F08AFFP (PDORGQR)

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

This routine is intended to be used after a call to F08AEFP (PDGEQRF), which performs a QR factorization of an m by r real matrix A_s. F08AEFP represents the m by m orthogonal matrix Q as a product of elementary reflectors.

F08AFFP (PDORGQR) may be used to generate Q explicitly as a square matrix, or to form the n leading columns of Q, where $n \leq m$.

Alternatively, this routine may be called to compute the orthogonal matrix Q of the QR factorization of the k leading columns of the matrix A_s for $k \leq r$.

This routine returns the matrix Q in array A which on entry contained the details of elementary reflectors computed by F08AEFP (PDGEQRF). The position of the $(1,1)$ element of the matrix Q in the array A is given by (i_A, j_A), which originally identified the $(1,1)$ element of the matrix A_s in the factorization routine F08AEFP (PDGEQRF). The distribution of the matrix Q conforms to the details in the description array IDESCA.

2 Specification

```
      SUBROUTINE F08AFFP(M, N, K, A, IA, JA, IDESCA, TAU, WORK, LWORK,
     1                   INFO)
      ENTRY      PDORGQR(M, N, K, A, IA, JA, IDESCA, TAU, WORK, LWORK,
     1                   INFO)
      DOUBLE PRECISION   A(*), TAU(*), WORK(LWORK)
      INTEGER            M, N, K, 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, K, IA, JA and the array elements IDESCA(1) and IDESCA(3),...,IDESCA(8) are all global and so must have the same value 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

On entry to this routine, the input values of M, A, IA, JA, IDESCA and TAU must be identically equal to the output values of the corresponding arguments on exit from the QR factorization routine F08AEFP (PDGEQRF).

In F08AEFP (PDGEQRF), the matrix Q is represented as a set of elementary reflectors but in F08AFFP (PDORGQR), the matrix Q is explicitly computed. In both these routines, the details of Q are stored in local arrays denoted by A but the minimal storage requirements (as specified by the second dimensions) of A on a particular processor are, in general, different. In particular, if $r < n$ then F08AFFP (PDORGQR) requires larger local arrays for A than in F08AEFP (PDGEQRF).

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: m, the number of rows of the Q. This should be identical to the number of rows of the matrix A_s as supplied to the QR factorization routine F08AEFP (PDGEQRF).

Constraint: $0 \le$ M $\le$ IDESCA(3).

2: N — INTEGER *Global Input*

On entry: n, the number of columns of the matrix Q that are required.

Constraints: $0 \le$ N $\le$ IDESCA(4); N $\le$ M.

3: K — INTEGER *Global Input*

On entry: k, the number of columns of the matrix A_s which should be used in the computation of the matrix Q. This index also identifies the number of elementary reflectors whose product defines Q.

Constraint: $0 \le$ K $\le$ N.

4: 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
$\gamma \ge$ numroc(JA + max(N,K) $-$ 1, IDESCA(6), p_c, IDESCA(8), n_p).

On entry: details of the vectors which define the elementary reflectors as returned by F08AEFP (PDGEQRF).

On exit: the local parts of the first n columns of the m by m matrix Q. The distribution of the matrix Q is defined by the indices IA and JA and the description array IDESCA.

5: IA — INTEGER *Global Input*

On entry: the row index of matrix A, i_A, that identifies the first row of Q to be generated.

Constraints: $1 \le$ IA $\le$ IDESCA(3) $-$ M + 1.

6: JA — INTEGER *Global Input*

On entry: the column index of matrix A, j_A, that identifies the first column of Q to be generated.

Constraints: $1 \leq \text{JA} \leq \text{IDESCA}(4) - \max(\text{N,K}) + 1$.

7: 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)).

8: TAU(∗) — DOUBLE PRECISION array *Local Input*

Note: the dimension of the array TAU must be at least α, where
α = numroc(JA + K − 1, IDESCA(6), p_c, IDESCA(8), n_p).

On entry: details of the elementary reflectors, as returned by a call to F08AEFP (PDGEQRF).

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

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

10: LWORK — INTEGER *Local Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AFFP (PDORGQR) is called.

Constraint: LWORK $\geq$ IDESCA(6) × ($c_1 + c_2$ + 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)

11: 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 the jth element is invalid, INFO $= -(100 \times k + j)$.

This error occured 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

Often Q is determined from the QR factorization of an m by r matrix A with $m \geq r$. The matrix Q may be computed by calling:

```
CALL F08AFFP (M,M,r,A,IA,JA,IDESCA,TAU,WORK,LWORK,INFO)
```

The leading r columns of Q may be obtained by:

```
CALL F08AFFP (M,r,r,A,IA,JA,IDESCA,TAU,WORK,LWORK,INFO)
```

The columns of Q returned by the last call form an orthonormal basis for the space spanned by the columns of A; thus F08AEFP (PDGEQRF) followed by F08AFFP (PDORGQR) can be used to orthogonalise the columns of A.

6.1 Algorithmic Detail

See [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 matrix Q differs from an exactly orthogonal matrix by a matrix E such that

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

where ϵ is the ***machine precision***, $p(m,n)$ 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 form the leading four columns of the orthogonal matrix Q from the QR factorization of the matrix A, where

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

The columns of Q form an orthonormal basis for the space spanned by the columns of A. The example uses a 2 by 2 logical processor grid and a 1 by 1 block for A.

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

```
*     F08AFFP 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, LW
      PARAMETER        (MB=2,NB=MB,LW=100)
      INTEGER          MMAX, NMAX, LDA, IAROW, IACOL
      PARAMETER        (MMAX=6,NMAX=4,LDA=MMAX,IAROW=0,IACOL=0)
*     .. Local Scalars ..
      INTEGER          IA, ICNTXT, IFAIL, INFO, JA, LWORK, M, MP, N, NP
      LOGICAL          ROOT
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,NMAX), TAU(MMAX), WORK(LW)
      INTEGER          IDESCA(9)
*     .. External Functions ..
      INTEGER          Z01CBFP
      LOGICAL          Z01ACFP
      EXTERNAL         Z01CBFP, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F08AEFP, F08AFFP, X04BCFP, X04BDFP, Z01AAFP,
     +                 Z01ABFP
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F08AFFP Example Program Results'
*
      MP = 2
      NP = 2
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      OPEN (NIN,FILE='f08affpe.d')
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, FORMAT
```

```
*
      IF (M.LE.MMAX .AND. N.LE.NMAX) 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)
*
         IA = 1
         JA = 1
         LWORK = MIN(LW,Z01CBFP(M,N,IA,JA,IDESCA))
*
         CALL F08AEFP(M,N,A,IA,JA,IDESCA,TAU,WORK,LWORK,INFO)
         IF (INFO.EQ.0) THEN
*
*           Generate Q
*
            CALL F08AFFP(M,N,N,A,IA,JA,IDESCA,TAU,WORK,LWORK,INFO)
*
            IF (INFO.EQ.0) THEN
               IF (ROOT) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,99999) N
                  WRITE (NOUT,*)
               END IF
               IFAIL = 0
*
               CALL X04BDFP(NOUT,M,N,A,IA,JA,IDESCA,FORMAT,WORK,IFAIL)
*
            END IF
         END IF
*
      END IF
*
      CLOSE (NIN)
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
*
99999 FORMAT (' The leading ',I2,' columns of Q')
      END
```

8.2 Example Data

```
F08AFFP Example Program Data
 6 4 '(4F12.4)'           :Values of M, N 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
```

8.3 Example Results

```
 F08AFFP Example Program Results

 The leading  4 columns of Q

     -0.1576      0.6744     -0.4571      0.4489
     -0.5335     -0.3861      0.2583      0.3898
      0.6358     -0.2928      0.0165      0.1930
     -0.5335     -0.1692     -0.0834     -0.2350
      0.0415     -0.1593      0.1475      0.7436
     -0.0055     -0.5064     -0.8339      0.0335
```