F08AEFP (PDGEQRF)

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

1 Description

F08AEFP (PDGEQRF) computes the QR factorization of a real m by n matrix A_s (i.e., $A_s = QR$), where A_s is a submatrix of a larger m_A by n_A matrix A, i.e.,

$$A_s(1:m, 1:n) \equiv A(i_A : i_A + m - 1, j_A : j_A + n - 1).$$

Note: if $i_A = j_A = 1$, $m = m_A$ and $n = n_A$, then $A_s = A$.

The orthogonal matrix Q is not formed explicitly but is represented as a product of elementary reflectors

$$Q = H_0 H_1 \ldots H_{k-1}, \quad \text{where } k = \min(m, n).$$

Each H_ℓ has the form

$$H_\ell = I - \tau_\ell v_\ell v_\ell^T,$$

where τ is a real scalar, and v is real vector. No pivoting is performed by F08AEFP (PDGEQRF).

2 Specification

```
SUBROUTINE F08AEFP(M, N, A, IA, JA, IDESCA, TAU, WORK, LWORK, INFO)
ENTRY      PDGEQRF(M, N, A, IA, JA, IDESCA, TAU, WORK, LWORK, INFO)
DOUBLE PRECISION   A(*), TAU(*), WORK(LWORK)
INTEGER            M, N, IA, JA, IDESCA(9), LWORK, INFO
```

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

3 Data Distribution

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m_p	–	the number of rows in the logical processor grid.
n_p	–	the number of columns in the logical processor grid.
p_r	–	the row grid coordinate of the calling processor.
p_c	–	the column grid coordinate of the calling processor.
M_b^X	–	the blocking factor for the distribution of the rows of a matrix X.
N_b^X	–	the blocking factor for the distribution of the columns of a matrix X.
numroc$(\alpha, b_\ell, q, s, k)$	–	a function which gives the **num**ber of **r**ows **o**r **c**olumns of a distributed matrix owned by the processor with the row or column coordinate q (p_r or p_c), where α is the total number of rows or columns of the matrix, b_ℓ is the blocking factor used (M_b^X or N_b^X), s is the row or column coordinate of the processor that possesses the first row or column of the distributed matrix and k is either n_p or m_p. The Library provides the function Z01CAFP (NUMROC) for the evaluation of this function.
indxg2p(i_g, b_ℓ, q, s, k)	–	a function which gives the processor row or column coordinate which possess the row or column index i_g of the distributed full matrix A. The arguments b_ℓ, q, s and k have the same meaning as in the function numroc. The Library provides the function Z01CDFP (INDXG2P) for the evaluation of this function.

3.2 Global and Local Arguments

The input arguments M, N, IA, JA and the array elements IDESCA(1) and IDESCA(3),...,IDESCA(8) are all global and so must have the same values on entry to the routine on every processor. The output argument INFO is global and so will have the same value on exit from the routine on each processor. The remaining arguments are local.

3.3 Distribution Strategy

The matrix A must be partitioned into M_b^A by N_b^A rectangular blocks which are stored in an array A in a cyclic 2-d block distribution. This data distribution is described in more detail in the F08 Chapter Introduction. The array TAU is distributed across the processor columns in a cyclic 2-d block fashion, and is aligned with the rows matrix A.

This routine assumes that the data has already been correctly distributed, and if this is not the case will fail to produce correct results. However, the Library provides some utility routines which assist you in distributing data correctly. Descriptions of these routines can be found in Chapters F01 and X04 of the NAG Parallel Library Manual.

4 Arguments

Warning: This routine is derived from ScaLAPACK and accurately reflects the specification of the equivalent ScaLAPACK routine. The current release (1.2) of ScaLAPACK imposed a global change in the specification of descriptor arrays. Consequently any applications developed using this routine from Release 1 of the Library will not run correctly, without change, using this Release.

1: M — INTEGER *Global Input*

On entry: the number of rows of the submatrix A_s, m.

Constraint: $0 \leq \text{M} \leq \text{IDESCA}(3)$.

2: N — INTEGER *Global Input*

On entry: the number of columns of the submatrix A_s, n.

