F08BEF (SGEQPF/DGEQPF) – NAG Fortran Library Routine Document

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

F08BEF (SGEQPF/DGEQPF) computes the QR factorization, with column pivoting, of a real m by n matrix.

2 Specification

```
SUBROUTINE F08BEF(M, N, A, LDA, JPVT, TAU, WORK, INFO)
ENTRY      sgeqpf(M, N, A, LDA, JPVT, TAU, WORK, INFO)
INTEGER           M, N, LDA, JPVT(*), INFO
real              A(LDA,*), TAU(*), WORK(*)
```

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

3 Description

This routine forms the QR factorization with column pivoting of an arbitrary rectangular real m by n matrix.

If $m \geq n$, the factorization is given by:

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where R is an n by n upper triangular matrix, Q is an m by m orthogonal matrix and P is an n by n permutation matrix. It is sometimes more convenient to write the factorization as

$$AP = (Q_1 Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$AP = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m - n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$AP = Q(R_1 R_2),$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < n$, the information returned in the first k columns of the array A represents a QR factorization of the first k columns of the permuted matrix AP.

The routine allows specified columns of A to be moved to the leading columns of AP at the start of the factorization and fixed there. The remaining columns are free to be interchanged so that at the ith stage the pivot column is chosen to be the column which maximizes the 2-norm of elements i to m over columns i to n.

4 References

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

5 Parameters

1: M — INTEGER *Input*

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

Constraint: M $\geq$ 0.

2: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

3: A(LDA,$*$) — ***real*** array *Input/Output*

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

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the orthogonal matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the orthogonal matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

4: LDA — INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08BEF (SGEQPF/DGEQPF) is called.

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

5: JPVT($*$) — INTEGER array *Input/Output*

Note: the dimension of the array JPVT must be at least max(1,N).

On entry: if JPVT(i) $\neq$ 0, then the ith column of A is moved to the beginning of AP before the decomposition is computed and is fixed in place during the computation. Otherwise, the ith column of A is a free column (i.e., one which may be interchanged during the computation with any other free column).

On exit: details of the permutation matrix P. More precisely, if JPVT(i) = k, then the kth column of A is moved to become the ith column of AP; in other words, the columns of AP are the columns of A in the order JPVT(1),JPVT(2),...,JPVT(n).

6: TAU($*$) — ***real*** array *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the orthogonal matrix Q.

7: WORK($*$) — ***real*** array *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

8: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

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 factorization is the exact factorization of a nearby matrix $A + E$, where

$$\| E \|_2 = O(\epsilon) \| A \|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^2(3m - n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n - m)$ if $m < n$.

To form the orthogonal matrix Q this routine may be followed by a call to F08AFF (SORGQR/DORGQR):

```
      CALL SORGQR (M,M,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the second dimension of the array A must be at least M, which may be larger than was required by F08BEF.

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
      CALL SORGQR (M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary real rectangular matrix C, this routine may be followed by a call to F08AGF (SORMQR/DORMQR). For example,

```
      CALL SORMQR ('Left','Transpose',M,P,MIN(M,N),A,LDA,TAU,C,LDC,WORK,
     +             LWORK,INFO)
```

forms $C = Q^T C$, where C is m by p.

To compute a QR factorization without column pivoting, use F08AEF (SGEQRF/DGEQRF).

The complex analogue of this routine is F08BSF (CGEQPF/ZGEQPF).

9 Example

To solve the linear least-squares problem

$$\text{minimize} \| Ax_i - b_i \|_2 \text{ for } i = 1, 2$$

where b_1 and b_2 are the columns of the matrix B,

where

$$A = \begin{pmatrix} -0.09 & 0.14 & -0.46 & 0.68 & 1.29 \\ -1.56 & 0.20 & 0.29 & 1.09 & 0.51 \\ -1.48 & -0.43 & 0.89 & -0.71 & -0.96 \\ -1.09 & 0.84 & 0.77 & 2.11 & -1.27 \\ 0.08 & 0.55 & -1.13 & 0.14 & 1.74 \\ -1.59 & -0.72 & 1.06 & 1.24 & 0.34 \end{pmatrix} \text{ and } B = \begin{pmatrix} -0.01 & -0.04 \\ 0.04 & -0.03 \\ 0.05 & 0.01 \\ -0.03 & -0.02 \\ 0.02 & 0.05 \\ -0.06 & 0.07 \end{pmatrix}.$$

Here A is approximately rank-deficient, and hence it is preferable to use F08BEF (SGEQPF/DGEQPF) rather than F08AEF (SGEQRF/DGEQRF).

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.

```
*     F08BEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, LDX, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=MMAX,LDX=MMAX,
     +                 NRHMAX=NMAX,LWORK=64*NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      real             TOL
      INTEGER          I, IFAIL, INFO, J, K, M, N, NRHS
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                 WORK(LWORK), X(LDX,NRHMAX)
      INTEGER          JPVT(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgeqpf, sormqr, strsv, F06DBF, F06FBF, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08BEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Initialize JPVT to be zero so that all columns are free
*
         CALL F06DBF(N,0,JPVT,1)
*
*        Compute the QR factorization of A
*
         CALL sgeqpf(M,N,A,LDA,JPVT,TAU,WORK,INFO)
*
*        Choose TOL to reflect the relative accuracy of the input data
*
         TOL = 0.01e0
*
*        Determine which columns of R to use
*
         DO 20 K = 1, N
            IF (ABS(A(K,K)).LE.TOL*ABS(A(1,1))) GO TO 40
   20    CONTINUE
*
*        Compute C = (Q**T)*B, storing the result in B
*
```

```
   40    K = K - 1
*
         CALL sormqr('Left','Transpose',M,NRHS,N,A,LDA,TAU,B,LDB,WORK,
     +               LWORK,INFO)
*
*        Compute least-squares solution by backsubstitution in R*B = C
*
         DO 60 I = 1, NRHS
*
            CALL strsv('Upper','No transpose','Non-Unit',K,A,LDA,B(1,I),
     +                 1)
*
*           Set the unused elements of the I-th solution vector to zero
*
            CALL F06FBF(N-K,ZERO,B(K+1,I),1)
*
   60    CONTINUE
*
*        Unscramble the least-squares solution stored in B
*
         DO 100 I = 1, N
            DO 80 J = 1, NRHS
               X(JPVT(I),J) = B(I,J)
   80       CONTINUE
  100    CONTINUE
*
*        Print least-squares solution
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,NRHS,X,LDX,
     +               'Least-squares solution',IFAIL)
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F08BEF Example Program Data
  6  5  2                                 :Values of M, N and NRHS
 -0.09   0.14  -0.46   0.68   1.29
 -1.56   0.20   0.29   1.09   0.51
 -1.48  -0.43   0.89  -0.71  -0.96
 -1.09   0.84   0.77   2.11  -1.27
  0.08   0.55  -1.13   0.14   1.74
 -1.59  -0.72   1.06   1.24   0.34   :End of matrix A
 -0.01  -0.04
  0.04  -0.03
  0.05   0.01
 -0.03  -0.02
  0.02   0.05
 -0.06   0.07                         :End of matrix B
```

9.3 Program Results

```
F08BEF Example Program Results

Least-squares solution
          1       2
1  -0.0370 -0.0044
2   0.0647 -0.0335
3   0.0000  0.0000
4  -0.0515  0.0018
5   0.0066  0.0102
```