F01WAFP

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

F01WAFP gathers an m by n real distributed matrix A_s to a user specified logical processor (the destination processor) and stores it in the natural (non-distributed) format. The matrix A_s can be considered as a submatrix of a larger m_A by n_A distributed 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 $A_s=A$.

It is assumed that the matrix A has been distributed on a logical grid of processors in the cyclic two-dimensional block format. However, only the elements of the submatrix A_s are referenced by this routine.

It is also possible to gather copies of the matrix A_s on processors which are either on a particular row or column of the processor grid. Alternatively, all processors on the Library Grid can receive a copy of A_s.

This routine is useful for gathering full or partial solutions which have been computed using (ScaLAPACK) routines in the F07 and F08 Chapter Introductions.

2 Specification

```
 SUBROUTINE F01WAFP(M, N, A, IA, JA, IDESCA, ID, JD, B, LDB, WORK,
1                   LWORK, IFAIL)
 DOUBLE PRECISION   A(*), B(LDB,*), WORK(*)
 INTEGER            M, N, IA, JA, IDESCA(*), ID, JD, 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_d	–	the row grid coordinate of the destination processor.
j_d	–	the column grid coordinate of the destination 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, ID, JD, IDESCA(1), IDESCA(3:8), IFAIL

Global output arguments: IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

It is assumed that the matrix A has been partitioned into M_b^A by N_b^A rectangular blocks and stored in local arrays A in a cyclic two-dimensional block distribution. That is, the matrix A or at least its submatrix A_s is already available as a distributed matrix on the Library Grid. 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 A_s.

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

2: N — INTEGER *Global Input*

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

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

3: A($*$) — DOUBLE PRECISION array *Local Input*

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 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: $1 \leq$ IA $\leq$ IDESCA(3) $-$ 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: $1 \leq$ 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) $\geq$ 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: ID — INTEGER *Global Input*

8: JD — INTEGER *Global Input*

On entry: $\{i_d, j_d\}$, the coordinate of the (destination) processor(s) which will gather A_s.

If ID = −1, then all processors in column j_d of the grid will receive a copy of A_s.
If JD = −1, then all processors in row i_d of the grid will receive a copy of A_s.
If ID = JD = −1 then all processors of the grid will receive a copy of A_s.

Constraints:

$0 \leq$ ID $\leq m_p - 1$ or ID = −1;
$0 \leq$ JD $\leq n_p - 1$ or JD = −1.

9: B(LDB,*) — DOUBLE PRECISION array *Local Output*

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

On exit: a copy of the matrix A_s on the destination processor(s) as defined by the processor coordinate {ID,JD}.

10: LDB — INTEGER *Local Input*

On entry: the size of the first dimension of the array B as declared in the (sub)program from which F01WAFP is called.

Constraint:

LDB $\geq$ max(1,m); on destination processor(s).
LDB $\geq$ 1 otherwise.

11: WORK($*$) — DOUBLE PRECISION array *Local Workspace/Global Output*

Note: the dimension of the array WORK must be at least max(3,LWORK). WORK is used as workspace only by the processor which has the coordinate {ID,JD} where ID $\neq -1$ and JD $\neq -1$. If ID = −1 and JD $\neq -1$ then the coordinate of the processor where WORK is referenced is {0,JD}. The second coordinate JD is interpreted in a similar way. If ID = JD = −1 then the coordinate of the processor where WORK is referenced is $\{0, 0\}$..

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 F01WAFP 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

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

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 first three elements 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 destination processor sequentially gathers the relevant parts of A_s from 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 gathering of a matrix A_s that has been generated using routine F01ZQFP.

8.1 Example Text

```
*     F01WAFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          M, N
      PARAMETER        (M=10,N=5)
      INTEGER          DT, NB
      PARAMETER        (DT=1,NB=3)
      INTEGER          NA
      PARAMETER        (NA=25)
      INTEGER          LDA, TDA, LWORK
      PARAMETER        (LDA=NA,TDA=NA,LWORK=LDA)
*     .. Local Scalars ..
      INTEGER          I, IA, IAA, ICNTXT, IFAIL, IR, J, JA, JAA, JR,
     +                 MP, NP
      LOGICAL          ROOT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,TDA), B(LDA,TDA), WORK(LWORK)
      INTEGER          IDESCA(9)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01WAFP, F01ZQFP, GMATA, Z01AAFP, Z01ABFP
*     .. Intrinsic Functions ..
      INTRINSIC        NINT
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F01WAFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
      MP = 2
      NP = 2
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
```

