NAG Library Routine Document

F08BFF (DGEQP3)

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

F08BFF (DGEQP3) computes the QR factorization, with column pivoting, of a real m by n matrix.

2 Specification

```
SUBROUTINE F08BFF (M, N, A, LDA, JPVT, TAU, WORK, LWORK, INFO)
INTEGER            M, N, LDA, JPVT(*), LWORK, INFO
REAL (KIND=nag_wp) A(LDA,*), TAU(*), WORK(max(1,LWORK))
```

The routine may be called by its LAPACK name ***dgeqp3***.

3 Description

F08BFF (DGEQP3) 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

$$AP = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix},$$

which reduces to

$$AP = Q_1R,$$

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

$$AP = Q\begin{pmatrix} R_1 & R_2 \end{pmatrix},$$

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 F08 Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 9).

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

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 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

1: M – INTEGER *Input*

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

Constraint: $\mathrm{M} \geq 0$.

2: N – INTEGER *Input*

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

Constraint: $\mathrm{N} \geq 0$.

3: $\mathrm{A}(\mathrm{LDA}, *)$ – REAL (KIND=nag_wp) array *Input/Output*

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

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 F08BFF (DGEQP3) is called.

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

5: $\mathrm{JPVT}(*)$ – INTEGER array *Input/Output*

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

On entry: if $\mathrm{JPVT}(j) \neq 0$, then the j th 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 j th 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 $\mathrm{JPVT}(j) = k$, then the kth column of A is moved to become the j th column of AP; in other words, the columns of AP are the columns of A in the order $\mathrm{JPVT}(1), \mathrm{JPVT}(2), \ldots, \mathrm{JPVT}(n)$.

6: $\mathrm{TAU}(*)$ – REAL (KIND=nag_wp) array *Output*

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

On exit: the scalar factors of the elementary reflectors.

7: WORK(max(1, LWORK)) – REAL (KIND=nag_wp) array *Workspace*

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

8: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08BFF (DGEQP3) is called.

If LWORK $= -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

Suggested value: for optimal performance, LWORK $\geq 2 \times \mathrm{N} + (\mathrm{N} + 1) \times nb$, where nb is the optimal ***block size***.

Constraint: LWORK $\geq 3 \times \mathrm{N} + 1$ or LWORK $= -1$.

9: 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 $= -i$, argument i 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 Parallelism and Performance

F08BFF (DGEQP3) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08BFF (DGEQP3) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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 F08BFF (DGEQP3) may be followed by a call to F08AFF (DORGQR):

```
CALL DORGQR(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 F08BFF (DGEQP3).

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

To apply Q to an arbitrary real rectangular matrix C, F08BFF (DGEQP3) may be followed by a call to F08AGF (DORMQR). For example,

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

forms $C = Q^{\mathrm{T}}C$, where C is m by p.

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

The complex analogue of this routine is F08BTF (ZGEQP3).

10 Example

This example solves the linear least squares problems

$$\min_x \|b_j - Ax_j\|_2, \quad j = 1, 2$$

for the basic solutions x_1 and x_2, where

$$A = \begin{pmatrix} -0.09 & 0.14 & -0.46 & 0.68 & 1.29 \\ -1.56 & 0.20 & 0.29 & 1.09 & 0.51 \\ -1.48 & -0.43 & 0.89 & -0.71 & -0.96 \\ -1.09 & 0.84 & 0.77 & 2.11 & -1.27 \\ 0.08 & 0.55 & -1.13 & 0.14 & 1.74 \\ -1.59 & -0.72 & 1.06 & 1.24 & 0.34 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 7.4 & 2.7 \\ 4.2 & -3.0 \\ -8.3 & -9.6 \\ 1.8 & 1.1 \\ 8.6 & 4.0 \\ 2.1 & -5.7 \end{pmatrix}$$

and b_j is the jth column of the matrix B. The solution is obtained by first obtaining a QR factorization with column pivoting of the matrix A. A tolerance of 0.01 is used to estimate the rank of A from the upper triangular factor, R.

Note that the block size (NB) of 64 assumed in this example is not realistic for such a small problem, but should be suitable for large problems.

10.1 Program Text

