

# F01ZQFP

## 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

F01ZQFP generates and distributes an  $m$  by  $n$  real matrix  $A_s$  on a logical grid of processors in a cyclic 2-d block distribution;  $A_s$  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  $i = j = 1$ ,  $m = m_A$  and  $n = n_A$ , then  $A_s = A$ .

This routine distributes matrices in the form required by a number of the routines in Chapters F07 and F08. A user-supplied subroutine is required to generate a block of the matrix  $A_s$ .

### 2 Specification

```
SUBROUTINE F01ZQFP(GMAT, M, N, A, IA, JA, IDESCA, IFAIL)
DOUBLE PRECISION  A(*)
INTEGER           M, N, IA, JA, IDESCA(9), IFAIL
EXTERNAL         GMAT
```

### 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 $X$ .
$\text{numroc}(\alpha, b_\ell, q, s, k)$	–	a function which gives the <b>number of rows or columns</b> of a distributed matrix owned by the processor with the row or column coordinate $q$ ( $p_r$ or $p_c$ ), where $\alpha$ is the total number of rows or columns of the matrix, $b_\ell$ 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.

#### 3.2 Global and Local Arguments

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

#### 3.3 Distribution Strategy

The matrix  $A$  will be partitioned into  $M_b^A$  by  $N_b^A$  rectangular blocks and stored in an array A in a cyclic 2-d block distribution. This data distribution is described in more detail in the F07 and F08 Chapter Introductions.

## 4 Arguments

- 1: GMAT — SUBROUTINE, supplied by the user. *Global External Procedure*  
 GMAT must return the block  $A_s(i_1 : i_2, j_1 : j_2)$  of the matrix to be distributed, in the array AL.  
 Its specification is:

	SUBROUTINE	GMAT(I1, I2, J1, J2, AL, LDAL)	
	DOUBLE PRECISION	AL(LDAL,*)	
	INTEGER	I1, I2, J1, J2, LDAL	
1:	I1 — INTEGER		<i>Local Input</i>
	<i>On entry:</i>	$i_1$ , the first row of the block of $A_s$ to be generated.	
2:	I2 — INTEGER		<i>Local Input</i>
	<i>On entry:</i>	$i_2$ , the last row of the block of $A_s$ to be generated.	
3:	J1 — INTEGER		<i>Local Input</i>
	<i>On entry:</i>	$j_1$ , the first column of the block of $A_s$ to be generated.	
4:	J2 — INTEGER		<i>Local Input</i>
	<i>On entry:</i>	$j_2$ , the last column of the block of $A_s$ to be generated.	
5:	AL(LDAL,*) — DOUBLE PRECISION array		<i>Local Output</i>
	<i>On exit:</i>	AL must contain the block $A_s(i_1 : i_2, j_1 : j_2)$ of the matrix $A_s$ in its first $(I2-I1+1)$ rows and $(J2-J1+1)$ columns.	
6:	LDAL — INTEGER		<i>Local Input</i>
	<i>On entry:</i>	the first dimension of the array AL.	

GMAT must be declared as EXTERNAL in the (sub)program from which F01ZQFP is called. Arguments denoted as *Input* must **not** be changed by this procedure.

- 2: M — INTEGER *Global Input*  
*On entry:* the number of rows of the matrix  $A_s$ ,  $m$ .  
*Constraint:*  $0 \leq M \leq \text{IDESCA}(3)$ .
- 3: N — INTEGER *Global Input*  
*On entry:* the number of columns of the matrix  $A_s$ ,  $n$ .  
*Constraint:*  $0 \leq N \leq \text{IDESCA}(4)$ .
- 4: A(\*) — DOUBLE PRECISION array *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  $(\text{IDESCA}(9), \gamma)$ , where  $\gamma \geq \text{numroc}(\text{JA} + \text{N} - 1, \text{IDESCA}(6), p_c, \text{IDESCA}(8), n_p)$ . (See the Example Program.)  
*On exit:* the local parts of the matrix  $A_s$ , distributed in a cyclic 2-d block fashion.
- 5: IA — INTEGER *Global Input*  
*On entry:* the row index of A,  $i_A$ , that identifies the first row of the submatrix  $A_s$ .  
*Constraint:*  $1 \leq \text{IA} \leq \text{IDESCA}(3) - M + 1$ .
- 6: JA — INTEGER *Global Input*  
*On entry:* the column index of A,  $j_A$ , that identifies the first column of the submatrix  $A_s$ .  
*Constraint:*  $1 \leq \text{JA} \leq \text{IDESCA}(4) - N + 1$ .

