F01WUFP

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

F01WUFP distributes an m by n complex matrix A_s available in its natural form on a (source) processor to the processors on the Library Grid in the cyclic two-dimensional block format. The distributed version of B may be regarded as a submatrix of a larger distributed matrix A, i.e.,

$$B(1:m,1:n) \equiv A(i_A:i_A+m-1, j_A:j_A+n-1).$$

Note: if $i=j=1$, $m=m_A$ and $n=n_A$, then $B=A$.

This routine is useful for distributing matrices in a form required by (ScaLAPACK) routines in Chapters F07 and F08.

2 Specification

```
      SUBROUTINE F01WUFP(M, N, A, IA, JA, IDESCA, IS, JS, B, LDB, WORK,
     1                   LWORK, IFAIL)
      COMPLEX*16         A(*), B(LDB,*), WORK(*)
      INTEGER            M, N, IA, JA, IDESCA(*), IS, JS, LDB, LWORK,
     1                   IFAIL
```

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.
i_s	–	the row grid coordinate of the source processor.
j_s	–	the column grid coordinate of the source 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, IS, JS, IDESCA(1), IDESCA(3:8), IFAIL

Global output arguments: IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

On exit, the matrix A will be nominally partitioned into M_b^A by N_b^A rectangular blocks and stored in local arrays A in a cyclic two-dimensional block distribution. However, only the elements of the submatrix A_s are referenced by this routine; the other elements of the matrix A are untouched. This data distribution is described in more detail in the F07 and F08 Chapter Introductions.

4 Arguments

1: M — INTEGER *Global Input*

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

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

2: N — INTEGER *Global Input*

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

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

3: 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).

On exit: the relevant parts of the distributed matrix A.

4: IA — INTEGER *Global Input*

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

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

5: JA — INTEGER *Global Input*

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

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

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) $\geq$ 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)).

7: IS — INTEGER *Global Input*

8: JS — INTEGER *Global Input*

On entry: $\{i_s, j_s\}$, the coordinate of the (source) processor from which B is distributed.

Constraints:

$0 \leq i_s \leq m_p - 1$;
$0 \leq j_s \leq n_p - 1$.

9: B(LDB,*) — COMPLEX*16 array *Local Input*

Note: the size of the second dimension of the array B must be at least max(1,N).

On entry: matrix B to be distributed. This array is only referenced on the source processor as defined by the processor coordinate $\{i_s, j_s\}$.

10: LDB — INTEGER *Local Input*

Note: B and LDB are referenced only by the processor which has the coordinate $\{i_s, j_s\}$.

On entry: the size of the first dimension of the array B as declared in the (sub)program from which F01WUFP is called.

Constraints:

LDB $\geq$ max(1,M) on the source processor;
LDB $\geq$ 1 otherwise.

11: WORK($*$) — COMPLEX*16 array *Local Workspace/Global Output*

Note: the dimension of the array WORK must be at least max(3,LWORK). WORK is used as a workspace only by the (source) processor which has the coordinate $\{i_s, j_s\}$.

On exit: WORK(i) = l_i, $i = 1, 2, 3$. See LWORK for the definitions of l_i.

12: LWORK — INTEGER *Local Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F01WUFP is called. The minimum requirement for LWORK is max(4,min(l_1,l_2)), but the higher value max(4,l_3) is recommended for higher efficiency where

$$
\begin{aligned}
l_1 &= \max_{i=0,\ldots,m_p-1} [\alpha_1(i) - \alpha_2(i)] \\
\alpha_1(i) &= \text{numroc(M+IA}-1\text{,IDESCA(5)},i\text{,IDESCA(7)},m_p) \\
\alpha_2(i) &= \text{numroc(IA}-1\text{,IDESCA(5)},i\text{,IDESCA(7)},m_p) \\
l_2 &= \max_{j=0,\ldots,n_p-1} [\beta_1(j) - \beta_2(j)] \\
\beta_1(j) &= \text{numroc(N+JA}-1\text{,IDESCA(6)},j\text{,IDESCA(8)},n_p) \\
\beta_2(j) &= \text{numroc(JA}-1\text{,IDESCA(6)},j\text{,IDESCA(8)},n_p) \\
l_3 &= \max_{i=0,\ldots,m_p-1} \max_{j=0,\ldots,n_p-1} [\alpha_1(i) - \alpha_2(i)][\beta_1(j) - \beta_2(j)]
\end{aligned}
$$

