NAG Library Function Document

nag_dorgrq (f08cjc)

1 Purpose

nag_dorgrq (f08cjc) generates all or part of the real n by n orthogonal matrix Q from an RQ factorization computed by nag_dgerqf (f08chc).

2 Specification

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

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

3 Description

nag_dorgrq (f08cjc) is intended to be used following a call to nag_dgerqf (f08chc), which performs an RQ factorization of a real matrix A and represents the orthogonal matrix Q as a product of k elementary reflectors of order n.

This function may be used to generate Q explicitly as a square matrix, or to form only its trailing rows.

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

```
nag_dorgrq(order,n,n,p,a,pda,tau,info)
```

(note that the matrix A must have at least n rows), or its trailing p rows as:

```
nag_dorgrq(order,p,n,p,a,pda,tau,info)
```

The rows of Q returned by the last call form an orthonormal basis for the space spanned by the rows of A; thus nag_dgerqf (f08chc) followed by nag_dorgrq (f08cjc) can be used to orthogonalize the rows of A.

The information returned by nag_dgerqf (f08chc) also yields the RQ factorization of the trailing k rows of A, where $k < p$. The orthogonal matrix arising from this factorization can be computed by:

```
nag_dorgrq(order,n,n,k,a,pda,tau,info)
```

or its leading k columns by:

```
nag_dorgrq(order,k,n,k,a,pda,tau,info)
```

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

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: **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_dgerqf (f08chc).

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: $\mathbf{tau}[i-1]$ must contain the scalar factor of the elementary reflector H_i, as returned by nag_dgerqf (f08chc).

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, **m** $= \langle value \rangle$.
Constraint: **m** ≥ 0.

On entry, **pda** $= \langle value \rangle$.
Constraint: **pda** > 0.

NE_INT_2

On entry, **m** $= \langle value \rangle$ and **k** $= \langle value \rangle$.
Constraint: **m** $\geq$ **k** ≥ 0.

On entry, **n** $= \langle value \rangle$ and **m** $= \langle value \rangle$.
Constraint: **n** $\geq$ **m**.

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

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

$$\|E\|_2 = O\epsilon$$

and ϵ is the ***machine precision***.

8 Parallelism and Performance

nag_dorgrq (f08cjc) is not threaded by NAG in any implementation.

nag_dorgrq (f08cjc) 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 $m=k$ this becomes $\frac{2}{3}m^2(3n-m)$.

The complex analogue of this function is nag_zungrq (f08cwc).

10 Example

This example generates the first four rows of the matrix Q of the RQ factorization of A as returned by nag_dgerqf (f08chc), where

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

10.1 Program Text

```
/* nag_dorgrq (f08cjc) 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_dorgrq (f08cjc) 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(27, char)) ||
      !(a = NAG_ALLOC(m*n, double)) ||
      !(tau = NAG_ALLOC(m, 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_dgerqf (f08chc).
   * Compute the RQ factorization of A.
   */
  nag_dgerqf(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgerqf (f08chc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_dorgrq (f08cjc).
   * Form the leading m rows of Q explicitly.
   */
  nag_dorgrq(order, m, n, m, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dorgrq (f08cjc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

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

  /* nag_gen_real_mat_print (x04cac).
   * Print the leading m rows 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_dorgrq (f08cjc) Example Program Data

  4      6                                :Values of m and n

 -0.57  -1.93   2.30  -1.93   0.15  -0.02
 -1.28   1.08   0.24   0.64   0.30   1.03
 -0.39  -0.31   0.40  -0.66   0.15  -1.43
  0.25  -2.14  -0.35   0.08  -2.13   0.50 :End of matrix A
```

10.3 Program Results

```
nag_dorgrq (f08cjc) Example Program Results

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