**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)  $\geq$  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), $m_p$ )).

**8: IFAIL — INTEGER** *Global Input/Global Output*

*On entry:* IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;  
 IFAIL = -1, if multigridding is employed.

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

## 5 Errors and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output from the root processor (or processor {0,0} when the root processor is not available) on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = -2000

The routine has been called with a value of ICNTXT (stored in IDESCA(2)) which was not returned by a call to Z01AAFP on one or more processors.

IFAIL = -1000

The utility routine Z01AAFP has not been called to define the logical processor grid and initialise the internal variables used by the Library.

IFAIL < 0

On entry, one of the arguments was invalid:

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

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

This routine may be used to distribute the data in the form required by a number of the routines in Chapters F07 and F08. This use is illustrated in the Example Program in Section 8. This routine may also be used to distribute a vector, and this is also illustrated by the Example Program in Section 8.

### 6.1 Algorithmic Detail

The routine generates the real matrix on a logical processor by a cyclic 2-d block distribution.

### 6.2 Parallelism Detail

The routine generates the matrix on each logical processor independently.

## 7 References

- [1] Dongarra J J and Whaley R C (1995) A users' guide to the BLACS v1.0. *LAPACK Working Note 94 (Technical Report CS-95-281)* Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA.  
URL: <http://www.netlib.org/lapack/lawns/lawn94.ps>

## 8 Example

To solve the equations

$$Ax = b,$$

where the matrix  $A$  and the vector  $b$  are given by

$$a_{ij} = \max(i, j), \quad b_i = i.$$

The exact solution is the vector  $e_1$ , the first column of the unit matrix. The ScaLAPACK routines F07ADFP (PDGETRF) and F07AEFP (PDGETRS) (see Chapter F07) are called to solve the equations following the distribution of  $A$  and  $b$ , and routine X04DBFP (see Chapter X04) is used to print the solution vector  $x$ . The example illustrates the solution for five equations, with a 2 by 2 logical grid of processors and a 2 by 2 block.

### 8.1 Example Text

```
*      F01ZQFP Example Program Text
*      NAG Parallel Library Release 2. NAG Copyright 1996.
*      .. Parameters ..
INTEGER          NOUT
PARAMETER       (NOUT=6)
INTEGER          N
PARAMETER       (N=5)
INTEGER          DT
PARAMETER       (DT=1)
INTEGER          NMAX
PARAMETER       (NMAX=100)
INTEGER          MG, NG
PARAMETER       (MG=2, NG=2)
INTEGER          MB, NB
PARAMETER       (MB=2, NB=MB)
INTEGER          LDA, TDA
PARAMETER       (LDA=NMAX/MG+MB, TDA=NMAX/NG+NB)
INTEGER          LDB
PARAMETER       (LDB=LDA)
INTEGER          LIPIV, LWORK
```

```

PARAMETER      (LIPIV=LDA+NB,LWORK=NMAX)
*
* .. Local Scalars ..
INTEGER        IA, IB, ICNTXT, IFAIL, INFO, JA, JB, NCOLS, NROWS
LOGICAL        ROOT
*
* .. Local Arrays ..
DOUBLE PRECISION A(LDA,TDA), B(LDB), WORK(LWORK)
INTEGER        IDESCA(9), IDESCB(9), IPIV(LIPIV)
*
* .. External Functions ..
LOGICAL        Z01ACFP
EXTERNAL       Z01ACFP
*
* .. External Subroutines ..
EXTERNAL       F01ZQFP, F07ADFP, F07AEFP, GMATA, GRHS, X04BDFP,
+             Z01AAFP, Z01ABFP
*
* .. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) WRITE (NOUT,*) 'F01ZQFP Example Program Results'
*
NROWS = MG
NCOLS = NG
*
IFAIL = 0
CALL Z01AAFP(ICNTXT,NROWS,NCOLS,IFAIL)
*
*
* Set the array descriptors of A and the right hand side
*
IA = 1
JA = 1
IDESCA(1) = DT
IDESCA(2) = ICNTXT
IDESCA(3) = NMAX
IDESCA(4) = NMAX
IDESCA(5) = MB
IDESCA(6) = NB
IDESCA(7) = 0
IDESCA(8) = 0
IDESCA(9) = LDA
IB = 1
JB = 1
IDESCB(1) = DT
IDESCB(2) = ICNTXT
IDESCB(3) = NMAX
IDESCB(4) = NMAX
IDESCB(5) = MB
IDESCB(6) = NB
IDESCB(7) = 0
IDESCB(8) = 0
IDESCB(9) = LDB
*
IF (IA+N-1.LE.NMAX .AND. JA+N-1.LE.NMAX) THEN
  IFAIL = 0
*
*   Distribute the matrix A
*
  CALL F01ZQFP(GMATA,N,N,A,IA,JA,IDESCA,IFAIL)
*
*   Distribute the right hand side
*

