NAG Library Function Document

nag_dgeqrt (f08abc)

1 Purpose

nag_dgeqrt (f08abc) recursively computes, with explicit blocking, the QR factorization of a real \( m \) by \( n \) matrix.

2 Specification

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

void nag_dgeqrt (Nag_OrderType order, Integer m, Integer n, Integer nb,
                 double a[], Integer pda, double t[], Integer pdt, NagError *fail)
```

3 Description

nag_dgeqrt (f08abc) forms the QR factorization of an arbitrary rectangular real \( m \) by \( n \) matrix. No pivoting is performed.

It differs from nag_dgeqrf (f08aec) in that it: requires an explicit block size; stores reflector factors that are upper triangular matrices of the chosen block size (rather than scalars); and recursively computes the QR factorization based on the algorithm of Elmroth and Gustavson (2000).

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 upper 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 represents a \( QR \) factorization of the first \( k \) columns of the original matrix \( A \).
4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType 
   \( \text{Input} \)
   
   On entry: the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( m \) – Integer 
   \( \text{Input} \)
   
   On entry: \( m \), the number of rows of the matrix \( A \).

   Constraint: \( m \geq 0 \).

3: \( n \) – Integer 
   \( \text{Input} \)
   
   On entry: \( n \), the number of columns of the matrix \( A \).

   Constraint: \( n \geq 0 \).

4: \( nb \) – Integer 
   \( \text{Input} \)
   
   On entry: the explicitly chosen block size to be used in computing the \( QR \) factorization. See Section 9 for details.

   Constraints:
   
   \( nb \geq 1 \);
   if \( \min(m, n) > 0 \), \( nb \leq \min(m, n) \).

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

   Note: the dimension, \( dim \), of the array \( a \) must be at least

   \[ \max(1, \text{pda} \times n) \] when \( \text{order} = \text{Nag_ColMajor} \);

   \[ \max(1, m \times \text{pda}) \] when \( \text{order} = \text{Nag_RowMajor} \).

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

   \[ \text{a}[(j - 1) \times \text{pda} + i - 1] \] when \( \text{order} = \text{Nag_ColMajor} \);

   \[ \text{a}[(i - 1) \times \text{pda} + j - 1] \] when \( \text{order} = \text{Nag_RowMajor} \).

   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 \( R \).

6: \( \text{pda} \) – Integer 
   \( \text{Input} \)

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

if $\text{order} = \text{Nag\_ColMajor}$, $pda \geq \max(1, m)$;
if $\text{order} = \text{Nag\_RowMajor}$, $pda \geq \max(1, n)$.

7: $t[dim]$ – double

**Output**

Note: the dimension, $dim$, of the array $t$ must be at least

$max(1, pdt \times \min(m, n))$ when $\text{order} = \text{Nag\_ColMajor};$
$max(1, nb \times pdt)$ when $\text{order} = \text{Nag\_RowMajor}$.

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

$t[(j - 1) \times pdt + i - 1]$ when $\text{order} = \text{Nag\_ColMajor};$
$t[(i - 1) \times pdt + j - 1]$ when $\text{order} = \text{Nag\_RowMajor}$.

On exit: further details of the orthogonal matrix $Q$. The number of blocks is $b = \left\lceil \frac{k}{nb} \right\rceil$, where $k = \min(m, n)$ and each block is of order $nb$ except for the last block, which is of order $k - (b - 1) \times nb$. For each of the blocks, an upper triangular block reflector factor is computed: $T_1, T_2, \ldots, T_b$. These are stored in the $nb$ by $n$ matrix $T$ as $T = [T_1|T_2|\ldots|T_b]$.

8: $pdt$ – Integer

**Input**

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

Constraints:

if $\text{order} = \text{Nag\_ColMajor}$, $pdt \geq nb;$
if $\text{order} = \text{Nag\_RowMajor}$, $pdt \geq \max(1, \min(m, n))$.

9: $fail$ – NagError *

**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: $n \geq 0$.

**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: $pda \geq \max(1, n)$.

On entry, $pdt = \langle value \rangle$ and $nb = \langle value \rangle$.
Constraint: $pdt \geq nb$.
NE_INT_3

On entry, \( nb = \langle \text{value} \rangle \), \( m = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( nb \geq 1 \) and
if \( \min(m, n) > 0 \), \( nb \leq \min(m, n) \).

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

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_dgeqrt (f08abc) is not threaded by NAG in any implementation.

nag_dgeqrt (f08abc) 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 apply \( Q \) to an arbitrary real rectangular matrix \( C \), nag_dgeqrt (f08abc) may be followed by a call to nag_dgemqrt (f08acc). For example,

\[
\text{nag_dgemqrt(order,Nag_LeftSide,Nag_Trans,m,p,MIN(m,n),nb,a,pda,t,pdt,c,pdc,&fail)}
\]

forms \( C = QT \), where \( C \) is \( m \) by \( p \).

To form the orthogonal matrix \( Q \) explicitly, simply initialize the \( m \) by \( m \) matrix \( C \) to the identity matrix and form \( C = QC \) using nag_dgemqrt (f08acc) as above.

The block size, \( nb \), used by nag_dgeqrt (f08abc) is supplied explicitly through the interface. For moderate and large sizes of matrix, the block size can have a marked effect on the efficiency of the algorithm with the optimal value being dependent on problem size and platform. A value of \( nb = 64 \ll \min(m, n) \) is likely to achieve good efficiency and it is unlikely that an optimal value would exceed 340.

To compute a \( QR \) factorization with column pivoting, use nag_dtpqrt (f08bbc) or nag_dgeqpf (f08bec).
The complex analogue of this function is nag_zgeqrt (f08apc).

10 Example

This example solves the linear least squares problems

\[
\minimize \|Ax_i - b_i\|_2, \quad i = 1, 2
\]

where \(b_1\) and \(b_2\) are the columns of the matrix \(B\),

\[
A = \begin{pmatrix}
-0.57 & -1.28 & -0.39 & 0.25 \\
-1.93 & 1.08 & -0.31 & -2.14 \\
2.30 & 0.24 & 0.40 & -0.35 \\
-1.93 & 0.64 & -0.66 & 0.08 \\
0.15 & 0.30 & 0.15 & -2.13 \\
-0.02 & 1.03 & -1.43 & 0.50
\end{pmatrix}
\quad \text{and} \quad
B = \begin{pmatrix}
-2.67 & 0.41 \\
-0.55 & -3.10 \\
3.34 & -4.01 \\
-0.77 & 2.76 \\
0.48 & -6.17 \\
4.10 & 0.21
\end{pmatrix}
\]

10.1 Program Text

/* nag_dgeqrt (f08abc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 24, 2013.
*/

#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double rnorm;
    Integer exit_status = 0;
    Integer pda, pdb, pdt;
    Integer i, j, m, n, nb, nrhs;
    /* Arrays */
    double *a = 0, *b = 0, *t = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda + I-1]
    #define B(I,J) b[(J-1)*pdb + I-1]
    #define T(I,J) t[(J-1)*pdt + I-1]
    order = Nag_ColMajor;
    #else
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dgeqrt (f08abc) Example Program Results\n\n");
    fflush(stdout);

    /* Skip heading in data file*/
    #ifdef _WIN32
    scanf_s("%*[\n"];
    #else
    scanf("%*[\n"];
    
Mark 25 | f08abc.5
```c
#include "NAG_INT"  
#include "NAG_INT"  
#include "NAG_INT"  
#include "NAG_INT"

