F08ASFP (PZGEQRF)

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

F08ASFP (PZGEQRF) computes the QR factorization of a complex 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 unitary matrix Q is not formed explicitly but is represented as a product of elementary reflectors

$$Q = H_1 H_2 \ldots H_k, \quad \text{where} \quad k = \min(m, n).$$

Each elementary reflector H_ℓ has the form

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

where τ_ℓ is a complex scalar, and v_ℓ is a complex vector of length m. No pivoting is performed by F08ASFP (PZGEQRF).

2 Specification

```
SUBROUTINE F08ASFP(M, N, A, IA, JA, IDESCA, TAU, WORK, LWORK, INFO)
ENTRY      PZGEQRF(M, N, A, IA, JA, IDESCA, TAU, WORK, LWORK, INFO)
COMPLEX*16          A(*), TAU(*), WORK(*)
INTEGER             M, N, IA, JA, IDESCA(*), LWORK, INFO
```

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.
m_p	–	the number of rows 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:	M, N, IA, JA IDESCA(1), IDESCA(3:8)
Global output arguments:	INFO

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 two-dimensional block distribution. This data distribution is described in more detail in the the F08 Chapter Introduction. The array TAU is distributed across the processor columns in a cyclic two-dimensional block fashion, and is aligned with the rows of the matrix A.

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 F08ASFP (PZGEQRF):

Complex matrix generation:	F01ZVFP
Complex matrix input:	X04BRFP
Complex matrix output:	X04BSFP
Complex matrix gather:	F01WGFP
Complex matrix scatter:	F01WUFP

4 Arguments

1: M — INTEGER *Global Input*

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

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

2: N — INTEGER *Global Input*

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

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

3: A($*$) — COMPLEX*16 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+N$-$1,IDESCA(6),p_c,IDESCA(8),n_p).

On entry: the local part of the m by n matrix 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 unitary 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 unitary 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: i_A, the row index of matrix A that identifies the first row of the matrix A_s to be factorized.

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

5: JA — INTEGER *Global Input*

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

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

6: 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. 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) ≥ 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($*$) — COMPLEX*16 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($*$) — COMPLEX*16 array *Local Workspace/Local Output*

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 the real part of array element WORK(1).

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

9: 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 $\geq$ IDESCA(6) $\times$ ($c_1 + c_2$ + IDESCA(6)) where

c_1 = numroc(M+d_1,IDESCA(5),p_r,d_3,m_p);
c_2 = numroc(N+d_2,IDESCA(6),p_c,d_4,n_p);
d_1 = mod(IA−1,IDESCA(5));
d_2 = mod(JA−1,IDESCA(6));
d_3 = indxg2p(IA,IDESCA(5),p_r,IDESCA(7),m_p);
d_4 = indxg2p(JA,IDESCA(6),p_c,IDESCA(8),n_p).

10: 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 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 $\frac{8}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{8}{3}m^2(3n-m)$ if $m < n$. To solve the system of equations $A_sX = B_s$, this routine may be followed by a call to F08ATFP (PZUNGQR) 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 $RX = Q^HB_s$. If R is triangular then X may be computed using the PBLAS routine PZTRSM. F08AUFP (PZUNMQR) may be used to form Q^HB_s.

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 unitary 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

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

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] and Blackford *et al.* [2] for details of the block algorithm 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, 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

To solve the linear least-squares problem

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

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

$$A = \begin{pmatrix} 0.96-0.81i & -0.03+0.96i & -0.91+2.06i & -0.05+0.41i \\ -0.98+1.98i & -1.20+0.19i & -0.66+0.42i & -0.81+0.56i \\ 0.62-0.46i & 1.01+0.02i & 0.63-0.17i & -1.11+0.60i \\ -0.37+0.38i & 0.19-0.54i & -0.98-0.36i & 0.22-0.20i \\ 0.83+0.51i & 0.20+0.01i & -0.17-0.46i & 1.47+1.59i \\ 1.08-0.28i & 0.20-0.12i & -0.07+1.23i & 0.26+0.26i \end{pmatrix}$$

and

$$C = \begin{pmatrix} -1.54+0.76i & 3.17-2.09i \\ 0.12-1.92i & -6.53+4.18i \\ -9.08-4.31i & 7.28+0.73i \\ 7.49+3.65i & 0.91-3.97i \\ -5.63-2.12i & -5.46-1.64i \\ 2.37+8.03i & -2.84-5.86i \end{pmatrix}.$$

