NAG Toolbox

nag_lapack_ztpmqrt (f08bq)

1 Purpose

nag_lapack_ztpmqrt (f08bq) multiplies an arbitrary complex matrix C by the complex unitary matrix Q from a QR factorization computed by nag_lapack_ztpqrt (f08bp).

2 Syntax

```
[c1, c2, info] = nag_lapack_ztpmqrt(side, trans, l, v, t, c1, c2, 'm', m, 'n', n, 'k', k, 'nb', nb)
[c1, c2, info] = f08bq(side, trans, l, v, t, c1, c2, 'm', m, 'n', n, 'k', k, 'nb', nb)
```

3 Description

nag_lapack_ztpmqrt (f08bq) is intended to be used after a call to nag_lapack_ztpqrt (f08bp) which performs a QR factorization of a triangular-pentagonal matrix containing an upper triangular matrix A over a pentagonal matrix B. The unitary matrix Q is represented as a product of elementary reflectors.

This function may be used to form the matrix products

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

where the complex rectangular m_c by n_c matrix C is split into component matrices C_1 and C_2.

If Q is being applied from the left (QC or $Q^{\mathrm{H}}C$) then

$$C = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}$$

where C_1 is k by n_c, C_2 is m_v by n_c, $m_c = k + m_v$ is fixed and m_v is the number of rows of the matrix V containing the elementary reflectors (i.e., **m** as passed to nag_lapack_ztpqrt (f08bp)); the number of columns of V is n_v (i.e., **n** as passed to nag_lapack_ztpqrt (f08bp)).

If Q is being applied from the right (CQ or CQ^{H}) then

$$C = \begin{pmatrix} C_1 & C_2 \end{pmatrix}$$

where C_1 is m_c by k, and C_2 is m_c by m_v and $n_c = k + m_v$ is fixed.

The matrices C_1 and C_2 are overwriten by the result of the matrix product.

A common application of this routine is in updating the solution of a linear least squares problem as illustrated in Section 10 in nag_lapack_ztpqrt (f08bp).

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^{H} is to be applied to C.

side = 'L'
Q or Q^{H} is applied to C from the left.

side = 'R'
Q or Q^{H} is applied to C from the right.

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

2: **trans** – CHARACTER(1)

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

trans = 'N'
Q is applied to C.

trans = 'C'
Q^{H} is applied to C.

Constraint: **trans** = 'N' or 'C'.

3: **l** – INTEGER

l, the number of rows of the upper trapezoidal part of the pentagonal composite matrix V, passed (as **b**) in a previous call to nag_lapack_ztpqrt (f08bp). This must be the same value used in the previous call to nag_lapack_ztpqrt (f08bp) (see **l** in nag_lapack_ztpqrt (f08bp)).

Constraint: $0 \leq \mathbf{l} \leq \mathbf{k}$.

4: **v**$(ldv, :)$ – COMPLEX (KIND=nag_wp) array

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

The m_v by n_v matrix V; this should remain unchanged from the array **b** returned by a previous call to nag_lapack_ztpqrt (f08bp).

5: **t**$(ldt, :)$ – COMPLEX (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 $\max(1, \mathbf{k})$.

This must remain unchanged from a previous call to nag_lapack_ztpqrt (f08bp) (see **t** in nag_lapack_ztpqrt (f08bp)).

6: **c1**$(ldc1, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension, *ldc1*, of the array **c1** must satisfy

if **side** = 'L', $ldc1 \geq \max(1, \mathbf{k})$;
if **side** = 'R', $ldc1 \geq \max(1, \mathbf{m})$.

The second dimension of the array **c1** must be at least $\max(1, \mathbf{n})$ if **side** = 'L' and at least $\max(1, \mathbf{k})$ if **side** = 'R'.

C_1, the first part of the composite matrix C:

if **side** = 'L'
then **c1** contains the first k rows of C;

if **side** = 'R'
then **c1** contains the first k columns of C.

7: **c2**$(ldc2,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **c2** must be at least $\max(1,\mathbf{m})$.

The second dimension of the array **c2** must be at least $\max(1,\mathbf{n})$.

C_2, the second part of the composite matrix C.

if **side** = 'L'
then **c2** contains the remaining m_v rows of C;

if **side** = 'R'
then **c2** contains the remaining m_v columns of C;

5.2 Optional Input Parameters

1: **m** – INTEGER

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

The number of rows of the matrix C_2, that is,

if **side** = 'L'
then m_v, the number of rows of the matrix V;

if **side** = 'R'
then m_c, the number of rows of the matrix C.

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

2: **n** – INTEGER

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

The number of columns of the matrix C_2, that is,

if **side** = 'L'
then n_c, the number of columns of the matrix C;

if **side** = 'R'
then n_v, the number of columns of the matrix V.