Constraint: $0 \leq \text{N} \leq \text{IDESCA}(4)$.

3: A($*$) — DOUBLE PRECISION array *Local Input/Local Output*

Note: array A is formally defined as a vector. However, you may find it more convenient to consider A as a 2-d array of dimension (IDESCA(9),γ), where
$\gamma \geq \text{numroc}(\text{JA}+\text{N}-1, \text{IDESCA}(6), p_c, \text{IDESCA}(8), n_p)$. See Section 8.

On entry: the local part of the matrix A which may contain parts of the m by n submatrix A_s to be factorized.

On exit: if $m \geq n$, the elements below the diagonal of A_s 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 of A_s 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: IA — INTEGER *Global Input*

On entry: the row index of matrix A, i_A, that identifies the first row of the submatrix A_s to be factorized.

Constraint: $1 \leq \text{IA} \leq \text{IDESCA}(3) - \text{M} + 1$.

5: JA — INTEGER *Global Input*

On entry: the column index of matrix A, j_A, that identifies the first column of the submatrix A_s to be factorized.

Constraint: $1 \leq \text{JA} \leq \text{IDESCA}(4) - \text{N} + 1$.

6: IDESCA(9) — INTEGER array *Local Input*

Distribution: the array elements IDESCA(1) and IDESCA(3),...,IDESCA(8) must be global to the processor grid and the elements IDESCA(2) and IDESCA(9) are local to each processor.

On entry: the description array for the matrix A. This array must contain details of the distribution of the matrix A and the logical processor grid.

IDESCA(1), the descriptor type. For this routine, which uses a cyclic 2-d block distribution, IDESCA(1) = 1;
IDESCA(2), the BLACS context (ICNTXT) for the processor grid, usually returned by Z01AAFP;
IDESCA(3), the number of rows, m_A, of the matrix A;
IDESCA(4), the number of columns, n_A, of the matrix A;
IDESCA(5), the blocking factor, M_b^A, used to distribute the rows of the matrix A;
IDESCA(6), the blocking factor, N_b^A, used to distribute the columns of the matrix A;
IDESCA(7), the processor row index over which the first row of the matrix A is distributed;
IDESCA(8), the processor column index over which the first column of the matrix A is distributed;
IDESCA(9), the leading dimension of the conceptual 2-d array A.

Constraints:

IDESCA(1) = 1;
IDESCA(3) ≥ 0; IDESCA(4) ≥ 0;
IDESCA(5) ≥ 1; IDESCA(6) ≥ 1;
$0 \leq$ IDESCA(7) $\leq m_p - 1$; $0 \leq$ IDESCA(8) $\leq n_p - 1$;
IDESCA(9) $\geq$ max(1,numroc(IDESCA(3),IDESCA(5),p_r,IDESCA(7),m_p)).

7: TAU($*$) — DOUBLE PRECISION array *Local Output*

Note: the dimension of the array TAU must be at least α, where
α = numroc(β,IDESCA(6),p_c,IDESCA(8),n_p) and β = JA + min(M,N) $-$ 1.

On exit: the scalar factors τ_ℓ of the elementary reflectors H_ℓ.

8: WORK(LWORK) — DOUBLE PRECISION array *Local Workspace/Local Output*

On exit: WORK(1) contains the minimum required value of LWORK.

9: LWORK — INTEGER *Local Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AEFP (PDGEQRF) is called.

Constraint: LWORK $\geq$ IDESCA(6) $\times$ ($c_1 + c_2$+IDESCA(6)), where

c_1 = numroc(M+d_1,IDESCA(5),p_r,e_1,m_p);
c_2 = numroc(N+d_2,IDESCA(6),p_c,e_2,n_p);
d_1 = mod(IA$-$1,IDESCA(5));
d_2 = mod(JA$-$1,IDESCA(6));
e_1 = indxg2p(IA,IDESCA(5),p_r,IDESCA(7),m_p);
e_2 = indxg2p(JA,IDESCA(6),p_c,IDESCA(8),n_p).

