NAG Toolbox for MATLAB

f08cx

1 Purpose

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

2 Syntax

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

3 Description

f08cx is intended to be used following a call to 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 9 of the document for 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 URL: 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 – string**

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 – string**

If **trans** = 'N', no transpose, apply Q.

If **trans** = 'C', transpose, apply Q^{H}.

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

3: **a**(**lda**,∗) – **complex array**

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

The second dimension of the array 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 f08cv.

4: **tau**(∗) – **complex array**

Note: 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 f08cv.

5: **c**(**ldc**,∗) – **complex array**

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

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

The m by n matrix C.

5.2 Optional Input Parameters

1: **m** – **int32 scalar**

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

m, the number of rows of the matrix C.

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

2: **n** – **int32 scalar**

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

n, the number of columns of the matrix C.

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

3: **k** – **int32 scalar**

Default: 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 Input Parameters Omitted from the MATLAB Interface

lda, ldc, work, lwork

5.4 Output Parameters

1: **a**(**lda**,∗) – **complex array**

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

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

Is modified by f08cx but restored on exit.

2: **c(ldc,∗) – complex array**

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

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

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

3: **info – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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 f08ck.

9 Example

```
side = 'Left';
trans = 'Conjugate transpose';
a    =   [complex(-0.4690956832366332,   +0.3700184209622965),   complex(-
1.435639356874927, ...
       +0),  complex(-0.5819159055165758,  +1.404651327772382),  complex(-
0.001153842927256641, -0.7616868332407685);
                     complex(0.2378076938977844,    +0.7549978409895591),
complex(0.1432135026706612, ...
               -0.4602166315584957),    complex(2.087706110000044,    +0),
complex(0.2326749346354928, +0.8027736748678639);
                    complex(-0.2371764099566727,    +0.1304426521734852),
complex(0.2798420970655579, ...
                -0.5051559483085669),    complex(0.3816069639813201,    -
0.04079263938687998), complex(1.993338907461549, +0)];
tau = [complex(1.381955486385161, +0.356416692505387);
     complex(1.023239359107476, +0.2321622731044034);
     complex(1.040133667035014, +0.5066875463170494)];
c = [complex(0, +0);
     complex(0.8662588905682393, +8.268193770083355);
     complex(6.267416433349819, -0.6323977917946874);
     complex(-3.797648243188182, +3.476578906908071)];
```

```
[aOut, cOut, info] = f08cx(side, trans, a, tau, c)
```

```
aOut =
   -0.4691 + 0.3700i   -1.4356               -0.5819 + 1.4047i   -0.0012 -
0.7617i
    0.2378 + 0.7550i    0.1432 - 0.4602i    2.0877                0.2327 +
0.8028i
  -0.2372 + 0.1304i   0.2798 - 0.5052i   0.3816 - 0.0408i   1.9933
cOut =
  -2.8501 + 6.4683i
   1.6264 - 0.7799i
   6.9290 + 4.6481i
   1.4048 + 3.2400i
info =
           0
```