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

3: **k** – INTEGER

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

k, the number of elementary reflectors whose product defines the matrix Q.

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

4: **nb** – INTEGER

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

nb, the blocking factor used in a previous call to nag_lapack_ztpqrt (f08bp) to compute the QR factorization of a triangular-pentagonal matrix containing composite matrices A and B.

Constraints:

$\mathbf{nb} \geq 1$;
if $\mathbf{k} > 0$, $\mathbf{nb} \leq \mathbf{k}$.

5.3 Output Parameters

1: **c1**$(ldc1,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, $ldc1$, of the array **c1** will be

if **side** = 'L', $ldc1 = \max(1, \mathbf{k})$;
if **side** = 'R', $ldc1 = \max(1, \mathbf{m})$.

The second dimension of the array **c1** will be $\max(1, \mathbf{n})$ if **side** = 'L' and at least $\max(1, \mathbf{k})$ if **side** = 'R'.

c1 stores the corresponding block of QC or $Q^{\mathrm{H}}C$ or CQ or CQ^{H}.

2: **c2**$(ldc2,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **c2** will be $\max(1, \mathbf{m})$.

The second dimension of the array **c2** will be $\max(1, \mathbf{n})$.

c2 stores the corresponding block of QC or $Q^{\mathrm{H}}C$ or CQ or CQ^{H}.

3: **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 real analogue of this function is nag_lapack_dtpmqrt (f08bc).

9 Example

See Section 10 in nag_lapack_ztpqrt (f08bp).

9.1 Program Text

```
function f08bq_example

fprintf('f08bq 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 = [ 0.96 - 0.81i,  -0.03 + 0.96i,  -0.91 + 2.06i,  -0.05 + 0.41i;
```

```
     -0.98 + 1.98i,  -1.20 + 0.19i,  -0.66 + 0.42i,  -0.81 + 0.56i;
      0.62 - 0.46i,   1.01 + 0.02i,   0.63 - 0.17i,  -1.11 + 0.60i;
     -0.37 + 0.38i,   0.19 - 0.54i,  -0.98 - 0.36i,   0.22 - 0.20i;
      0.83 + 0.51i,   0.20 + 0.01i,  -0.17 - 0.46i,   1.47 + 1.59i;
      1.08 - 0.28i,   0.20 - 0.12i,  -0.07 + 1.23i,   0.26 + 0.26i];
b = [-2.09 + 1.93i,   3.26-2.70i;
      3.34 - 3.53i,  -6.22+1.16i;
     -4.94 - 2.04i,   7.94-3.13i;
      0.17 + 4.23i,   1.04-4.26i;
     -5.19 + 3.63i,  -2.31-2.12i;
      0.98 + 2.53i,  -1.39-4.05i];

nb = n;
% Compute the QR Factorisation of first n rows of A
[QRn, Tn, info] = f08ap( ...
  nb,a(1:n,:));

% Compute C = (C1) = (Q^H)*B
[c1, info] = f08aq( ...
     'Left', 'Conjugate Transpose', QRn, Tn, b(1:n,:));

% Compute least-squares solutions by backsubstitution in R*X = C1
[x, info] = f07ts( ...
    'Upper', 'No Transpose', 'Non-Unit', QRn, c1);

% Print first n-row solutions
disp('Solution for n rows');
disp(x(1:n,:));

% Add the remaining rows and perform QR update
nb2 = m-n;
l = nag_int(0);
[R, Q, T, info] = f08bp( ...
  l, nb2, QRn, a(n+1:m,:));

% Apply orthogonal transformations to C
[c1,c2,info] = f08bq( ...
       'Left','Conjugate Transpose', l, Q, T, c1, b(n+1:m,:));

% Compute least-squares solutions for first n rows: R*X = C1
[x, info] = f07ts( ...
    'Upper', 'No transpose', 'Non-Unit', R, c1);
% Print least-squares solutions for all m rows
disp('Least squares solution');
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(c2(:,j));
end
fprintf('Square roots of the residual sums of squares\n');
fprintf('%12.2e', rnorm);
fprintf('\n');
```

9.2 Program Results

```
     f08bq example results

Solution for n rows
  -0.5091 - 1.2428i   0.7569 + 1.4384i
  -2.3789 + 2.8651i   5.1727 - 3.6193i
   1.4634 - 2.2064i  -2.6613 + 2.1339i
   0.4701 + 2.6964i  -2.6933 + 0.2724i

Least squares solution
  -0.5044 - 1.2179i   0.7629 + 1.4529i
  -2.4281 + 2.8574i   5.1570 - 3.6089i
```

```
   1.4872 - 2.1955i  -2.6518 + 2.1203i
   0.4537 + 2.6904i  -2.7606 + 0.3318i

Square roots of the residual sums of squares
    6.88e-02    1.87e-01
```