NAG Toolbox

nag_lapack_dgemqrt (f08ac)

1 Purpose

nag_lapack_dgemqrt (f08ac) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from a QR factorization computed by nag_lapack_dgeqrt (f08ab).

2 Syntax

```
[c, info] = nag_lapack_dgemqrt(side, trans, v, t, c, 'm', m, 'n', n, 'k', k, 'nb', nb)
[c, info] = f08ac(side, trans, v, t, c, 'm', m, 'n', n, 'k', k, 'nb', nb)
```

3 Description

nag_lapack_dgemqrt (f08ac) is intended to be used after a call to nag_lapack_dgeqrt (f08ab) which performs a QR factorization of a real matrix A. The orthogonal matrix Q is represented as a product of elementary reflectors.

This function may be used to form one of the matrix products

$$QC, Q^{\mathrm{T}}C, CQ \text{ or } CQ^{\mathrm{T}},$$

overwriting the result on C (which may be any real rectangular matrix).

A common application of this function is in solving linear least squares problems, as described in the F08 Chapter Introduction and illustrated in Section 10 in nag_lapack_dgeqrt (f08ab).

4 References

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

5 Parameters

5.1 Compulsory Input Parameters

1: **side** – CHARACTER(1)

Indicates how Q or Q^{T} is to be applied to C.

side = 'L'
Q or Q^{T} is applied to C from the left.

side = 'R'
Q or Q^{T} is applied to C from the right.

Constraint: **side** = 'L' or 'R'.

2: **trans** – CHARACTER(1)

Indicates whether Q or Q^{T} is to be applied to C.

trans = 'N'
Q is applied to C.

trans = 'T'
Q^{T} is applied to C.

Constraint: **trans** = 'N' or 'T'.

3: **v**$(ldv,:)$ – REAL (KIND=nag_wp) array

The first dimension, ldv, of the array **v** must satisfy

if **side** = 'L', $ldv \geq \max(1,\mathbf{m})$;
if **side** = 'R', $ldv \geq \max(1,\mathbf{n})$.

The second dimension of the array **v** must be at least $\max(1,\mathbf{k})$.

Details of the vectors which define the elementary reflectors, as returned by nag_lapack_dgeqrt (f08ab) in the first k columns of its array argument **a**.

4: **t**$(ldt,:)$ – REAL (KIND=nag_wp) array

The first dimension of the array **t** must be at least **nb**.

The second dimension of the array **t** must be at least $\max(1,\mathbf{k})$.

Further details of the orthogonal matrix Q as returned by nag_lapack_dgeqrt (f08ab). The number of blocks is $b = \left\lceil \frac{k}{\mathbf{nb}} \right\rceil$, where $k = \min(m,n)$ and each block is of order **nb** except for the last block, which is of order $k - (b-1) \times \mathbf{nb}$. For the b blocks the upper triangular block reflector factors $\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_b$ are stored in the **nb** by n matrix T as $\boldsymbol{T} = [\boldsymbol{T}_1 | \boldsymbol{T}_2 | \ldots | \boldsymbol{T}_b]$.

5: **c**$(ldc,:)$ – REAL (KIND=nag_wp) array

The first dimension of the array **c** must be at least $\max(1,\mathbf{m})$.

The second dimension of the array **c** must be at least $\max(1,\mathbf{n})$.

The m by n matrix C.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the first dimension of the array **c**.

m, the number of rows of the matrix C.

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

2: **n** – INTEGER

Default: the second dimension of the array **c**.

n, the number of columns of the matrix C.

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

3: **k** – INTEGER

Default: the second dimension of the arrays **v**, **t**.

k, the number of elementary reflectors whose product defines the matrix Q. Usually $\mathbf{k} = \min(m_A, n_A)$ where m_A, n_A are the dimensions of the matrix A supplied in a previous call to nag_lapack_dgeqrt (f08ab).

Constraints:

if **side** = 'L', $\mathbf{m} \geq \mathbf{k} \geq 0$;
if **side** = 'R', $\mathbf{n} \geq \mathbf{k} \geq 0$.

4: **nb** – INTEGER

Default: the first dimension of the array **t**.

The block size used in the QR factorization performed in a previous call to nag_lapack_dgeqrt (f08ab); this value must remain unchanged from that call.

Constraints:

nb ≥ 1;
if **k** > 0, **nb** $\leq$ **k**.

5.3 Output Parameters

1: **c**$(ldc,:)$ – REAL (KIND=nag_wp) array

The first dimension of the array **c** will be $\max(1,\mathbf{m})$.

The second dimension of the array **c** will be $\max(1,\mathbf{n})$.

c stores QC or $Q^{\mathrm{T}}C$ or CQ or CQ^{T} as specified by **side** and **trans**.

2: **info** – INTEGER

info $= 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

info < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

7 Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ϵ is the ***machine precision***.

8 Further Comments

The total number of floating-point operations is approximately $2nk(2m-k)$ if **side** = 'L' and $2mk(2n-k)$ if **side** = 'R'.

The complex analogue of this function is nag_lapack_zgemqrt (f08aq).

9 Example

See Section 10 in nag_lapack_dgeqrt (f08ab).

9.1 Program Text

```
function f08ac_example

fprintf('f08ac example results\n\n');

% Minimize ||Ax - b|| using recursive QR for m-by-n A and m-by-p B

m = nag_int(6);
n = nag_int(4);
p = nag_int(2);
a = [-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];
b = [-2.67,  0.41;
     -0.55, -3.10;
      3.34, -4.01;
     -0.77,  2.76;
      0.48, -6.17;
      4.10,  0.21];

% Compute the QR Factorisation of A
[QR, T, info] = f08ab(n,a);

% Compute C = (C1) = (Q^T)*B
[c1, info] = f08ac(...
                  'Left', 'Transpose', QR, T, b);

% Compute least-squares solutions by backsubstitution in R*X = C1
[x, info] = f07te(...
                  'Upper', 'No Transpose', 'Non-Unit', QR, c1, 'n', n);

% Print least-squares solutions
disp('Least-squares solutions');
disp(x(1:n,:));

% Compute and print estimates of the square roots of the residual
% sums of squares
for j=1:p
  rnorm(j) = norm(x(n+1:m,j));
end
fprintf('\nSquare roots of the residual sums of squares\n');
fprintf('%12.2e', rnorm);
fprintf('\n');
```

9.2 Program Results

```
     f08ac example results

Least-squares solutions
    1.5339   -1.5753
    1.8707    0.5559
   -1.5241    1.3119
    0.0392    2.9585

Square roots of the residual sums of squares
    2.22e-02    1.38e-02
```