

# NAG Library Function Document

## nag\_dgelqf (f08ahc)

### 1 Purpose

nag\_dgelqf (f08ahc) computes the  $LQ$  factorization of a real  $m$  by  $n$  matrix.

### 2 Specification

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

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

### 3 Description

nag\_dgelqf (f08ahc) forms the  $LQ$  factorization of an arbitrary rectangular real  $m$  by  $n$  matrix. No pivoting is performed.

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

$$A = (L \ 0)Q$$

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

$$A = (L \ 0) \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

which reduces to

$$A = LQ_1,$$

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

If  $m > n$ ,  $L$  is trapezoidal, and the factorization can be written

$$A = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} Q$$

where  $L_1$  is lower triangular and  $L_2$  is rectangular.

The  $LQ$  factorization of  $A$  is essentially the same as the  $QR$  factorization of  $A^T$ , since

$$A = (L \ 0)Q \Leftrightarrow A^T = Q^T \begin{pmatrix} L^T \\ 0 \end{pmatrix}.$$

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  $k < m$ , the information returned in the first  $k$  rows of the array **a** represents an  $LQ$  factorization of the first  $k$  rows of the original matrix  $A$ .

### 4 References

None.

## 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:*  $m \geq 0$ .
- 3: **n** – Integer *Input*  
*On entry:*  $n$ , the number of columns of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .
- 4: **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  $m \leq n$ , the elements above the diagonal are overwritten by details of the orthogonal matrix  $Q$  and the lower triangle is overwritten by the corresponding elements of the  $m$  by  $m$  lower triangular matrix  $L$ .  
If  $m > n$ , the strictly upper 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  $n$  lower trapezoidal matrix  $L$ .
- 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, **pda**  $\geq \max(1, \mathbf{m})$ ;  
if **order** = Nag\_RowMajor, **pda**  $\geq \max(1, \mathbf{n})$ .
- 6: **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  $\epsilon$  is the *machine precision*.

## 8 Parallelism and Performance

nag\_dgelqf (f08ahc) 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  $\frac{2}{3}m^2(3n - m)$  if  $m \leq n$  or  $\frac{2}{3}n^2(3m - n)$  if  $m > n$ .

To form the orthogonal matrix  $Q$  `nag_dgelqf` (f08ahc) may be followed by a call to `nag_dorglq` (f08ajc):

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

but note that the first dimension of the array **a**, specified by the argument **pda**, must be at least **n**, which may be larger than was required by `nag_dgelqf` (f08ahc).

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

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

To apply  $Q$  to an arbitrary real rectangular matrix  $C$ , `nag_dgelqf` (f08ahc) may be followed by a call to `nag_dormlq` (f08akc). For example,

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

forms the matrix product  $C = Q^T C$ , where  $C$  is  $m$  by  $p$ .

The complex analogue of this function is `nag_zgelqf` (f08avc).

## 10 Example

This example finds the minimum norm solutions of the under-determined systems of linear equations

$$Ax_1 = b_1 \quad \text{and} \quad Ax_2 = b_2$$

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

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -2.87 & -5.23 \\ 1.63 & 0.29 \\ -3.52 & 4.76 \\ 0.45 & -8.41 \end{pmatrix}.$$

### 10.1 Program Text

```
/* nag_dgelqf (f08ahc) 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_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_dgelsq (f08ahc) 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 = n;
#else
    pda = n;
    pdb = nrhs;
#endif

    tau_len = MIN(m, n);

    /* Allocate memory */
    if (!(a = NAG_ALLOC(m * n, double)) ||
        !(b = NAG_ALLOC(n * 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 LQ factorization of A */
/* nag_dgelqf (f08ahc).
 * LQ factorization of real general rectangular matrix
 */
nag_dgelqf(order, m, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgelqf (f08ahc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Solve L*Y = B, storing the result in B */
/* nag_dtrtrs (f07tec).
 * Solution of real triangular system of linear equations,
 * multiple right-hand sides
 */
nag_dtrtrs(order, Nag_Lower, Nag_NoTrans, Nag_NonUnitDiag, m,
           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;
}
/* Set rows (M+1) to N of B to zero */
if (m < n) {
    for (i = m + 1; i <= n; ++i) {
        for (j = 1; j <= nrhs; ++j)
            B(i, j) = 0.0;
    }
}

/* Compute minimum-norm solution X = (Q^T)*B in B */
/* nag_dormlq (f08akc).
 * Apply orthogonal transformation determined by nag_dgelqf (f08ahc)
 */
nag_dormlq(order, Nag_LeftSide, Nag_Trans, n, nrhs, m, a, pda,
           tau, b, pdb, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dormlq (f08akc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print minimum-norm 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, "Minimum-norm 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_dgelqf (f08ahc) Example Program Data
  4  6  2                               :Values of M, N and NRHS
-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
-2.87 -5.23
  1.63  0.29
-3.52  4.76
  0.45 -8.41                               :End of matrix B
```

**10.3 Program Results**

```
nag_dgelqf (f08ahc) Example Program Results
```

```
Minimum-norm solution(s)
      1      2
  1      0.2371      0.7383
  2     -0.4575      0.0158
  3     -0.0085     -0.0161
  4     -0.5192      1.0768
  5      0.0239     -0.6436
  6     -0.0543     -0.6613
```

---