NAG Fortran Library Routine Document

F08AKF (SORMLQ/DORMLQ)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of **bold italicised** terms and other implementation-dependent details.

1 Purpose

F08AKF (SORMLQ/DORMLQ) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from an LQ factorization computed by F08AHF (SGELQF/DGELQF).

2 Specification

```
SUBROUTINE F08AKF(SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
1                 LWORK, INFO)
ENTRY      sormlq (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
1                 LWORK, INFO)
INTEGER           M, N, K, LDA, LDC, LWORK, INFO
real              A(LDA,*), TAU(*), C(LDC,*), WORK(*)
CHARACTER*1       SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3 Description

This routine is intended to be used after a call to F08AHF (SGELQF/DGELQF), which performs an LQ factorization of a real matrix A. The orthogonal matrix Q is represented as a product of elementary reflectors.

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

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

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

4 References

Golub G H and van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1:   SIDE – CHARACTER*1   *Input*

On entry: indicates how Q or Q^T is to be applied to C as follows:

    if SIDE = 'L', Q or Q^T is applied to C from the left;

    if SIDE = 'R', Q or Q^T is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2:   TRANS – CHARACTER*1   *Input*

On entry: indicates whether Q or Q^T is to be applied to C as follows:

    if TRANS = 'N', Q is applied to C;

    if TRANS = 'T', Q^T is applied to C.

Constraint: TRANS = 'N' or 'T'.

3:    M – INTEGER   *Input*

   On entry: m, the number of rows of the matrix C.

   Constraint: $M \geq 0$.

4:    N – INTEGER   *Input*

   On entry: n, the number of columns of the matrix C.

   Constraint: $N \geq 0$.

5:    K – INTEGER   *Input*

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

   Constraints:

   $M \geq K \geq 0$ if SIDE = 'L',
   $N \geq K \geq 0$ if SIDE = 'R'.

6:    A(LDA,*) – **real** array   *Input*

   Note: the second dimension of the array A must be at least $\max(1, M)$ if SIDE = 'L' and at least $\max(1, N)$ if SIDE = 'R'.

   On entry: details of the vectors which define the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

   On exit: used as internal workspace prior to being restored and hence is unchanged.

7:    LDA – INTEGER   *Input*

   On entry: the first dimension of the array A as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called.

   Constraint: $LDA \geq \max(1, K)$.

8:    TAU(∗) – **real** array   *Input*

   Note: the dimension of the array TAU must be at least $\max(1, K)$.

   On entry: further details of the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

9:    C(LDC,*) – **real** array   *Input/Output*

   Note: the second dimension of the array C must be at least $\max(1, N)$.

   On entry: the m by n matrix C.

   On exit: C is overwritten by QC or $Q^T C$ or CQ or CQ^T as specified by SIDE and TRANS.

10:   LDC – INTEGER   *Input*

   On entry: the first dimension of the array C as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called.

   Constraint: $LDC \geq \max(1, M)$.

11:   WORK(∗) – **real** array   *Workspace*

   Note: the dimension of the array WORK must be at least $\max(1, LWORK)$.

   On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

12:    LWORK – INTEGER   *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called, unless LWORK $= -1$, in which case a workspace query is assumed and the routine only calculates the optimal dimension of WORK (using the formula given below).

Suggested value: for optimum performance LWORK should be at least N $\times nb$ if SIDE = 'L' and at least M $\times nb$ if SIDE = 'R', where nb is the **blocksize**.

Constraints:

LWORK $\geq \max(1, N)$ or LWORK $= -1$ if SIDE = 'L',
LWORK $\geq \max(1, M)$ or LWORK $= -1$ if SIDE = 'R'.

13:    INFO – INTEGER   *Output*

On exit: INFO $= 0$ unless the routine detects an error (see Section 6).

6   Error Indicators and Warnings

Errors or warnings detected by the routine:

INFO < 0

If INFO $= -i$, the ith parameter 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 routine is F08AXF (CUNMLQ/ZUNMLQ).

9   Example

See Section 9 of the document for F08AHF (SGELQF/DGELQF).