F08AGFP (PDORMQR)
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 (PDGEQRF) represents the real orthogonal matrix Q as a product of elementary reflectors.

F08AGFP (PDORMQR) multiplies an m by n real matrix C_s by Q, where C_s is a submatrix of a larger m_C by n_C matrix C, i.e.,

$$C_s(1:m, 1:n) \equiv C(i_C : i_C + m - 1, j_C : j_C + n - 1).$$

Note: if $i_c = j_c = 1$, $m = m_c$ and $n = n_c$, then $C_s = C$.

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

$$QC_s, Q^T C_s, C_s Q \quad \text{or} \quad C_s Q^T,$$

overwriting the result on C_s.

2 Specification

```
 SUBROUTINE F08AGFP(SIDE, TRANS, M, N, K, A, IA, JA, IDESCA, TAU,
1                   C, IC, JC, IDESCC, WORK, LWORK, INFO)
 ENTRY      PDORMQR(SIDE, TRANS, M, N, K, A, IA, JA, IDESCA, TAU,
1                   C, IC, JC, IDESCC, WORK, LWORK, INFO)
 DOUBLE PRECISION   A(*), TAU(*), C(*), WORK(*)
 INTEGER            M, N, K, IA, JA, IDESCA(*), IC, JC, IDESCC(*),
1                   LWORK, INFO
 CHARACTER*1        SIDE, TRANS
```

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

3 Usage

3.1 Definitions

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

m_p	–	the number of rows in the Library Grid.
n_p	–	the number of columns in the Library 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(α,b_ℓ,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 m_p or n_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 following global **input** arguments must have the same value on entry to the routine on each processor and the global **output** arguments will have the same value on exit from the routine on each processor:

Global input arguments:	SIDE, TRANS, M, N, K, IA, JA, IC, JC, IDESCA(1), IDESCA(3:8), IDESCC(1), IDESCC(3:8)
Global output arguments:	INFO

The remaining arguments are local.

3.3 Distribution Strategy

On entry to this routine, the input values of A, IA, JA, IDESCA and TAU must be identical to the output values of the corresponding arguments on exit from the QR factorization routine F08AEFP (PDGEQRF). If the product QC on Q^TC are computed, M must be equal to m, otherwise N must be equal to m.

The matrix C should be partitioned into M_b^C by N_b^C rectangular blocks, which are stored in the array C in a cyclic two-dimensional block distribution. This data distribution is described in more detail in the the F08 Chapter Introduction.

3.4 Related Routines

This routine assumes that the data has already been correctly distributed, and if this is not the case will fail to produce correct results. The Library provides many support routines for the generation, scattering/gathering and input/output of matrices/vectors in cyclic two-dimensional block form. The following routines may be used in conjunction with F08AGFP (PDORMQR):

Real matrix generation:	F01ZQFP
Real matrix input:	X04BCFP
Real matrix output:	X04BDFP
Real matrix gather:	F01WAFP
Real matrix scatter:	F01WNFP

4 Arguments

1: SIDE — CHARACTER*1 *Global Input*

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

if SIDE = 'L', then Q or Q^T is applied to C_s from the left;
if SIDE = 'R', then Q or Q^T is applied to C_s from the right.

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

2: TRANS — CHARACTER*1 *Global Input*

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

if TRANS = 'N', then Q is applied to C_s;
if TRANS = 'T', then Q^T is applied to C_s.

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

3: M — INTEGER *Global Input*

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

Constraints:

if SIDE = 'L', $0 \leq$ M$\leq$ min(IDESCA(3), IDESCC(3));
if SIDE = 'R', $0 \leq$ M $\leq$ IDESCC(3).

4: N — INTEGER *Global Input*

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

Constraints:

if SIDE = 'R', $0 \leq$ N $\leq \min$(IDESCA(3), IDESCC(4));
if SIDE = 'L', $0 \leq$ N $\leq$ IDESCC(4).

5: K — INTEGER *Global Input*

On entry: k, the number of elementary reflectors whose product defines Q.

Constraints:

if SIDE = 'L', $0 \leq$ K $\leq$ M;
if SIDE = 'R', $0 \leq$ K $\leq$ N.

6: 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 two-dimensional array of dimension (IDESCA(9),γ), where
$\gamma \geq$ numroc(JA+K−1,IDESCA(6),p_c,IDESCA(8),n_p). See Section 8 of the document for F08AEFP (PDGEQRF).

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

On exit: A is used as workspace, but restored on exit.

7: IA — INTEGER *Global Input*

On entry: i_A, the row index of matrix A that identified the first row of the submatrix A_s as defined in F08AEFP (PDGEQRF).

Constraints: $1 \leq$ IA $\leq$ IDESCA(3) − M + 1.

8: JA — INTEGER *Global Input*

On entry: j_A, the column index of matrix A that identified the first column of the submatrix A_s as defined in F08AEFP (PDGEQRF).

Constraints: $1 \leq$ JA $\leq$ IDESCA(4) − K + 1.

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

Note: the dimension of the array IDESCA must be at least 9.

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 as defined in the QR factorization routine F08AEFP (PDGEQRF). 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 two-dimensional block distribution, IDESCA(1) = 1;
IDESCA(2), the Library context, usually returned by a call to the Library Grid initialisation routine 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 two-dimensional array A.

Constraints:

IDESCA(1) = 1;
IDESCA(3) $\geq$ 0; IDESCA(4) $\geq$ 0; IDESCA(5) $\geq$ 1; IDESCA(6) $\geq$ 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)).

