F01ZVFP

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

F01ZVFP generates a distributed m by n complex matrix A_s on a logical grid of processors in a cyclic two-dimensional 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 $i_A = j_A = 1$, $m = m_A$ and $n = n_A$, then $A_s = A$.

This routine generates 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 F01ZVFP(GMAT, M, N, A, IA, JA, IDESCA, IFAIL)
COMPLEX*16         A(*)
INTEGER            M, N, IA, JA, IDESCA(*), IFAIL
EXTERNAL           GMAT
```

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.

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), IFAIL

Global output arguments: IFAIL

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 two-dimensional block distribution. This data distribution is described in more detail in the the F07 Chapter Introduction and the F08 Chapter Introduction.

4 Arguments

1: GMAT — SUBROUTINE, supplied by the user. *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)
COMPLEX*16          AL(LDAL,*)
INTEGER             I1, I2, J1, J2, LDAL
```

1: I1 — INTEGER *Local Input*

On entry: i_1, the first row of the block of A_s to be generated.

2: I2 — INTEGER *Local Input*

On entry: i_2, the last row of the block of A_s to be generated.

3: J1 — INTEGER *Local Input*

On entry: j_1, the first column of the block of A_s to be generated.

4: J2 — INTEGER *Local Input*

On entry: j_2, the last column of the block of A_s to be generated.

5: AL(LDAL,*) — COMPLEX*16 array *Local Output*

On exit: AL must contain the block $A_s(i_1 : i_2, j_1 : j_2)$ of the matrix A_s in its first $(i_2 - i_1 + 1)$ rows and $(j_2 - j_1 + 1)$ columns.

6: LDAL — INTEGER *Local Input*

On entry: the size of the first dimension of the array AL as declared in the (sub)program from which F01ZVFP is called.

GMAT must be declared as EXTERNAL in the (sub)program from which F01ZVFP is called. Arguments denoted as *Input* must **not** be changed by this procedure.

2: M — INTEGER *Global Input*

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

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

3: N — INTEGER *Global Input*

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

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

4: A(∗) — COMPLEX*16 array *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). (See Section 8.)

On exit: the local parts of the matrix A_s, distributed in a cyclic two-dimensional block fashion.

5: IA — INTEGER *Global Input*

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

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

6: JA — INTEGER *Global Input*

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

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

7: 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) $\geq$ 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*

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 entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this argument (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 (or −9999 if reduced error checking is enabled) 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).

5.1 Full Error Checking Mode Only

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

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

This routine may be used to distribute the data in the form required by a number of the routines in Chapter F07 and Chapter F08. This routine may also be used to distribute a vector.

6.1 Algorithmic Detail

The routine generates the matrix on a logical processor by column block.

6.2 Parallelism Detail

The routine generates the matrix on each logical processor independently.

7 References

[1] 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

8 Example

To solve the system of equations

$$Ax = b,$$

where the matrix A and the vector b are given by

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

The exact solution is the vector e_1, (the first column of the unit matrix) with all complex components equal to zero.

The ScaLAPACK routines F07ARFP (PZGETRF) and F07ASFP (PZGETRS) (see Chapter F07) are called to solve the equations following the distribution of A and b, and routine X04BSFP (see Chapter X04) is used to print the solution vector x. The example illustrates the solution of a system of equations with five unknowns. A 2 by 2 Library Grid is used with 2 by 2 blocking.

8.1 Example Text

```
*     F01ZVFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     CHARACTER*4      FORMAT
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          DT
      PARAMETER        (DT=1)
      INTEGER          N
```

```
      PARAMETER        (N=5)
      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
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      COMPLEX*16       A(LDA,TDA), B(LDB), WORK(LWORK)
      INTEGER          IDESCA(9), IDESCB(9), IPIV(LIPIV)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01ZVFP, F07ARFP, F07ASFP, GMATA, GRHS, X04BSFP,
     +                 Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F01ZVFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
      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 F01ZVFP(GMATA,N,N,A,IA,JA,IDESCA,IFAIL)
*
*        Distribute the right hand side
*
         CALL F01ZVFP(GRHS,N,1,B,IB,JB,IDESCB,IFAIL)
*
         CALL F07ARFP(N,N,A,IA,JA,IDESCA,IPIV,INFO)
*
         IF (INFO.EQ.0) THEN
*
            CALL F07ASFP('No transpose',N,1,A,IA,JA,IDESCA,IPIV,B,IB,JB,
     +                   IDESCB,INFO)
*
            IF (INFO.EQ.0) THEN
*
               FORMAT = '(4(:,'' ('',F7.3,'','',F7.3,'')''))'
               CALL X04BSFP(NOUT,N,1,B,IB,JB,IDESCB,FORMAT,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
*
      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), max(i,j))
*
*     in the array AL.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, J1, J2, LDAL
*     .. Array Arguments ..
      COMPLEX*16       AL(LDAL,*)
*     .. Local Scalars ..
```

```
      INTEGER          I, J, K, L
*     .. Intrinsic Functions ..
      INTRINSIC        CMPLX, MAX
*     .. Executable Statements ..
      L = 1
      DO 40 J = J1, J2
         K = 1
         DO 20 I = I1, I2
            AL(K,L) = CMPLX(MAX(I,J),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, i)
*
*     in the array BL.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, J1, J2, LDBL
*     .. Array Arguments ..
      COMPLEX*16       BL(LDBL)
*     .. Local Scalars ..
      INTEGER          I, K
*     .. Intrinsic Functions ..
      INTRINSIC        CMPLX
*     .. Executable Statements ..
      K = 1
      DO 20 I = I1, I2
         BL(K) = CMPLX(I,I)
         K = K + 1
   20 CONTINUE
*
      RETURN
*
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F01ZVFP Example Program Results

(  1.000,  0.000)
(  0.000,  0.000)
(  0.000,  0.000)
(  0.000,  0.000)
(  0.000,  0.000)
```