F02WRFP

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

F02WRFP computes the Singular Value Decomposition (SVD) of a complex matrix whose columns are distributed on a logical 2-d processor grid.

The SVD of an m by n complex rectangular matrix A (where $m \geq n$) may be defined in the form

$$A = \left[U|\tilde{U}\right] \left[\frac{\Sigma}{0}\right] V^H = U\Sigma V^H$$

where U is an $m \times n$ matrix of left singular vectors, $\tilde{U}$ is an $m \times (m-n)$ matrix, V is an $n \times n$ unitary matrix of right singular vectors, and Σ is an $n \times n$ diagonal matrix of singular values. The singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$ are non-negative and in non-increasing order of magnitude. The matrix $\left[U|\tilde{U}\right]$ is unitary.

For the case, $m < n$ the SVD may be defined in the form

$$A = U\left[\Sigma|0\right]\left[V|\tilde{V}\right]^H = U\Sigma V^H$$

where U is an $m \times m$ unitary matrix, V is an $n \times m$ matrix of right singular vectors, $\tilde{V}$ is an $n \times (n-m)$ matrix, and Σ is an $m \times m$ diagonal matrix of singular values. The matrix $\left[V|\tilde{V}\right]$ is unitary.

In general, matrices $\tilde{V}$ and $\tilde{U}$ are not unique. For convenience in the description of the routine, the definition of singular values is extended when $m < n$ by defining $\sigma_{m+1} = \cdots = \sigma_n = 0$. In that case, the columns of $\tilde{V}$ may be considered as right singular vectors corresponding to zero singular values.

F02WRFP computes only the left singular vectors U which correspond to non-zero singular values. Optionally all right singular vectors V (including $\tilde{V}$ if $m < n$) are computed.

2 Specification

```
SUBROUTINE F02WRFP(ICNTXT, M, N, A, LDA, VWANT, NX, NR, IFAIL)
COMPLEX*16         A(0:LDA-1,0:*)
INTEGER            ICNTXT, M, N, LDA, NX, NR, IFAIL
LOGICAL            VWANT
```

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	–	$m_p \times n_p$, the total number of processors in the logical processor grid.
p_d	–	the number of logical processors which hold columns of the matrix A.
N_b	–	the maximum number of columns of the matrix A held locally on a logical processor.
N_x	–	the actual number of columns of the matrix A held locally on a logical processor, where $0 \leq N_x \leq N_b$.
n_r	–	the number of non-zero singular values computed by a logical processor, where $0 \leq n_r \leq N_x$.
$\lceil x \rceil$	–	the ceiling function of x which gives the smallest integer which is not less than x.

3.2 Global and Local Arguments

The input arguments M, N, VWANT and IFAIL are global and so must have the same value on entry to the routine on each processor. The output argument IFAIL is global and so will have the same value on exit from the routine on each processor. The remaining arguments are local.

3.3 Distribution Strategy

Columns of the matrix A are allocated to logical processors on the 2-d grid row by row (i.e., in row major ordering of the grid) starting from the $\{0,0\}$ logical processor. Each logical processor that contains columns of the matrix contains $N_b = \lceil n/p \rceil$ columns, except the last processor that actually contains data, for which the number of columns held may be less than N_b. This processor will contain $\mathrm{mod}(n, N_b)$ columns if $\mathrm{mod}(n, N_b) \neq 0$, and will contain N_b columns otherwise. Some logical processors may not contain any columns of the matrix if n is not large relative to p, but if $n > (p-1)^2$ then all processors will certainly contain columns of the matrix.

The number of logical processors that contain columns of the matrix is given by $p_d = \lceil n/N_b \rceil$.

The following example illustrates a case where the last processor with data is not the last processor of the grid. Furthermore the number of columns on the last processor with data is not equal to the number of columns on other processors.

If $m_p = 2$, $n_p = 4$ then $p = m_p \times n_p = 8$. If $n = 41$ then $N_b = \lceil n/p \rceil = \lceil 5.125 \rceil = 6$, $\mathrm{mod}(n, N_b) = 5 \neq 0$ and $p_d = \lceil n/N_b \rceil = \lceil 6.833 \rceil = 7$.

processor {0,0} $N_x = 6$ columns (1:6)	processor {0,1} $N_x = 6$ columns (7:12)	processor {0,2} $N_x = 6$ columns (13:18)	processor {0,3} $N_x = 6$ columns (19:24)
processor {1,0} $N_x = 6$ columns (25:30)	processor {1,1} $N_x = 6$ columns (31:36)	processor {1,2} $N_x = 5$ columns (37:41)	processor {1,3} $N_x = 0$

If the data is distributed incorrectly, the routine may fail to produce correct results or will exit with an error flag. Routines to assist with distribution of data can be found in Chapters F01 and X04.

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the BLACS context used by the communication mechanism, usually returned by a call to Z01AAFP.

2: M — INTEGER *Global Input*

On entry: m, the number of rows of A.

Constraint: M ≥ 0.