This value of LWORK can be calculated by using the Library function Z01CBFP; i.e.,

LWORK = Z01CBFP(M,N,IA,JA,IDESCA)

10: INFO — INTEGER *Global Output*

On exit: INFO = 0 unless the routine detects an error (see Section 5).

5 Errors and Warnings

If INFO < 0 an explanatory message is output and control returned to the calling program.

INFO < 0

On entry, one of the arguments was invalid:

if the kth argument is a scalar INFO $= -k$;
if the kth argument is an array and its jth element is invalid, INFO $= -(100 \times k + j)$.

This error occurred either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect. An explanatory message distinguishes between these two cases.

6 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 solve the system of equations $A_sX = B_s$, this routine may be followed by a call to F08AFFP (PDORGQR) which forms Q explicitly. In terms of the QR factorization, the linear system $A_sX = B_s$ is given by $QRX = B_s$, and hence $RX = Q^TB_s$. If R is triangular then X may be computed using the PBLAS routine PDTRSM. F08AGFP (PDORMQR) may be used to form Q^TB.

6.1 Algorithmic Detail

For a general m by n matrix A, if $m \geq n$, the factorization is given by:

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

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

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

which reduces to

$$A = Q_1R,$$

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

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

$$A = Q(R_1R_2)$$

where R_1 is an m by m upper triangular matrix and R_2 is an m by n rectangular matrix.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors. See Anderson *et al.* [1] for details of the block method used by the routine.

6.2 Parallelism Detail

The Level 3 BLAS operations are carried out in parallel within the routine.

6.3 Accuracy

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

$$\|E\|_2 = \epsilon p(m,n)\|A\|_2,$$

ϵ is ***machine precision*** and $p(m,n)$ is a modest function of m and n.

7 References

[1] Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S and Sorensen D (1995) *LAPACK Users' Guide* (2nd Edition) SIAM, Philadelphia

[2] Golub G H and Van Loan C F (1989) *Matrix Computations* Johns Hopkins University Press (2nd Edition), Baltimore

8 Example

To solve the linear least-squares problem

$$\min \|Ax^{(i)-c^{(i)}\|_2} \quad \text{for} \quad {\scriptstyle i=1,2}$$

where $c^{(1)}$ and $c^{(2)}$ are the columns of the matrix C,

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix} \text{ and } \quad C = \begin{pmatrix} -3.15 & 2.19 \\ -0.11 & -3.64 \\ 1.99 & 0.57 \\ -2.70 & 8.23 \\ 0.26 & -6.35 \\ 4.50 & -1.48 \end{pmatrix}.$$

The example uses a 2 by 2 logical processor grid and a 2 by 2 block for both A and C.

Note: the listing of the Example Program presented below does not give a full pathname for the data file being opened, but in general the user must give the full pathname in this and any other OPEN statement.