#define MIN(m, n)

#define NAG_ALLOC(m, n, double)

#define NAG_COLUMN_MAJOR

#define _WIN32

int main()
{
    int m, n, nrhs;
    double A[m][n];
    double B[m][nrhs];

    if ( !(a = NAG_ALLOC(m*n, double))
        || !(b = NAG_ALLOC(m*nrhs, double))
        || !(t = NAG_ALLOC(nb*MIN(m, n), double)) )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    if (#ifdef _WIN32
        scanf_s("%lf", &A(i, j));
    #else
        scanf("%lf", &A(i, j));
    #endif
    #ifdef _WIN32
        scanf_s("%lf", &B(i, j));
    #else
        scanf("%lf", &B(i, j));
    #endif
    /* nag_dgeqrt (f08abc).
    * Compute the QR factorization of A by recursive algorithm.
    */
    nag_dgeqrt(order, m, n, nb, a, pda, t, pdt, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dgeqrt (f08abc).\n", fail.message);
        exit_status = 1;
        goto END;
    }
    /* nag_dgemqrt (f08acc).
    * Compute C = (C1) = (Q^T)*B, storing the result in B
    * (C2)
    * by applying Q^T from left.
    */
    nag_dgemqrt(order, Nag_LeftSide, Nag_Trans, m, nrhs, n, nb, a, pda, t, pdt, b, pdb, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dgemqrt (f08acc).\n", fail.message);
    }
    return 0;
}
```
exit_status = 2;
goto END;
}

/* nag_dtrtrs (f07tec).
 * Compute least-squares solutions by backsubstitution in R*X = C1.
 */

nag_dtrtrs(order, Nag_Upper, Nag_NoTrans, Nag_NoUnitDiag, 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 = 3;
goto END;
}

/* nag_gen_real_mat_print (x04cac).
 * Print least-squares solutions.
 */
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NoUnitDiag, 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 = 4;
goto END;
}
printf(" Square root(s) of the residual sum(s) of squares\n");
for ( j=1; j<=nrhs; j++) {
/* nag_dge_norm (f16rac).
 * Compute and print estimate of the square root of the residual
 * sum of squares.
 */
nag_dge_norm(order, Nag_FrobeniusNorm, m - n, 1, &B(n + 1,j), pdb, &rnorm,
&fail);
if (fail.code != NE_NOERROR) {
printf("Error from nag_dge_norm (f16rac).\n%s\n", fail.message);
exit_status = 5;
goto END;
}
printf(" %11.2e ", rnorm);
}
printf("\n");

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(t);
return exit_status;
}

10.2 Program Data

nag_dgeqrt (f08abc) Example Program Data

f08 – Least-squares and Eigenvalue Problems (LAPACK)
### 10.3 Program Results

**nag_dgeqrt (f08abc) Example Program Results**

<table>
<thead>
<tr>
<th>Least-squares solution(s)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5339</td>
<td>-1.5753</td>
</tr>
<tr>
<td>2</td>
<td>1.8707</td>
<td>0.5559</td>
</tr>
<tr>
<td>3</td>
<td>-1.5241</td>
<td>1.3119</td>
</tr>
<tr>
<td>4</td>
<td>0.0392</td>
<td>2.9585</td>
</tr>
</tbody>
</table>

**Square root(s) of the residual sum(s) of squares**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.22e-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.38e-02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>