3: N — INTEGER *Global Input*

On entry: n, the number of columns of A.

Constraint: N ≥ 0.

4: A(0:LDA−1,0:∗) — COMPLEX*16 array *Local Input/Local Output*

Note: the size of the second dimension of the array A must be at least $N_x + 1$ where N_x is the number of columns of A held locally by the logical processor. The array A is not referenced if $N_x = 0$.

On entry: A$(1 : m, 1 : N_x)$ must contain columns of the matrix A, as defined by the distribution strategy (see Section 3.3).

On exit: The real parts of A$(0, 1 : N_x)$ contain N_x singular values of the matrix A stored on this logical processor. They are ordered locally and globally (in the row major ordering of the processors) in non-increasing order of magnitude.

A$(1 : m, 1 : n_r)$ contains the left singular vectors corresponding to non-zero singular values. A left singular vector is not computed if the corresponding singular value and the corresponding elements of A are set to zero.

If VWANT = .TRUE., then A$(m+1 : m+n, 1 : N_x)$ contains the right singular vectors corresponding to the singular values held on this logical processor.

The remainder of the array is used as workspace and contains no useful information.

5: LDA — INTEGER *Local Input*

On entry: the size of the first dimension of the array A as declared in the (sub)program from which F02WRFP is called.

Constraint: LDA $\geq$ M + N + 2 if VWANT is .TRUE.; otherwise LDA $\geq$ M + 2.

6: VWANT — LOGICAL *Global Input*

On entry: VWANT must be set to .TRUE. if right singular vectors are also required.

7: NX — INTEGER *Local Output*

On exit: N_x, the actual number of columns of the matrix A held on the logical processor.

8: NR — INTEGER *Local Output*

On exit: n_r, the number of non-zero singular values held on the logical processor.

9: 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 an invalid value of ICNTXT on one or more processors.

IFAIL = -1000

The logical processor grid and library mechanism (Library Grid) have not been correctly defined, see Z01AAFP.

IFAIL = $-i$

On entry, the ith argument had an invalid value. 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.

IFAIL = 1

The Jacobi algorithm has not converged.

6 Further Comments

6.1 Algorithmic Detail

The algorithm is based on a one-sided Jacobi method, see Hestenes [2].

6.2 Parallelism Detail

The algorithm uses a linear array of logical processors. This linear array is mapped to the 2-d array based on the row major ordering beginning from the $\{0,0\}$ logical processor on the 2-d array. Most of the communication is between neighbours on the linear array of processors.

6.3 Accuracy

The computed factors U, Σ and V satisfy the relation

$$U\Sigma V^H = A + E,$$

where

$$\|E\| \leq c\epsilon\|A\|,$$

ϵ being the ***machine precision***, c is a modest function of m and n and $\|.\|$ denotes the 2-norm.

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

[2] Hestenes M R (1958) Inversion of matrices by biorthogonalization and related results *J. SIAM* **6** 51–90

8 Example

To find the singular value decomposition of the 4 by 7 complex matrix A given by

$$A = \left(\begin{array}{cc|cc|cc|c} 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i \\ 0.0 & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i \\ 0.0 & 0.0 & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i \\ 0.0 & 0.0 & 0.0 & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i & 1.0+1.0i \end{array}\right)$$

and to print results on the root processor. The routine F01ZWFP is used to generate the matrix A on a 2 by 2 logical processor grid. The number of columns of the matrix A on each logical processor, N_x, is equal to 2 on logical processors $\{0,0\}$, $\{0,1\}$, and $\{1,0\}$. On the final logical processor $\{1,1\}$, $N_x = 1$.

The routine X04BFFP is used to bring the singular values to the root processor and print them. The left and right singular vectors are printed using the routine X04BUFP.

