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

```c
#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

\[
A = (L \ 0)Q \iff A^T = Q^T\begin{pmatrix} L_1^T \\ 0 \end{pmatrix}
\]

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: \textbf{order} – Nag_OrderType \hspace{1cm} Input

On entry: the \textbf{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 \textbf{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \textbf{order} = Nag_RowMajor or Nag_ColMajor.

2: \textbf{m} – Integer \hspace{1cm} Input

On entry: \(m\), the number of rows of the matrix \(A\).

\textit{Constraint:} \(m \geq 0\).

3: \textbf{n} – Integer \hspace{1cm} Input

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

\textit{Constraint:} \(n \geq 0\).

4: \textbf{a} \([\text{dim}]\) – double \hspace{1cm} Input/Output

\textit{Note:} the dimension, \textit{dim}, of the array \textbf{a} must be at least

\[
\max(1, pda \times n) \text{ when } \textbf{order} = \text{Nag}_\text{ColMajor}; \\
\max(1, m \times pda) \text{ when } \textbf{order} = \text{Nag}_\text{RowMajor}.
\]

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

\[
a[(j - 1) \times pda + i - 1] \text{ when } \textbf{order} = \text{Nag}_\text{ColMajor}; \\
a[(i - 1) \times pda + j - 1] \text{ when } \textbf{order} = \text{Nag}_\text{RowMajor}.
\]

On entry: the \(m\) by \(n\) matrix \(A\).

\textit{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: \textbf{pda} – Integer \hspace{1cm} Input

On entry: the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{a}.

\textit{Constraints:}

\[
\begin{align*}
\text{if } \textbf{order} = \text{Nag}_\text{ColMajor}, & & pda \geq \max(1, m); \\
\text{if } \textbf{order} = \text{Nag}_\text{RowMajor}, & & pda \geq \max(1, n).
\end{align*}
\]

6: \textbf{tau} \([\text{dim}]\) – double \hspace{1cm} Output

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

On exit: further details of the orthogonal matrix \(Q\).

7: \textbf{fail} – NagError * \hspace{1cm} 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.
See Section 3.2.1.2 in the Essential Introduction 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: \(m \geq 0\).
On entry, \(n = \langle value\rangle\).
Constraint: \(n \geq 0\).
On entry, \(pda = \langle value\rangle\).
Constraint: \(pda > 0\).

**NE_INT_2**
On entry, \(pda = \langle value\rangle\) and \(m = \langle value\rangle\).
Constraint: \(pda \geq \max(1, m)\).
On entry, \(pda = \langle value\rangle\) and \(n = \langle value\rangle\).
Constraint: \(pda \geq \max(1, n)\).

**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 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction 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) is not threaded by NAG in any implementation.

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.
Further Comments

The total number of floating-point operations is approximately \( \frac{3}{2}m^2(3n - m) \) if \( m \leq n \) or \( \frac{3}{2}n^2(3m - n) \) if \( m > n \).

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

\[
\text{nag_dorglq(order, n, n, MIN(m,n), &a, pda, tau, &fail)}
\]

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:

\[
\text{nag_dorglq(order, m, n, m, &a, pda, tau, &fail)}
\]

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

\[
\text{nag_dormlq(order,Nag_LeftSide,Nag_Trans,m,p,MIN(m,n),&a,pda,}
\]

\[
\text{tau, &c, pdc, &fail)}
\]

forms the matrix product \( C = QTQ \), where \( C \) is \( m \) by \( p \).

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

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 \( B \),

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

and

\[
B = \begin{pmatrix}
-2.87 & -5.23 \\
1.63 & 0.29 \\
-3.52 & 4.76 \\
0.45 & -8.41
\end{pmatrix}
\]

10.1 Program Text

\[
\text{/* nag_dgelqf (f08ahc) Example Program. */}
\]

\[
\text{/* Copyright 2014 Numerical Algorithms Group. */}
\]

\[
\text{/* Mark 7, 2001. */}
\]

#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]
    #endif

    order = Nag_ColMajor;

    // Rest of the program...
}
#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_dgelqf (f08ahc) Example Program Results\n\n");

/* Skip heading in data file */
#endif //WIN32
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif //WIN32

#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 //WIN32

#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
 */
void nag_dgelqf(int order, int m, int n, double a[], int pda, double tau[], NagError *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
 */
void nag_dtrtrs(int order, Nag_Lower, Nag_NoTrans, Nag_NoUnitDiag, int m,
                 int nrhs, double a[], int pda, double b[], int pdb, NagError *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)
 */
void nag_dormlq(int order, Nag_LeftSide, Nag_Trans, int n, int nrhs, int m, double a[],
                 int pda, double tau[], double b[], int pdb, NagError *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)
 */
void nag_gen_real_mat_print(int order, Nag_GeneralMatrix, Nag_NoUnitDiag, int n, int nrhs, double b[],
                            int pdb, NagError *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)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2371</td>
<td>0.7383</td>
</tr>
<tr>
<td>2</td>
<td>-0.4575</td>
<td>0.0158</td>
</tr>
<tr>
<td>3</td>
<td>-0.0085</td>
<td>-0.0161</td>
</tr>
<tr>
<td>4</td>
<td>-0.5192</td>
<td>1.0768</td>
</tr>
<tr>
<td>5</td>
<td>0.0239</td>
<td>-0.6436</td>
</tr>
<tr>
<td>6</td>
<td>-0.0543</td>
<td>-0.6613</td>
</tr>
</tbody>
</table>