NAG Fortran Library Routine Document

F08KGF (SORMBR/DORMBR)

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

F08KGF (SORMBR/DORMBR) multiplies an arbitrary real matrix C by one of the real orthogonal matrices Q or P which were determined by F08KEF (SGEBRD/DGEBRD) when reducing a real matrix to bidiagonal form.

2 Specification

```
      SUBROUTINE F08KGF(VECT, SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                  LWORK, INFO)
      ENTRY      sormbr (VECT, 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       VECT, 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 F08KEF (SGEBRD/DGEBRD), which reduces a real rectangular matrix A to bidiagonal form B by an orthogonal transformation: $A = QBP^T$. F08KEF represents the matrices Q and P^T as products of elementary reflectors.

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

$$QC,\ Q^TC,\ CQ,\ CQ^T,\ PC,\ P^TC,\ CP \text{ or } CP^T,$$

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

Note: in the description below, r denotes the order of Q or P^T: $r =$ M if SIDE = 'L' and $r =$ N if SIDE = 'R'.

1: VECT – CHARACTER*1 *Input*

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

if VECT = 'Q', Q or Q^T is applied to C;

if VECT = 'P', P or P^T is applied to C.

Constraint: VECT = 'Q' or 'P'.

2: SIDE – CHARACTER*1 *Input*

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

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

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

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

3: TRANS – CHARACTER*1 *Input*

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

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

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

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

4: M – INTEGER *Input*

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

Constraint: M ≥ 0.

5: N – INTEGER *Input*

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

Constraint: N ≥ 0.

6: K – INTEGER *Input*

On entry: if VECT = 'Q', the number of columns in the original matrix A; if VECT = 'P', the number of rows in the original matrix A.

Constraint: K ≥ 0.

7: A(LDA,*) – ***real*** array *Input/Output*

Note: the second dimension of the array A must be at least $\max(1, \min(r, \mathrm{K}))$ if VECT = 'Q' and at least $\max(1, r)$ if VECT = 'P' .

On entry: details of the vectors which define the elementary reflectors, as returned by F08KEF (SGEBRD/DGEBRD).

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

8: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) is called.

Constraints:

LDA $\geq \max(1, r)$ if VECT = 'Q',
LDA $\geq \max(1, \min(r, \mathrm{K}))$ if VECT = 'P'.

9: TAU($*$) – ***real*** array *Input*

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

On entry: further details of the elementary reflectors, as returned by F08KEF (SGEBRD/DGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

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

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

On entry: the matrix C.

On exit: C is overwritten by QC or Q^TC or CQ or CQ^T or PC or P^TC or CP or CP^T as specified by VECT, SIDE and TRANS.

11: LDC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) is called.

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

12: WORK(*) – ***real*** array *Workspace*

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

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

13: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) 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 $\geq \max(1, \mathrm{N})$ or LWORK $= -1$ if SIDE = 'L',
LWORK $\geq \max(1, \mathrm{M})$ or LWORK $= -1$ if SIDE = 'R'.

14: 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

$2n_Ck(2m_C - k)$ if SIDE = 'L' and $m_C \geq k$;

$2m_Ck(2n_C - k)$ if SIDE = 'R' and $n_C \geq k$;

$2m_C^2n_C$ if SIDE = 'L' and $m_C < k$;

$2m_Cn_C^2$ if SIDE = 'R' and $n_C < k$;

where k is the value of the parameter K.

The complex analogue of this routine is F08KUF (CUNMBR/ZUNMBR).

9 Example

For this routine two examples are presented. Both illustrate how the reduction to bidiagonal form of a matrix A may be preceded by a QR or LQ factorization of A.

In the first example, $m > n$, and

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

The routine first performs a QR factorization of A as $A = Q_aR$ and then reduces the factor R to bidiagonal form B: $R = Q_bBP^T$. Finally it forms Q_a and calls F08KGF (SORMBR/DORMBR) to form $Q = Q_aQ_b$.

In the second example, $m < n$, and

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix}.$$

The routine first performs an LQ factorization of A as $A = LP_a^T$ and then reduces the factor L to bidiagonal form B: $L = QBP_b^T$. Finally it forms P_b^T and calls F08KGF (SORMBR/DORMBR) to form $P^T = P_b^TP_a^T$.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08KGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDPT, LDU, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDPT=NMAX,LDU=MMAX,
     +                 LWORK=64*(MMAX+NMAX))
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      INTEGER          I, IC, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), D(NMAX), E(NMAX-1), PT(LDPT,NMAX),
     +                 TAU(NMAX), TAUP(NMAX), TAUQ(NMAX), U(LDU,NMAX),
     +                 WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgebrd, sgelqf, sgeqrf, sorglq, sorgqr, sormbr,
     +                 F06QFF, F06QHF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 IC = 1, 2
         READ (NIN,*) M, N
         IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
*
*           Read A from data file
*
            READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
            IF (M.GE.N) THEN
```

```
*
*              Compute the QR factorization of A
*
               CALL sgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*              Copy A to U
*
               CALL F06QFF('Lower',M,N,A,LDA,U,LDU)
*
*              Form Q explicitly, storing the result in U
*
               CALL sorgqr(M,M,N,U,LDU,TAU,WORK,LWORK,INFO)
*
*              Copy R to PT (used as workspace)
*
               CALL F06QFF('Upper',N,N,A,LDA,PT,LDPT)
*
*              Set the strictly lower triangular part of R to zero
*
               CALL F06QHF('Lower',N-1,N-1,ZERO,ZERO,PT(2,1),LDPT)
*
*              Bidiagonalize R
*
               CALL sgebrd(N,N,PT,LDPT,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*              Update Q, storing the result in U
*
               CALL sormbr('Q','Right','No transpose',M,N,N,PT,LDPT,
     +                     TAUQ,U,LDU,WORK,LWORK,INFO)
*
*              Print bidiagonal form and matrix Q
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 1: bidiagonal matrix B'
               WRITE (NOUT,*) 'Diagonal'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*) 'Super-diagonal'
               WRITE (NOUT,99999) (E(I),I=1,N-1)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',M,N,U,LDU,
     +                     'Example 1: matrix Q',IFAIL)
*
            ELSE
*
*              Compute the LQ factorization of A
*
               CALL sgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*              Copy A to PT
*
               CALL F06QFF('Upper',M,N,A,LDA,PT,LDPT)
*
*              Form Q explicitly, storing the result in PT
*
               CALL sorglq(N,N,M,PT,LDPT,TAU,WORK,LWORK,INFO)
*
*              Copy L to U (used as workspace)
*
               CALL F06QFF('Lower',M,M,A,LDA,U,LDU)
*
*              Set the strictly upper triangular part of L to zero
*
               CALL F06QHF('Upper',M-1,M-1,ZERO,ZERO,U(1,2),LDU)
*
*              Bidiagonalize L
*
               CALL sgebrd(M,M,U,LDU,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*              Update P**T, storing the result in PT
```

```
*
               CALL sormbr('P','Left','Transpose',M,N,M,U,LDU,TAUP,PT,
     +                     LDPT,WORK,LWORK,INFO)
*
*              Print bidiagonal form and matrix P**T
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 2: bidiagonal matrix B'
               WRITE (NOUT,*) 'Diagonal'
               WRITE (NOUT,99999) (D(I),I=1,M)
               WRITE (NOUT,*) 'Super-diagonal'
               WRITE (NOUT,99999) (E(I),I=1,M-1)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',M,N,PT,LDPT,
     +                     'Example 2: matrix P**T',IFAIL)
*
            END IF
         END IF
   20 CONTINUE
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

9.2 Program Data

