
#else
    pda = n;
#endif

    /* Allocate memory */
    if (!(title = NAG_ALLOC(31, char)) ||
        !(a = NAG_ALLOC(m*n, double)) ||
        !(tau = NAG_ALLOC(n, double)))
    {
        printf("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)
        scanf("%lf", &A(i, j));
scanf("%*[\n]");

/* nag_dgeqlf (f08cec).
 * Compute the QL factorization of A .
 */
nag_dgeqlf(order, m, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgeqlf (f08cec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_dorgql (f08cfc).
 * Form the leading n columns of Q explicitly.
 */
nag_dorgql(order, m, n, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dorgql (f08cfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Form the heading for x04cac */
sprintf(title, "The leading %4ld columns of Q", n);

/* nag_gen_real_mat_print (x04cac).
 * Print the leading n columns of Q.
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n, a,
                        pda, title, 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
           fail.message);
    exit_status = 1;
}

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

    return exit_status;
}

#undef A

```

## 10.2 Program Data

nag\_dorgql (f08cfc) 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

```

### 10.3 Program Results

nag\_dorgql (f08cfc) Example Program Results

The leading 4 columns of Q

	1	2	3	4
1	-0.0833	0.9100	-0.2202	-0.0809
2	0.2972	-0.1080	-0.2706	0.6922
3	-0.6404	-0.2351	0.2220	0.1132
4	0.4461	-0.1620	-0.3866	-0.0259
5	-0.2938	0.2022	0.0015	0.6890
6	-0.4575	-0.1946	-0.8243	-0.1617

---