NAG Toolbox

nag_lapack_zunmrq (f08cx)

1 Purpose

nag_lapack_zunmrq (f08cx) multiplies a general complex m by n matrix C by the complex unitary matrix Q from an RQ factorization computed by nag_lapack_zgerqf (f08cv).

2 Syntax

```
[a, c, info] = nag_lapack_zunmrq(side, trans, a, tau, c, 'm', m, 'n', n, 'k', k)
[a, c, info] = f08cx(side, trans, a, tau, c, 'm', m, 'n', n, 'k', k)
```

3 Description

nag_lapack_zunmrq (f08cx) is intended to be used following a call to nag_lapack_zgerqf (f08cv), which performs an RQ factorization of a complex matrix A and represents the unitary matrix Q as a product of elementary reflectors.

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

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

overwriting the result on C, which may be any complex rectangular m by n matrix.

A common application of this function is in solving underdetermined linear least squares problems, as described in the F08 Chapter Introduction, and illustrated in Section 10 in nag_lapack_zgerqf (f08cv).

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd 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: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

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

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

The ith row of **a** must contain the vector which defines the elementary reflector H_i, for $i = 1,2,\ldots,k$, as returned by nag_lapack_zgerqf (f08cv).

4: **tau**$(:)$ – COMPLEX (KIND=nag_wp) array

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

tau(i) must contain the scalar factor of the elementary reflector H_i, as returned by nag_lapack_zgerqf (f08cv).

5: **c**$(ldc,:)$ – COMPLEX (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 first dimension of the array **a** and the dimension of the array **tau**.

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

Constraints:

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

5.3 Output Parameters

1: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

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

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

Is modified by nag_lapack_zunmrq (f08cx) but restored on exit.

2: **c**$(ldc,:)$ – COMPLEX (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 $Q^{\mathrm{H}}C$ or CQ or CQ^{H} as specified by **side** and **trans**.

3: **info** – INTEGER

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

6 Error Indicators and Warnings

info $= -i$

If **info** $= -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **side**, 2: **trans**, 3: **m**, 4: **n**, 5: **k**, 6: **a**, 7: **lda**, 8: **tau**, 9: **c**, 10: **ldc**, 11: **work**, 12: **lwork**, 13: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

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 $8nk(2m-k)$ if **side** = 'L' and $8mk(2n-k)$ if **side** = 'R'.

The real analogue of this function is nag_lapack_dormrq (f08ck).

9 Example

See Section 10 in nag_lapack_zgerqf (f08cv).

9.1 Program Text

```
function f08cx_example

fprintf('f08cx example results\n\n');

% Find x that minimizes norm(c-Ax) subject to Bx = d .
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];
[m,n] = size(a);
p = 2;

b = complex([ 1 0 -1  0;
              0 1  0 -1]);
c = [-2.54 + 0.09i;
      1.65 - 2.26i;
     -2.11 - 3.96i;
```

```
      1.82 + 3.30i;
     -6.41 + 3.77i;
      2.07 + 0.66i];
d = complex([0;0]);

% Compute the generalized RQ factorization of (B,A) as
% A = ZRQ, B = TQ
[TQ, taub, ZR, taua, info] = ...
  f08zt(b, a);

% Set Qx = y. The problem reduces to:
% minimize (Ry - Z^Hc) subject to Ty = d

% Update c = Z^H*c -> minimize (Ry-c)
[cup, info] = f08au( ...
      'Left','Conjugate Transpose',ZR,taua,c);

% Solve Ty = d for last p elements
T12 = complex(triu(TQ(1:p,n-p+1:n)));

[y2, info] = f07ts( ...
     'Upper', 'No transpose', 'Non-unit', T12, d);

% (from Ry-c) R11*y1 + R12*y2 = c1 --> R11*y1 = c1 - R12*y2
% Update c1
c1 = cup(1:n-p) - ZR(1:n-p,n-p+1:n)*y2;

% Solve R11*y1 = c1 for y1
  R11 = complex(triu(ZR(1:n-p,1:n-p)));
[y1, info] = f07ts( ...
     'Upper', 'No transpose', 'Non-unit', R11, c1);

% Contruct y and backtransform for x = Q^Hy
y = [y1;y2];
[~, x, info] = f08cx( ...
       'Left', 'Conjugate Transpose', TQ, taub, y);
fprintf('Constrained least squares solution\n');
disp(x);

fprintf('Residuals computed directly\n');
res = a*x - c;
disp(res);
fprintf('Residual norm\n');
disp(norm(res));
```

9.2 Program Results

```
     f08cx example results

Constrained least squares solution
   1.0874 - 1.9621i
  -0.7409 + 3.7297i
   1.0874 - 1.9621i
  -0.7409 + 3.7297i

Residuals computed directly
  -0.0035 - 0.1423i
  -0.0325 + 0.0351i
  -0.0052 - 0.0100i
   0.0121 - 0.0039i
   0.0209 + 0.0326i
   0.0293 + 0.0033i

Residual norm
    0.1587
```