```
F08KGF Example Program Data
  6  4                                    :Values of M and N, Example 1
 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
  2.30   0.24   0.40  -0.35
 -1.93   0.64  -0.66   0.08
  0.15   0.30   0.15  -2.13
 -0.02   1.03  -1.43   0.50               :End of matrix A
  4  6                                    :Values of M and N, Example 2
 -5.42   3.28  -3.68   0.27   2.06   0.46
 -1.65  -3.40  -3.20  -1.03  -4.06  -0.01
 -0.37   2.35   1.90   4.31  -1.76   1.13
 -3.15  -0.11   1.99  -2.70   0.26   4.50   :End of matrix A
```

9.3 Program Results

```
 F08KGF Example Program Results

 Example 1: bidiagonal matrix B
 Diagonal
     3.6177 -2.4161  1.9213 -1.4265
 Super-diagonal
     1.2587 -1.5262  1.1895

 Example 1: matrix Q
          1       2       3       4
 1  -0.1576 -0.2690  0.2612  0.8513
 2  -0.5335  0.5311 -0.2922  0.0184
 3   0.6358  0.3495 -0.0250 -0.0210
 4  -0.5335  0.0035  0.1537 -0.2592
 5   0.0415  0.5572 -0.2917  0.4523
 6  -0.0055  0.4614  0.8585 -0.0532

 Example 2: bidiagonal matrix B
 Diagonal
    -7.7724  6.1573 -6.0576  5.7933
 Super-diagonal
     1.1926  0.5734 -1.9143

 Example 2: matrix P**T
          1       2       3       4       5       6
 1  -0.7104  0.4299 -0.4824  0.0354  0.2700  0.0603
```

```
 2   0.3583  0.1382 -0.4110  0.4044  0.0951 -0.7148
 3  -0.0507  0.4244  0.3795  0.7402 -0.2773  0.2203
 4   0.2442  0.4016  0.4158 -0.1354  0.7666 -0.0137
```