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

Note: the dimension of the array TAU must be at least 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).

11: C(∗) — DOUBLE PRECISION array *Local Input/Local Output*

Note: array C is formally defined as a vector. However, you may find it more convenient to consider C as a two-dimensional array of dimension (IDESCC(9),γ), where $\gamma \geq$ numroc(JC+N−1,IDESCC(6),p_c,IDESCC(8),n_p).

On entry: the local part of the matrix C.

On exit: overwritten by QC_s, Q^TC_s, C_sQ or C_sQ^T.

12: IC — INTEGER *Global Input*

On entry: i_C, the row index of matrix C that identifies the first row of the submatrix C_s.

Constraint: 1 $\leq$ IC $\leq$ IDESCC(3) − M + 1.

13: JC — INTEGER *Global Input*

On entry: j_C, the column index of matrix C that identifies the first column of the submatrix C_s.

Constraint: 1 $\leq$ JC $\leq$ IDESCC(4) − N + 1.

14: IDESCC(∗) — INTEGER array *Local Input*

Note: the dimension of the array IDESCC must be at least 9.

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

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

IDESCC(1), the descriptor type. For this routine, which uses a cyclic two-dimensional block distribution, IDESCC(1) = 1;
IDESCC(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;
IDESCC(3), the number of rows, m_C, of the matrix C;
IDESCC(4), the number of columns, n_C, of the matrix C;
IDESCC(5), the blocking factor, M_b^C, used to distribute the rows of the matrix C;
IDESCC(6), the blocking factor, N_b^C, used to distribute the columns of the matrix C;
IDESCC(7), the processor row index over which the first row of the matrix C is distributed;
IDESCC(8), the processor column index over which the first column of the matrix C is distributed;
IDESCC(9), the leading dimension of the conceptual two-dimensional array C.

Constraints:

IDESCC(1) = 1;
IDESCC(2) = IDESCA(2)
IDESCC(3) ≥ 0; IDESCC(4) ≥ 0;
IDESCC(5) ≥ 1; IDESCC(6) ≥ 1;
$0 \leq$ IDESCC(7) $\leq m_p - 1$; $0 \leq$ IDESCC(8) $\leq n_p - 1$;
IDESCC(9) ≥ max(1,numroc(IDESCC(3),IDESCC(5),p_r,IDESCC(7),m_p));
if SIDE = 'L',
    IDESCC(5) = IDESCA(5);
    mod(IA−1,IDESCA(5)) = mod(IC−1,IDESCC(5));
    mod(IDESCA(7)+(IA−1)/IDESCA(5),n_p) =
    mod(IDESCC(7)+(IC−1)/IDESCC(5),n_p);
if SIDE = 'R',
    IDESCC(6) = IDESCA(5); IDESCC(7) = IDESCA(7);
    mod(IA−1,IDESCA(5)) = mod(JC−1,IDESCC(6)).

15: WORK(∗) — DOUBLE PRECISION array *Local Workspace*

Note: the dimension of WORK must be at least max(1,LWORK). The minimum value of LWORK required to successfully call this routine can be obtained by setting LWORK = −1. The required size is returned in array element WORK(1).

On exit: WORK(1) contains the minimum dimension of the array WORK required to successfully complete the task.

16: LWORK — INTEGER *Local Input*

On entry: either −1 (see WORK) or the dimension of the array WORK required to successfully complete the task. If LWORK is set to −1 on entry this routine simply performs some initial error checking and then, if these checks are successful, calculates the minimum size of LWORK required.

Constraints:

either LWORK = −1,
or LWORK ≥ max((...)) + IDESCA(6)×IDESCA(6), where θ is defined differently depending on the value of SIDE:
    if SIDE = 'L', $\theta = d_1 + d_2$;
    if SIDE = 'R', $\theta = d_2$ + max(f_1+numroc(λ,IDESCA(6),0,0,l_{mn}/n_p),d_1); where
        λ = numroc(N+e_2,IDESCA(6),0,0,n_p),
        and l_{mn} is the least common multiple of m_p and n_p, and where
        d_1 = numroc(M+e_1,IDESCC(5),p_r,e_3,m_p);
        d_2 = numroc(N+e_2IDESCC(6),p_c,e_4,n_p);
        e_1 = mod(IC−1,IDESCC(5));
        e_2 = mod(JC−1,IDESCC(6));
        e_3 = indxg2p(IC,IDESCC(5),p_r,IDESCC(7),m_p);
        e_4 = indxg2p(JC,IDESCC(6),p_c,IDESCC(8),n_p);
        f_1 = numroc(N+f_2,IDESCA(5),p_r,f_3,m_p);
        f_2 = mod(IA−1,IDESCA(5));
        f_3 = indxg2p(IA,IDESCA(5),p_r,IDESCA(7), m_p).

17: INFO — INTEGER *Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On exit: INFO = 0 (or −9999 if reduced error checking is enabled) 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 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

6.1 Algorithmic Detail

See Anderson *et al.* [1], Blackford *et al.* [2] and Golub and Van Loan [3].

6.2 Parallelism Detail

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

7 References

[1] 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

[2] Blackford L S, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D and Whaley R C (1997) ScaLAPACK Users' Guide *SIAM* 3600 University City Science Center, Philadelpia, PA 19104-2688, USA. URL: http://www.netlib.org/scalapack/slug/scalapack_slug.html

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

8 Example

See Section 8 of the document for F08AEFP (PDGEQRF).