```

```

      CALL F01ZQFP(GRHS,N,1,B,IB,JB,IDESCB,IFAIL)
*
      CALL F07ADFP(N,N,A,IA,JA,IDESCA,IPIV,INFO)
*
      IF (INFO.EQ.0) THEN
*
          CALL F07AEFP('No transpose',N,1,A,IA,JA,IDESCA,IPIV,B,IB,JB,
+              IDESCB,INFO)
*
          IF (INFO.EQ.0) THEN
*
              CALL X04BDFP(NOUT,N,1,B,IB,JB,IDESCB,'(1X,F7.3)',WORK,
+              IFAIL)
*
              ELSE
                  IF (ROOT) WRITE (NOUT,*)
+              'Unable to solve triangular system'
                  END IF
              ELSE
                  IF (ROOT) WRITE (NOUT,*) 'Matrix is singular'
                  END IF
              END IF
          END IF
*
          IFAIL = 0
          CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
          STOP
          END
*
      SUBROUTINE GMATA(I1,I2,J1,J2,AL,LDAL)
*
      GMATA generates the block A(I1: I2, J1: J2) of the matrix A such
*
      that
*
          a(i,j) = max(i,j)
*
      in the array AL.
*
      .. Scalar Arguments ..
      INTEGER          I1, I2, J1, J2, LDAL
*
      .. Array Arguments ..
      DOUBLE PRECISION AL(LDAL,*)
*
      .. Local Scalars ..
      INTEGER          I, J, K, L
*
      .. Intrinsic Functions ..
      INTRINSIC        MAX
*
      .. Executable Statements ..
      L = 1
      DO 40 J = J1, J2
          K = 1
          DO 20 I = I1, I2
              AL(K,L) = MAX(I,J)
              K = K + 1
          20  CONTINUE
              L = L + 1
      40  CONTINUE
*
      RETURN
*

```

```
      END
*
*      SUBROUTINE GRHS(I1,I2,J1,J2,BL,LDBL)
*      GRHS generates the block B(I1: I2) of the right hand side
*      vector B such that
*
*          b(i) = i
*
*      in the array BL.
*
*      .. Scalar Arguments ..
      INTEGER          I1, I2, J1, J2, LDBL
*      .. Array Arguments ..
      DOUBLE PRECISION BL(LDBL)
*      .. Local Scalars ..
      INTEGER          I, K
*      .. Executable Statements ..
      K = 1
      DO 20 I = I1, I2
          BL(K) = I
          K = K + 1
      20 CONTINUE
*
*      RETURN
*
*      END
```

## 8.2 Example Data

None.

## 8.3 Example Results

F01ZQFP Example Program Results

```
1.000
0.000
0.000
0.000
0.000
```

---