Note: if LWORK = -1, then a workspace query for LWORK is assumed; the routine only calculates the required minimum sizes of the array WORK as defined by l_1, l_2 and l_3. These values are returned in the real parts of the first three entries of the array WORK.

Constraint: LWORK $\geq$ max[4,min(l_1,l_2)] or LWORK = -1.

13: 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

6.1 Algorithmic Detail

The performance of the algorithm depends upon the size of LWORK. The critical values of LWORK are l_i, $i = 1, 2, 3$. See LWORK for the definitions of l_i. For higher efficiency, LWORK should be set to max(l_3,4) (or greater). However, this routine will work with a workspace size of max(4,min(l_1,l_2)). Note that $l_3 \geq$ max(l_1,l_2).

6.2 Parallelism Detail

The source processor sequentially distributes B to other processors.

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

The example program illustrates the distribution of a matrix A_s.

8.1 Example Text

```
*     F01WUFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          M, N
      PARAMETER        (M=10,N=3)
      INTEGER          NB
      PARAMETER        (NB=3)
      INTEGER          DT, NA
      PARAMETER        (DT=1,NA=25)
      INTEGER          LDA, TDA, LWORK
      PARAMETER        (LDA=NA,TDA=NA,LWORK=LDA)
*     .. Local Scalars ..
      INTEGER          I, I1, I2, I3, IA, ICNTXT, IFAIL, IS, J, JA, JS,
     +                 MP, NP
      LOGICAL          ROOT
*     .. Local Arrays ..
      COMPLEX*16       A(LDA,TDA), B(LDA,TDA), C(LDA,TDA), WORK(LWORK)
      INTEGER          IDESCA(9)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01WGFP, F01WUFP, Z01AAFP, Z01ABFP
*     .. Intrinsic Functions ..
      INTRINSIC        CMPLX, DBLE, NINT
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F01WUFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
      MP = 2
      NP = 2
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Generate a matrix on the root processor
*
      IF (ROOT) THEN
         DO 20 J = 1, N
            DO 20 I = 1, M
               B(I,J) = CMPLX(DBLE(I),DBLE(J))
   20    CONTINUE
      END IF
*
*     Set up the indices of the first row and column and the descriptor
*     for distributed matrix
*
      IA = 1
```

```
      JA = 1
      IDESCA(1) = DT
      IDESCA(2) = ICNTXT
      IDESCA(3) = NA
      IDESCA(4) = NA
      IDESCA(5) = NB
      IDESCA(6) = NB
      IDESCA(7) = 1
      IDESCA(8) = 1
      IDESCA(9) = LDA
*
*     Distribute the 3rd column of the matrix from the root processor
*
      IFAIL = 0
      IS = 0
      JS = 0
      CALL F01WUFP(M,1,A,IA,JA,IDESCA,IS,JS,B(1,3),LDA,WORK,LWORK,IFAIL)
*
*     Gather this column of the matrix back to the root processor as
*     the 3rd column of the matrix C, and print the column
*