8.1 Example Text

```
*     F08AEFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          DT
      PARAMETER        (DT=1)
      INTEGER          MB, NB
      PARAMETER        (MB=2,NB=MB)
      INTEGER          MMAX, NMAX, LDA, RHSMAX, LDC, IAROW, IACOL,
     +                 ICROW, ICCOL, LW, LIWORK
      PARAMETER        (MMAX=6,NMAX=4,LDA=MMAX,RHSMAX=2,LDC=MMAX,
     +                 IAROW=0,IACOL=0,ICROW=0,ICCOL=0,LW=100,
     +                 LIWORK=100)
*     .. Local Scalars ..
      DOUBLE PRECISION RCOND
      INTEGER          IA, IC, ICNTXT, IFAIL, INFO, JA, JC, LWORKE,
     +                 LWORKG, M, MP, N, NP, NRHS
      LOGICAL          ROOT
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,NMAX), C(LDC,RHSMAX), TAU(MMAX), WORK(LW)
      INTEGER          IDESCA(9), IDESCC(9), IWORK(LIWORK)
*     .. External Functions ..
      DOUBLE PRECISION X02AJF
      INTEGER          Z01CBFP, Z01CCFP
      LOGICAL          Z01ACFP
      EXTERNAL         X02AJF, Z01CBFP, Z01CCFP, Z01ACFP
*     .. External Subroutines ..
```

```
      EXTERNAL         F07TGFP, F08AEFP, F08AGFP, PDTRSM, X04BCFP,
     +                 X04BDFP, Z01AAFP, Z01ABFP
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
*     .. Executable Statements ..
*
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F08AEFP Example Program Results'
*
      MP = 2
      NP = 2
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      OPEN (NIN,FILE='f08aefpe.d')
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS, FORMAT

      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. NRHS.LE.RHSMAX) THEN
*
*         Set array descriptor for A, and read A from data file
*
         IDESCA(1) = DT
         IDESCA(2) = ICNTXT
         IDESCA(3) = M
         IDESCA(4) = N
         IDESCA(5) = MB
         IDESCA(6) = NB
         IDESCA(7) = IAROW
         IDESCA(8) = IACOL
         IDESCA(9) = LDA
*
         IFAIL = 0
         CALL X04BCFP(NIN,M,N,A,1,1,IDESCA,IFAIL)
*
*         Set array descriptor for C, and read C from data file
*
         IDESCC(1) = DT
         IDESCC(2) = ICNTXT
         IDESCC(3) = M
         IDESCC(4) = NRHS
         IDESCC(5) = MB
         IDESCC(6) = NB
         IDESCC(7) = ICROW
         IDESCC(8) = ICCOL
         IDESCC(9) = LDC
*
         IFAIL = 0
*
         CALL X04BCFP(NIN,M,NRHS,C,1,1,IDESCC,IFAIL)
*
*         Factorize A
*
         IA = 1
         JA = 1
         LWORKE = MIN(LW,Z01CBFP(M,N,IA,JA,IDESCA))
```

```
*
         CALL F08AEFP(M,N,A,IA,JA,IDESCA,TAU,WORK,LWORKE,INFO)

         IF (INFO.EQ.0) THEN
*
*            Compute solution
*
            IC = 1
            JC = 1
            LWORKG = MIN(LW,Z01CCFP('L',M,N,IA,JA,IC,JC,IDESCA,IDESCC))
*
*             Apply Q to the right hand sides
*
            CALL F08AGFP('L','T',M,NRHS,N,A,IA,JA,IDESCA,TAU,C,IC,JC,
     +                   IDESCC,WORK,LWORKG,INFO)
            IF (INFO.EQ.0) THEN
*
*                Check the condition of the upper triangle R
*
               CALL F07TGFP('1-norm','Upper','Non-unit',N,A,IA,JA,
     +                      IDESCA,RCOND,WORK,LW,IWORK,LIWORK,INFO)
               IF (INFO.EQ.0 .AND. RCOND.GT.X02AJF()) THEN
*
*                    Solve the triangular system
*
                  CALL PDTRSM('Left','Upper','No transpose','Non-unit',
     +                        N,NRHS,1.0D0,A,IA,JA,IDESCA,C,IC,JC,
     +                        IDESCC)
*
                  IF (ROOT) THEN
                     WRITE (NOUT,*)
                     WRITE (NOUT,*) 'Least-squares solution(s).'
                     WRITE (NOUT,*)
                  END IF

                  IFAIL = 0
                  CALL X04BDFP(NOUT,N,NRHS,C,IC,JC,IDESCC,FORMAT,WORK,
     +                         IFAIL)
               ELSE
                  IF (INFO.EQ.0 .AND. ROOT) WRITE (NOUT,*)
     +                'Matrix is singular to working precision'
               END IF
            END IF
         END IF
*
      END IF
*
      CLOSE (NIN)
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
```

8.2 Example Data

```
F08AEFP Example Program Data
```

```
 6 4 2 '(4F12.4)'       :Values of M, N, NRHS and FORMAT
-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
-3.15  2.19
-0.11 -3.64
 1.99  0.57
-2.70  8.23
 0.26 -6.35
 4.50 -1.48              :End of matrix C
```

8.3 Example Results

```
F08AEFP Example Program Results

Least-squares solution(s).

     1.5146     -1.5838
     1.8621      0.5536
    -1.4467      1.3491
     0.0396      2.9600
```