NAG Toolbox for MATLAB

f08ag

1 Purpose

f08ag multiplies an arbitrary real matrix C by the real orthogonal matrix Q from a QR factorization computed by f08ae or f08be.

2 Syntax

```
[c, info] = f08ag(side, trans, a, tau, c, 'm', m, 'n', n, 'k', k)
```

3 Description

f08ag is intended to be used after a call to f08ae or f08be, which perform 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 9 of the document for f08ae.

4 References

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^{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 – string**

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: **a**(**lda**,∗) – **double array**

The first dimension, **lda**, of the array **a** must satisfy

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

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

Details of the vectors which define the elementary reflectors, as returned by f08ae or f08be.

4: **tau**(∗) – **double array**

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

Further details of the elementary reflectors as returned by f08ae or f08be.

5: **c**(**ldc**,∗) – **double 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 second dimension of the array **a** 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: **c**(**ldc**,∗) – **double 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 $Q^{\mathrm{T}}C$ or CQ or CQ^{T} as specified by **side** and **trans**.

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

The complex analogue of this function is f08au.

9 Example

```
side = 'Left';
trans = 'Transpose';
a    =    [3.61767881382524,    -0.5565999923223895,    0.847366545721238,
0.7460032078266114;
           0.4608758421558694,  -2.028077032202356,  0.5513872350020937,
1.16996276895585;
         -0.5492302782168392,  -0.04571098289280237,  1.374460641222295,  -
1.410473781059997;
         0.4608758421558694,  0.2828431690617352,  0.004430814804361739,  -
2.375527319588618;
       -0.03581936597066342,  0.0796426824688576,  -0.07728561757441148,  -
0.5213744847432364;
          0.004775915462755124,  0.3002942085609617,  0.801665355572228,
0.2558113872182322];
tau = [1.157559592582321;
     1.696915139470381;
     1.213106371299621;
     1.495583371241627];
c = [-2.67, 0.41;
     -0.55, -3.1;
     3.34, -4.01;
     -0.77, 2.76;
     0.48, -6.17;
     4.1, 0.21];
[cOut, info] = f08ag(side, trans, a, tau, c)
```

```
cOut =
    3.2456   -2.6896
   -4.5885    3.0574
```

```
   -2.1500   -2.3696
   -0.0931   -7.0279
   -0.0085    0.0009
    0.0204    0.0138
info =
           0
```