NAG Library Function Document

nag_dgeqrf (f08aec)

1 Purpose

nag_dgeqrf (f08aec) computes the QR factorization of a real m by n matrix.

2 Specification

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

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

3 Description

nag_dgeqrf (f08aec) forms the QR factorization of an arbitrary rectangular real m by n matrix. No pivoting is performed.

If $m \geq n$, the factorization is given by:

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where R is an n by n upper triangular matrix and Q is an m by m orthogonal matrix. It is sometimes more convenient to write the factorization as

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix},$$

which reduces to

$$A = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m - n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$A = Q\begin{pmatrix} R_1 & R_2 \end{pmatrix},$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m, n)$ elementary reflectors (see the f08 Chapter Introduction for details). Functions are provided to work with Q in this representation (see Section 9).

Note also that for any $k < n$, the information returned in the first k columns of the array **a** represents a QR factorization of the first k columns of the original matrix A.

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 number of rows of the matrix A.

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

3: **n** – Integer *Input*

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

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

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

The (i, j)th element of the matrix A is stored in

$\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the orthogonal matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the orthogonal matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

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

6: $\mathbf{tau}[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **tau** must be at least $\max(1, \min(\mathbf{m}, \mathbf{n}))$.

On exit: further details of the orthogonal matrix Q.

7: **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, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 0$.

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

On entry, $\mathbf{pda} = \langle value \rangle$.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

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.

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 factorization is the exact factorization of a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Parallelism and Performance

nag_dgeqrf (f08aec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dgeqrf (f08aec) 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 $\frac{2}{3}n^2(3m - n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n - m)$ if $m < n$.

To form the orthogonal matrix Q nag_dgeqrf (f08aec) may be followed by a call to nag_dorgqr (f08afc):

```
nag_dorgqr(order,m,m,MIN(m,n),&a,pda,tau,&fail)
```

but note that the second dimension of the array **a** must be at least **m**, which may be larger than was required by nag_dgeqrf (f08aec).

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
nag_dorgqr(order,m,n,n,&a,pda,tau,&fail)
```

To apply Q to an arbitrary real rectangular matrix C, nag_dgeqrf (f08aec) may be followed by a call to nag_dormqr (f08agc). For example,

```
nag_dormqr(order,Nag_LeftSide,Nag_Trans,m,p,MIN(m,n),&a,pda,tau,
 + &c,pdc,&fail)
```

forms $C = Q^{\mathrm{T}}C$, where C is m by p.

To compute a QR factorization with column pivoting, use nag_dtpqrt (f08bbc) or nag_dgeqpf (f08bec).

The complex analogue of this function is nag_zgeqrf (f08asc).

10 Example

This example solves the linear least squares problems

$$\text{minimize} \|Ax_i - b_i\|_2, \quad i = 1, 2$$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -3.15 & 2.19 \\ -0.11 & -3.64 \\ 1.99 & 0.57 \\ -2.70 & 8.23 \\ 0.26 & -6.35 \\ 4.50 & -1.48 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_dgeqrf (f08aec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer i, j, m, n, nrhs, pda, pdb, tau_len;
  Integer exit_status = 0;
  NagError fail;
```

```
  Nag_OrderType order;
  /* Arrays */
  double *a = 0, *b = 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]
#define B(I, J) b[(J - 1) * pdb + I - 1]
  order = Nag_ColMajor;
#else
#define A(I, J) a[(I - 1) * pda + J - 1]
#define B(I, J) b[(I - 1) * pdb + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_dgeqrf (f08aec) 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 "%" NAG_IFMT "%*[^\n] ", &m, &n, &nrhs);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &m, &n, &nrhs);
#endif
#ifdef NAG_COLUMN_MAJOR
  pda = m;
  pdb = m;
#else
  pda = n;
  pdb = nrhs;
#endif
  tau_len = MIN(m, n);

  /* Allocate memory */
  if (!(a = NAG_ALLOC(m * n, double)) ||
      !(b = NAG_ALLOC(m * nrhs, double)) ||
      !(tau = NAG_ALLOC(tau_len, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  /* Read A and B 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
  for (i = 1; i <= m; ++i) {
    for (j = 1; j <= nrhs; ++j)
#ifdef _WIN32
      scanf_s("%lf", &B(i, j));
#else
```

```
      scanf("%lf", &B(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;
  }

  /* Compute C = (Q^T)*B, storing the result in B */
  /* nag_dormqr (f08agc).
   * Apply orthogonal transformation determined by nag_dgeqrf (f08aec)
   * or nag_dgeqpf (f08bec)
   */
  nag_dormqr(order, Nag_LeftSide, Nag_Trans, m, nrhs, n, a, pda,
             tau, b, pdb, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dormqr (f08agc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Compute least squares solution by back-substitution in R*X = C */
  /* nag_dtrtrs (f07tec).
   * Solution of real triangular system of linear equations,
   * multiple right-hand sides
   */
  nag_dtrtrs(order, Nag_Upper, Nag_NoTrans, Nag_NonUnitDiag, n, nrhs,
             a, pda, b, pdb, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dtrtrs (f07tec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Print least squares solution(s) */
  /* 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, n, nrhs,
                         b, pdb, "Least squares solution(s)", 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(a);
  NAG_FREE(b);
  NAG_FREE(tau);
  return exit_status;
}
```

10.2 Program Data

```
nag_dgeqrf (f08aec) Example Program Data
  6  4  2                              :Values of M, N and NRHS
 -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
 -3.15   2.19
 -0.11  -3.64
  1.99   0.57
 -2.70   8.23
  0.26  -6.35
  4.50  -1.48                 :End of matrix B
```

10.3 Program Results

```
nag_dgeqrf (f08aec) Example Program Results

 Least squares solution(s)
            1          2
 1      1.5146    -1.5838
 2      1.8621     0.5536
 3     -1.4467     1.3491
 4      0.0396     2.9600
```