8.1 Example Text

```
*     F02WRFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          M, N, MM, NM
      PARAMETER        (M=4,N=7,MM=20,NM=20)
      INTEGER          MG, NG
      PARAMETER        (MG=2,NG=2)
      INTEGER          LDA, TDA, LDD
      PARAMETER        (LDA=MM+NM+2,TDA=(NM/(MG*NG)+2),LDD=2)
      CHARACTER*20     FORMT
      PARAMETER        (FORMT='F8.4')
      LOGICAL          VWANT
      PARAMETER        (VWANT=.TRUE.)
```

```
*     .. Local Scalars ..
      INTEGER          I, ICNTXT, ICOFF, IFAIL, MP, NP, NR, NX
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
*     .. Local Arrays ..
      COMPLEX*16       A(0:LDA-1,0:TDA-1), W(LDA,TDA)
      DOUBLE PRECISION D(0:1,TDA-1)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01ZWFP, F02WRFP, GMATA, X04BFFP, X04BUFP,
     +                 Z01AAFP, Z01ABFP
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      ROOT = Z01ACFP()
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F02WRFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
*     Define the 2D processor grid
*
      MP = MG
      NP = NG
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Generate the matrix A
*
      IFAIL = 0
*
      CALL F01ZWFP(ICNTXT,GMATA,M,N,A(1,1),LDA,NX,IFAIL)
*
*     Compute the SVD
*
      IFAIL = 0
*
      CALL F02WRFP(ICNTXT,M,N,A,LDA,VWANT,NX,NR,IFAIL)
*
*     Print singular values
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Singular values'
         WRITE (NOUT,*)
         TITOP = 'N'
         CNUMOP = 'G'
      END IF
      ICOFF = 0
      IFAIL = 0
*
      DO 20 I = 1, NX
         D(0,I) = DBLE(A(0,I))
   20 CONTINUE
*
```

```
      CALL X04BFFP(ICNTXT,NOUT,1,NR,D(0,1),LDD,FORMT,TITOP,CNUMOP,ICOFF,
     +             D(1,1),LDD,IFAIL)
*
*     Print left singular vectors
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Left singular vectors'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BUFP(ICNTXT,NOUT,M,NR,A(1,1),LDA,FORMT,TITOP,CNUMOP,ICOFF,
     +             W,LDA,IFAIL)
*
*     Print right singular vectors (full set)
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Right singular vectors (full set)'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BUFP(ICNTXT,NOUT,N,NX,A(M+1,1),LDA,FORMT,TITOP,CNUMOP,
     +             ICOFF,W,LDA,IFAIL)
*
*     Undefine the grid
*
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GMATA(M,J1,J2,AL,LDAL)
*
*  GMATA generates the block A( 1: M, J1: J2 ) of the matrix A such
*  that
*
*     a(i,j) = 0.0 if i $>$ j
*     a(i,j) = cmplx(1.0, 1.0) else
*
*  in the array AL.
*
*     .. Scalar Arguments ..
      INTEGER          J1, J2, LDAL, M
*     .. Array Arguments ..
      COMPLEX*16       AL(LDAL,*)
*     .. Local Scalars ..
      INTEGER          I, J, L
*     .. Intrinsic Functions ..
      INTRINSIC        DCMPLX
*     .. Executable Statements ..
      L = 1
      DO 40 J = J1, J2
         DO 20 I = 1, M
            IF (J.GE.I) THEN
               AL(I,L) = DCMPLX(1.0D0,1.0D0)
            ELSE
               AL(I,L) = 0.0D0
```

```
            END IF
   20     CONTINUE
          L = L + 1
   40  CONTINUE
*
*     End of GMATA.
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F02WRFP Example Program Results

Singular values

        1       2
   6.2912  1.6947

        3       4
   0.9833  0.7631

Left singular vectors

                     1                     2
  ( -0.3997, -0.3997) ( -0.4345, -0.4345)

  ( -0.3795, -0.3795) ( -0.1319, -0.1319)

  ( -0.3402, -0.3402) (  0.2625,  0.2625)

  ( -0.2836, -0.2836) (  0.4742,  0.4742)

                     3                     4
  ( -0.3417, -0.3417) ( -0.1861, -0.1861)

  (  0.3651,  0.3651) (  0.4530,  0.4530)

  (  0.3167,  0.3167) ( -0.4638, -0.4638)

  ( -0.3868, -0.3868) (  0.2123,  0.2123)

Right singular vectors (full set)

                     1                     2
  ( -0.1271,  0.0000) ( -0.5128,  0.0000)

  ( -0.2477,  0.0000) ( -0.6685,  0.0000)

  ( -0.3559,  0.0000) ( -0.3587,  0.0000)

  ( -0.4460,  0.0000) (  0.2009,  0.0000)

  ( -0.4460,  0.0000) (  0.2009,  0.0000)
```

```
( -0.4460,  0.0000) (  0.2009,  0.0000)

( -0.4460,  0.0000) (  0.2009,  0.0000)

                  3                   4
( -0.6950,  0.0000) ( -0.4877,  0.0000)

(  0.0477,  0.0000) (  0.6996,  0.0000)

(  0.6917,  0.0000) ( -0.5159,  0.0000)

( -0.0951,  0.0000) (  0.0405,  0.0000)

( -0.0951,  0.0000) (  0.0405,  0.0000)

( -0.0951,  0.0000) (  0.0405,  0.0000)

( -0.0951,  0.0000) (  0.0405,  0.0000)

                  5                   6
(  0.0000,  0.0000) (  0.0000,  0.0000)

(  0.0000,  0.0000) (  0.0000,  0.0000)

(  0.0000,  0.0000) (  0.0000,  0.0000)

(  0.8573,  0.0000) (  0.0000,  0.0000)

( -0.3435,  0.0000) ( -0.7071,  0.0000)

( -0.3435,  0.0000) (  0.7071,  0.0000)

( -0.1703,  0.0000) (  0.0000,  0.0000)

                  7
(  0.0000,  0.0000)

(  0.0000,  0.0000)

(  0.0000,  0.0000)

(  0.1225,  0.0000)

(  0.3633,  0.0000)

(  0.3633,  0.0000)

( -0.8491,  0.0000)
```