      CALL F01WGFP(M,1,A,IA,JA,IDESCA,IS,JS,C(1,3),LDA,WORK,LWORK,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The third column of the matrix",/)')
         DO 40 I = 1, M
            WRITE (NOUT,'(1X,( "(",F4.1,1X,",",F4.1,")" ))') C(I,3)
   40    CONTINUE
         WRITE (NOUT,*)
      END IF
*
*     Distribute the 2nd row of the matrix from the root processor
*
      IFAIL = 0
      IS = 0
      JS = 0
      CALL F01WUFP(1,N,A,IA,JA,IDESCA,IS,JS,B(2,1),LDA,WORK,LWORK,IFAIL)
*
*     Gather this row of the matrix back to the root processor as the
*     2nd row of the matrix C, and print the row
*
      CALL F01WGFP(1,N,A,IA,JA,IDESCA,IS,JS,C(2,1),LDA,WORK,LWORK,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The second row of the matrix",/)')
         WRITE (NOUT,'(1X,3( "(",F4.1,1X,",",F4.1,")",3X ))')
     +      (C(2,J),J=1,N)
         WRITE (NOUT,*)
      END IF
*
*     Distribute the whole matrix from the root processor
*
      IFAIL = 0
      IS = 0
      JS = 0
      CALL F01WUFP(M,N,A,IA,JA,IDESCA,IS,JS,B,LDA,WORK,LWORK,IFAIL)
*
*     Store the values of l(1), l(2) and l(3) in I1, I2 and I3
```

```
*
      I1 = NINT(DBLE(WORK(1)))
      I2 = NINT(DBLE(WORK(2)))
      I3 = NINT(DBLE(WORK(3)))
*
*     Gather the matrix back to the root processor as the matrix C, and
*     print the matrix
*
      CALL F01WGFP(M,N,A,IA,JA,IDESCA,IS,JS,C,LDA,WORK,LWORK,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The matrix",/)')
         DO 60 I = 1, M
            WRITE (NOUT,'(1X,3( "(",F4.1,1X,",",F4.1,")",3X ))')
     +        (C(I,J),J=1,N)
   60    CONTINUE
         WRITE (NOUT,*)
      END IF
*
*     Print the values l(1), l(2) and l(3) that determine the
*     recommended dimension of WORK
*
      IF (ROOT) THEN
         WRITE (NOUT,
     +      '(1X,"The values of l(1), l(2) and l(3) are:",/)')
         WRITE (NOUT,'(1X,"Real part of WORK(1) = ",I3)') I1
         WRITE (NOUT,'(1X,"Real part of WORK(2) = ",I3)') I2
         WRITE (NOUT,'(1X,"Real part of WORK(3) = ",I3)') I3
      END IF
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F01WUFP Example Program Results

The third column of the matrix

( 1.0 , 3.0)
( 2.0 , 3.0)
( 3.0 , 3.0)
( 4.0 , 3.0)
( 5.0 , 3.0)
( 6.0 , 3.0)
( 7.0 , 3.0)
( 8.0 , 3.0)
( 9.0 , 3.0)
(10.0 , 3.0)
```

```
The second row of the matrix

( 2.0 , 1.0)   ( 2.0 , 2.0)   ( 2.0 , 3.0)

The matrix

( 1.0 , 1.0)   ( 1.0 , 2.0)   ( 1.0 , 3.0)
( 2.0 , 1.0)   ( 2.0 , 2.0)   ( 2.0 , 3.0)
( 3.0 , 1.0)   ( 3.0 , 2.0)   ( 3.0 , 3.0)
( 4.0 , 1.0)   ( 4.0 , 2.0)   ( 4.0 , 3.0)
( 5.0 , 1.0)   ( 5.0 , 2.0)   ( 5.0 , 3.0)
( 6.0 , 1.0)   ( 6.0 , 2.0)   ( 6.0 , 3.0)
( 7.0 , 1.0)   ( 7.0 , 2.0)   ( 7.0 , 3.0)
( 8.0 , 1.0)   ( 8.0 , 2.0)   ( 8.0 , 3.0)
( 9.0 , 1.0)   ( 9.0 , 2.0)   ( 9.0 , 3.0)
(10.0 , 1.0)   (10.0 , 2.0)   (10.0 , 3.0)

The values of l(1), l(2) and l(3) are:

Real part of WORK(1) =   6
Real part of WORK(2) =   3
Real part of WORK(3) =  18
```