The example uses a 2 by 2 logical processor grid and a 2 by 2 block 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

```
*     F08ASFP Example Program Text
*     NAG Parallel Library Release 3 Revised. NAG Copyright 1999.
*     .. 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, LWORK
      PARAMETER        (MMAX=6,NMAX=4,LDA=MMAX,RHSMAX=2,LDC=MMAX,
     +                 IAROW=0,IACOL=0,ICROW=0,ICCOL=0,LWORK=100)
      COMPLEX*16       ONE
      PARAMETER        (ONE=(1.0D+0,0.0D+0))
*     .. Local Scalars ..
      INTEGER          I, IA, IC, ICNTXT, IFAIL, INFO, J, JA, JC, M, MP,
     +                 N, NP, NRHS, ID, JD
      LOGICAL          ROOT
*     .. Local Arrays ..
      COMPLEX*16       A(LDA,NMAX), C(LDC,RHSMAX), CG(NMAX,RHSMAX),
     +                 TAU(MMAX), WORK(LWORK)
      INTEGER          IDESCA(9), IDESCC(9)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01WGFP, F08ASFP, F08AUFP, PZTRSM, X04BRFP,
     +                 Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F08ASFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
      MP = 2
      NP = 2
      ID = 0
      JD = 0
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      OPEN (NIN,FILE='f08asfpe.d')
*
*     Skip heading in data file
*
      READ (NIN,*)
      READ (NIN,*) M, N
      READ (NIN,*) NRHS
*
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. NRHS.LE.RHSMAX) THEN
*
*        Set the starting address and array descriptor for A
*
         IA = 1
         JA = 1
         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
*
*        Read the matrix A from data file
*
         IFAIL = 0
         CALL X04BRFP(NIN,M,N,A,IA,JA,IDESCA,IFAIL)
*
*         Set the starting address and array descriptor for C
*
         IC = 1
         JC = 1
         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
*
*        Read the matrix C from data file
*
         IFAIL = 0
         CALL X04BRFP(NIN,M,NRHS,C,IC,JC,IDESCC,IFAIL)
*
*        Factorize A
*
         CALL F08ASFP(M,N,A,IA,JA,IDESCA,TAU,WORK,LWORK,INFO)
*
         IF (ROOT) THEN
            WRITE (NOUT,*) 'F08ASFP INFO = ', INFO
            WRITE (NOUT,*)
         END IF
         IF (INFO.NE.0) GO TO 40
*
*        Compute solution
*
*        First apply Q to the right hand sides
*
         CALL F08AUFP('L','C',M,NRHS,N,A,IA,JA,IDESCA,TAU,C,IC,JC,
     +                IDESCC,WORK,LWORK,INFO)
*
         IF (ROOT) THEN
            WRITE (NOUT,*) 'F08AUFP INFO = ', INFO
            WRITE (NOUT,*)
         END IF
         IF (INFO.NE.0) GO TO 40
*
```

```
*        Solve the triangular system
*
         CALL PZTRSM('Left','Upper','No transpose','Non-unit',N,NRHS,
     +               ONE,A,IA,JA,IDESCA,C,IC,JC,IDESCC)
*
*        Gather the solution matrix C to the (root) processor (0,0)
*        where it is denoted by CG
*
         IFAIL = 0
         CALL F01WGFP(N,NRHS,C,IC,JC,IDESCC,ID,JD,CG,N,WORK,LWORK,IFAIL)
*
*        Print the matrix CG
*
         IF (ROOT) THEN
            WRITE (NOUT,*) 'The least-squares solution'
            WRITE (NOUT,*)
            DO 20 I = 1, N
               WRITE (NOUT,'(1X,2(''('',F8.4,1X,'','',F8.4,'')'',2X))')
     +           (CG(I,J),J=1,NRHS)
   20       CONTINUE
         END IF
*
      END IF
*
   40 CONTINUE
      CLOSE (NIN)
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
```

8.2 Example Data

```
F08ASFP Example Program Data
 6 4                                                         :Values of M and N
 2                                                           :Value of NRHS
 ( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
 (-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
 ( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
 ( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)   :End of matrix A
 (-1.54, 0.76) ( 3.17,-2.09)
 ( 0.12,-1.92) (-6.53, 4.18)
 (-9.08,-4.31) ( 7.28, 0.73)
 ( 7.49, 3.65) ( 0.91,-3.97)
 (-5.63,-2.12) (-5.46,-1.64)
 ( 2.37, 8.03) (-2.84,-5.86)                                 :End of matrix C
```

8.3 Example Results

```
F08ASFP Example Program Results

F08ASFP INFO =            0

F08AUFP INFO =            0

The least-squares solution

( -0.4936 , -1.1993)  (  0.7535 ,  1.4404)
( -2.4708 ,  2.8373)  (  5.1726 , -3.6235)
(  1.5060 , -2.1830)  ( -2.6609 ,  2.1334)
(  0.4459 ,  2.6848)  ( -2.6966 ,  0.2711)
```