```
    Program f08bffe

!     F08BFF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: dgeqp3, dnrm2, dormqr, dtrsm, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0E0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0E0_nag_wp
      Integer, Parameter               :: inc1 = 1, nb = 64, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: tol
      Integer                          :: i, ifail, info, j, k, lda, ldb,      &
                                          lwork, m, n, nrhs
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), b(:,:), rnorm(:), tau(:),    &
                                          work(:)
      Integer, Allocatable             :: jpvt(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs
!     .. Executable Statements ..
      Write (nout,*) 'F08BFF Example Program Results'
      Write (nout,*)
```

```
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, nrhs
      lda = m
      ldb = m
      lwork = 2*n + (n+1)*nb
      Allocate (a(lda,n),b(ldb,nrhs),rnorm(n),tau(n),work(lwork),jpvt(n))

!     Read A and B from data file

      Read (nin,*)(a(i,1:n),i=1,m)
      Read (nin,*)(b(i,1:nrhs),i=1,m)

!     Initialize JPVT to be zero so that all columns are free

      jpvt(1:n) = 0

!     Compute the QR factorization of A
!     The NAG name equivalent of dgeqp3 is f08bff
      Call dgeqp3(m,n,a,lda,jpvt,tau,work,lwork,info)

!     Compute C = (C1) = (Q**T)*B, storing the result in B
!                 (C2)
!     The NAG name equivalent of dormqr is f08agf
      Call dormqr('Left','Transpose',m,nrhs,n,a,lda,tau,b,ldb,work,lwork,info)

!     Choose TOL to reflect the relative accuracy of the input data

      tol = 0.01_nag_wp

!     Determine and print the rank, K, of R relative to TOL

loop: Do k = 1, n
        If (abs(a(k,k))<=tol*abs(a(1,1))) Exit loop
      End Do loop
      k = k - 1

      Write (nout,*) 'Tolerance used to estimate the rank of A'
      Write (nout,99999) tol
      Write (nout,*) 'Estimated rank of A'
      Write (nout,99998) k
      Write (nout,*)
      Flush (nout)

!     Compute least-squares solutions by backsubstitution in
!     R(1:K,1:K)*Y = C1, storing the result in B

      Call dtrsm('Left','Upper','No transpose','Non-Unit',k,nrhs,one,a,lda,b, &
        ldb)

!     Compute estimates of the square roots of the residual sums of
!     squares (2-norm of each of the columns of C2)

!     The NAG name equivalent of dnrm2 is f06ejf
      Do j = 1, nrhs
        rnorm(j) = dnrm2(m-k,b(k+1,j),inc1)
      End Do

!     Set the remaining elements of the solutions to zero (to give
!     the basic solutions)

      b(k+1:n,1:nrhs) = zero

!     Permute the least-squares solutions stored in B to give X = P*Y

      Do j = 1, nrhs
        work(jpvt(1:n)) = b(1:n,j)
        b(1:n,j) = work(1:n)
      End Do

!     Print least-squares solutions
```

```
!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call x04caf('General',' ',n,nrhs,b,ldb,'Least-squares solution(s)', &
        ifail)

!     Print the square roots of the residual sums of squares

      Write (nout,*)
      Write (nout,*) 'Square root(s) of the residual sum(s) of squares'
      Write (nout,99999) rnorm(1:nrhs)

99999 Format (5X,1P,6E11.2)
99998 Format (1X,I8)
    End Program f08bffe
```

10.2 Program Data

```
F08BFF 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

  7.4    2.7
  4.2   -3.0
 -8.3   -9.6
  1.8    1.1
  8.6    4.0
  2.1   -5.7                       :End of matrix B
```

10.3 Program Results

```
 F08BFF Example Program Results

 Tolerance used to estimate the rank of A
        1.00E-02
 Estimated rank of A
        4

 Least-squares solution(s)
             1          2
 1      0.9767     4.0159
 2      1.9861     2.9867
 3      0.0000     0.0000
 4      2.9927     2.0032
 5      4.0272     0.9976

 Square root(s) of the residual sum(s) of squares
        2.54E-02   3.65E-02
```