NAG C Library Function Document

nag_dorglq (f08ajc)

1 Purpose

nag_dorglq (f08ajc) generates all or part of the real orthogonal matrix Q from an LQ factorization computed by nag_dgelqf (f08ahc).

2 Specification

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

3 Description

nag_dorglq (f08ajc) 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 generate Q explicitly as a square matrix, or to form only its leading rows.

Usually Q is determined from the LQ factorization of a p by n matrix A with $p \leq n$. The whole of Q may be computed by:

```
nag_dorglq (order,n,n,p,&a,pda,tau,&fail)
```

(note that the array **a** must have at least n rows) or its leading p rows by:

```
nag_dorglq (order,p,n,p,&a,pda,tau,&fail)
```

The rows of Q returned by the last call form an orthonormal basis for the space spanned by the rows of A; thus nag_dgelqf (f08ahc) followed by nag_dorglq (f08ajc) can be used to orthogonalise the rows of A.

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

```
nag_dorglq (order,n,n,k,&a,pda,tau,&fail)
```

or its leading k rows by:

```
nag_dorglq (order,k,n,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 number of rows of the matrix Q.

Constraint: **m** ≥ 0.

3: **n** – Integer *Input*

On entry: n, the number of columns of the matrix Q.

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

4: **k** – Integer *Input*

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

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

5: $\mathbf{a}[dim]$ – 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** and at least $\max(1, \mathbf{pda} \times \mathbf{m})$ when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix A is stored in $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix A is stored in $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$.

On entry: details of the vectors which define the elementary reflectors, as returned by nag_dgelqf (f08ahc).

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**, $\mathbf{pda} \geq \max(1, \mathbf{m})$;
if **order** = **Nag_RowMajor**, $\mathbf{pda} \geq \max(1, \mathbf{n})$.

7: $\mathbf{tau}[dim]$ – 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_dgelqf (f08ahc).

8: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

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, $\mathbf{n} = \langle value \rangle$, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq \mathbf{m}$.

On entry, $\mathbf{m} = \langle value \rangle$, $\mathbf{k} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq \mathbf{k} \geq 0$.

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

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

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 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 $m = k$, the number is approximately $\frac{2}{3}m^2(3n - m)$.

The complex analogue of this function is nag_zunglq (f08awc).

9 Example

To form the leading 4 rows of the orthogonal matrix Q from the LQ factorization of the matrix A, where

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix}.$$

The rows of Q form an orthonormal basis for the space spanned by the rows of A.

9.1 Program Text

```
/* nag_dorglq (f08ajc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#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, 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("f08ajc 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 = m;

  /* Allocate memory */
  if ( !(title = NAG_ALLOC(31, char)) ||
       !(a = NAG_ALLOC(m * n, double)) ||
       !(tau = NAG_ALLOC(m, 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 LQ factorization of A */
  f08ahc(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08ahc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Form the leading M rows of Q explicitly */
  f08ajc(order, m, n, m, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08ajc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

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

```
f08ajc Example Program Data
  4  6                                            :Values of M and N
 -5.42   3.28  -3.68   0.27   2.06   0.46
 -1.65  -3.40  -3.20  -1.03  -4.06  -0.01
 -0.37   2.35   1.90   4.31  -1.76   1.13
 -3.15  -0.11   1.99  -2.70   0.26   4.50   :End of matrix A
```

9.3 Program Results

```
f08ajc Example Program Results

 The leading  4 rows of Q

          1        2        3        4        5        6
 1  -0.7104   0.4299  -0.4824   0.0354   0.2700   0.0603
 2  -0.2412  -0.5323  -0.4845  -0.1595  -0.6311  -0.0027
 3   0.1287  -0.2619  -0.2108  -0.7447   0.5227  -0.2063
 4  -0.3403  -0.0921   0.4546  -0.3869  -0.0465   0.7191
```