```
*     Set up and generate an m by n matrix starting at row index 1 and
*     column index 1
*
      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
      CALL F01ZQFP(GMATA,M,N,A,IA,JA,IDESCA,IFAIL)
*
*     Gather the element (3,4) of the matrix to processor (0,0) and
*     print the element on the root processor
*
      IFAIL = 0
      IR = 0
      JR = 0
      IAA = (IA-1) + 3
      JAA = (JA-1) + 4
      CALL F01WAFP(1,1,A,IAA,JAA,IDESCA,IR,JR,B,LDA,WORK,LWORK,IFAIL)
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The (3,4) element of the matrix",/)')
         WRITE (NOUT,'(1X,F8.4)') B(1,1)
         WRITE (NOUT,*)
      END IF
*
*     Gather the 3rd column of the matrix to processor row 0 and print
*     the column on the root processor
*
      IFAIL = 0
      IR = 0
      JR = -1
      IAA = IA
      JAA = (JA-1) + 3
      CALL F01WAFP(M,1,A,IAA,JAA,IDESCA,IR,JR,B,LDA,WORK,LWORK,IFAIL)
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The third column of the matrix",/)')
         DO 20 I = 1, M
            WRITE (NOUT,'(1X,F8.4)') B(I,1)
   20    CONTINUE
         WRITE (NOUT,*)
      END IF
*
*     Gather the 2nd row of the matrix to processor column 0 and print
*     the row on the root processor
*
      IFAIL = 0
      IR = -1
      JR = 0
      IAA = (IA-1) + 2
      JAA = JA
      CALL F01WAFP(1,N,A,IAA,JAA,IDESCA,IR,JR,B,LDA,WORK,LWORK,IFAIL)
      IF (ROOT) THEN
```

```
            WRITE (NOUT,'(1X,"The second row of the matrix",/)')
            WRITE (NOUT,'(1X,5(F8.4,2X))') (B(1,J),J=1,N)
            WRITE (NOUT,*)
         END IF
*
*     Gather the matrix to all processors on the grid and print the
*     matrix on the root processor
*
      IFAIL = 0
      IR = -1
      JR = -1
      CALL F01WAFP(M,N,A,IA,JA,IDESCA,IR,JR,B,LDA,WORK,LWORK,IFAIL)
      IF (ROOT) THEN
         WRITE (NOUT,'(1X,"The matrix",/)')
         DO 40 I = 1, M
            WRITE (NOUT,'(1X,5(F8.4,2X))') (B(I,J),J=1,N)
   40    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,"WORK(1) = ",I3)') NINT(WORK(1))
         WRITE (NOUT,'(1X,"WORK(2) = ",I3)') NINT(WORK(2))
         WRITE (NOUT,'(1X,"WORK(3) = ",I3)') NINT(WORK(3))
      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) = dble(i) + dble(j)/10000.0
*
*     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        DBLE
*     .. Executable Statements ..
      L = 1
      DO 40 J = J1, J2
         K = 1
         DO 20 I = I1, I2
```

```
            AL(K,L) = DBLE(I) + DBLE(J)/10000.0D0
            K = K + 1
   20    CONTINUE
         L = L + 1
   40 CONTINUE
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F01WAFP Example Program Results

The (3,4) element of the matrix

  3.0004

The third column of the matrix

  1.0003
  2.0003
  3.0003
  4.0003
  5.0003
  6.0003
  7.0003
  8.0003
  9.0003
 10.0003

The second row of the matrix

  2.0001    2.0002    2.0003    2.0004    2.0005

The matrix

  1.0001    1.0002    1.0003    1.0004    1.0005
  2.0001    2.0002    2.0003    2.0004    2.0005
  3.0001    3.0002    3.0003    3.0004    3.0005
  4.0001    4.0002    4.0003    4.0004    4.0005
  5.0001    5.0002    5.0003    5.0004    5.0005
  6.0001    6.0002    6.0003    6.0004    6.0005
  7.0001    7.0002    7.0003    7.0004    7.0005
  8.0001    8.0002    8.0003    8.0004    8.0005
  9.0001    9.0002    9.0003    9.0004    9.0005
 10.0001   10.0002   10.0003   10.0004   10.0005

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

WORK(1) =   6
WORK(2) =   3
WORK(3) =  18
```