NAG Library Function Document

nag_dorglq (f08ajc)

1 Purpose

nag_dorglq (f08ajc) generates all or part of the real orthogonal matrix $Q$ from an $LQ$ factorization computed by nag_dgelqf (f08ahc).

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_dorglq (Nag_OrderType order, Integer m, Integer n, Integer k,
                 double a[], Integer pda, const double tau[], NagError *fail)
```

3 Description

nag_dorglq (f08ajc) is intended to be used after a call to nag_dgelqf (f08ahc), which performs an $LQ$ 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 rows.

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

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

(note that the array $a$ must have at least $n$ rows) or its leading $p$ rows by:

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

The rows of $Q$ returned by the last call form an orthonormal basis for the space spanned by the rows of $A$; thus nag_dgelqf (f08ahc) followed by nag_dorglq (f08ajc) can be used to orthogonalize the rows of $A$.

The information returned by the $LQ$ factorization functions also yields the $LQ$ factorization of the leading $k$ rows of $A$, where $k < p$. The orthogonal matrix arising from this factorization can be computed by:

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

or its leading $k$ rows by:

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

4 References


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 \hspace{1cm} \text{Input}

\textit{On entry:} \( m \), the number of rows of the matrix \( Q \).

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

3: \( n \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} \( n \), the number of columns of the matrix \( Q \).

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

4: \( k \) – Integer \hspace{1cm} \text{Input}

\textit{On entry:} \( k \), the number of elementary reflectors whose product defines the matrix \( Q \).

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

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

\textit{Note:} the dimension, \( dim \), of the array \( a \) must be at least
\[
\max(1,pda \times n) \quad \text{when order = Nag\_ColMajor};
\]
\[
\max(1,m \times pda) \quad \text{when order = Nag\_RowMajor}.
\]

\textit{On entry:} details of the vectors which define the elementary reflectors, as returned by \( \text{nag\_dgelqf} \) (f08ahc).

\textit{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 \hspace{1cm} \text{Input}

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

\textit{Constraints:}

if order = Nag\_ColMajor, \( pda \geq \max(1,m) \);

if order = Nag\_RowMajor, \( pda \geq \max(1,n) \).

7: \( \tau[dim] \) – const double \hspace{1cm} \text{Input}

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

\textit{On entry:} further details of the elementary reflectors, as returned by \( \text{nag\_dgelqf} \) (f08ahc).

8: \( \text{fail} \) – NagError * \hspace{1cm} \text{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 \(<value>\) had an illegal value.
On entry, \(m = \langle \text{value} \rangle\).
Constraint: \(m \geq 0\).

On entry, \(pda = \langle \text{value} \rangle\).
Constraint: \(pda > 0\).

On entry, \(m = \langle \text{value} \rangle\) and \(k = \langle \text{value} \rangle\).
Constraint: \(m \geq k \geq 0\).

On entry, \(n = \langle \text{value} \rangle\) and \(m = \langle \text{value} \rangle\).
Constraint: \(n \geq m\).

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

On entry, \(pda = \langle \text{value} \rangle\) and \(n = \langle \text{value} \rangle\).
Constraint: \(pda \geq \max(1, n)\).

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.

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.

The computed matrix \(Q\) differs from an exactly orthogonal matrix by a matrix \(E\) such that

\[\|E\|_2 = O(\epsilon),\]

where \(\epsilon\) is the machine precision.

The total number of floating-point operations is approximately

\[4mnk - 2(m + n)k^2 + \frac{4}{3}k^3;\]

when \(m = k\), the number is approximately \(\frac{2}{3}m^2(3n - m)\).

The complex analogue of this function is nag_zunglq (f08awc).
10 Example

This example forms the leading 4 rows of the orthogonal matrix $Q$ from the $LQ$ factorization of the matrix $A$, where

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

The rows of $Q$ form an orthonormal basis for the space spanned by the rows of $A$.

10.1 Program Text

/* nag_dorglq (f08ajc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* * Mark 7, 2001.
* * Mark 7b revised, 2004.
* /
#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;
    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_dorglq (f08ajc) 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

/* Allocate memory */
if (!(title = NAG_ALLOC(31, 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){
        if (_WIN32)
            scanf_s("%lf", &A(i, j));
        else
            scanf("%lf", &A(i, j));
    }
    if (_WIN32)
        scanf_s("%*[\n] ");
    else
        scanf("%*[\n] ");
}

/* Compute the LQ factorization of A */
/* nag_dgelqf (f08ahc).
* LQ factorization of real general rectangular matrix */
if (fail.code != NE_NOERROR){
    printf("Error from nag_dgelqf (f08ahc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Form the leading M rows of Q explicitly */
/* nag_dorglq (f08ajc).
* Form all or part of orthogonal Q from LQ factorization
* determined by nag_dgelqf (f08ahc) */
if (fail.code != NE_NOERROR){
    printf("Error from nag_dorglq (f08ajc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Print the leading M rows of Q only */
/* nag_gen_real_mat_print (x04cac).
* Print real general matrix (easy-to-use) */
fflush(stdout);
if (fail.code != NE_NOERROR){
    printf("Error from nag_gen_real_mat_print (x04cac).
%s
", fail.message);
}
exit_status = 1;
break;
}

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

return exit_status;
}

10.2 Program Data

nag_dorglq (f08ajc) Example Program Data

4 6
-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

10.3 Program Results

nag_dorglq (f08ajc) Example Program Results

The leading 4 rows of Q

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.7104</td>
<td>0.4299</td>
<td>-0.4824</td>
<td>0.0354</td>
<td>0.2700</td>
<td>0.0603</td>
</tr>
<tr>
<td>2</td>
<td>-0.2412</td>
<td>-0.5323</td>
<td>-0.4845</td>
<td>-0.1595</td>
<td>-0.6311</td>
<td>-0.0027</td>
</tr>
<tr>
<td>3</td>
<td>0.1287</td>
<td>-0.2619</td>
<td>-0.2108</td>
<td>-0.7447</td>
<td>0.5227</td>
<td>-0.2063</td>
</tr>
<tr>
<td>4</td>
<td>-0.3403</td>
<td>-0.0921</td>
<td>0.4546</td>
<td>-0.3869</td>
<td>-0.0465</td>
<td>0.7191</td>
</tr>
</tbody>
</table>