NAG 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), nag_dgeqpf (f08bec) or nag_dgeqp3 (f08bfc).

2 Specification

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

```
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), nag_dgeqpf (f08bec) or nag_dgeqp3 (f08bfc). 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 orthogonalize 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 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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 order of the orthogonal matrix Q.

Constraint: **m** ≥ 0.

3: **n** – Integer *Input*

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

Constraint: **m** $\geq$ **n** ≥ 0.

4: **k** – Integer *Input*

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

Constraint: **n** $\geq$ **k** ≥ 0.

5: **a**[*dim*] – double *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least

max(1, **pda** × **n**) when **order** = Nag_ColMajor;
max(1, **m** × **pda**) when **order** = Nag_RowMajor.

On entry: details of the vectors which define the elementary reflectors, as returned by nag_dgeqrf (f08aec), nag_dgeqpf (f08bec) or nag_dgeqp3 (f08bfc).

On exit: the m by n matrix Q.

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

If **order** = Nag_RowMajor, the (i, j)th element of the matrix is stored in **a**$[(i-1) \times$ **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, **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), nag_dgeqpf (f08bec) or nag_dgeqp3 (f08bfc).

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

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

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

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

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 Parallelism and Performance

nag_dorgqr (f08afc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dorgqr (f08afc) 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 x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also 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_zungqr (f08atc).

10 Example

This example forms 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.

10.1 Program Text

```
/* nag_dorgqr (f08afc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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_LOAD_FP
  /* The following line is needed to force the Microsoft linker
     to load floating point support */
  float force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

#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_dorgqr (f08afc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &m, &n);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &m, &n);
#endif
#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)))
  {
    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)
#ifdef _WIN32
      scanf_s("%lf", &A(i, j));
#else
      scanf("%lf", &A(i, j));
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Compute the QR factorization of A */
  /* nag_dgeqrf (f08aec).
   * QR factorization of real general rectangular matrix
   */
  nag_dgeqrf(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgeqrf (f08aec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Form the leading N columns of Q explicitly */
  /* nag_dorgqr (f08afc).
   * Form all or part of orthogonal Q from QR factorization
   * determined by nag_dgeqrf (f08aec) or nag_dgeqpf (f08bec)
   */
  nag_dorgqr(order, m, n, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dorgqr (f08afc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Print the leading N columns of Q only */
#ifdef _WIN32
  sprintf_s(title, 31, "The leading %2" NAG_IFMT " columns of Q\n", n);
```

```
#else
  sprintf(title, "The leading %2" NAG_IFMT " columns of Q\n", n);
#endif
  /* nag_gen_real_mat_print (x04cac).
   * Print real general matrix (easy-to-use)
   */
  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;
    goto END;
  }

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

  return exit_status;
}
```

10.2 Program Data

```
nag_dorgqr (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
```

10.3 Program Results

```
nag_dorgqr (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
```