X04BUFP

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

X04BUFP outputs a set of complex matrices distributed on a two-dimensional logical processor grid. Output is to units connected to the root processor.

Each matrix on a logical processor is nominally denoted by A. Each matrix A and the row dimension m and the column dimension n are local to each logical processor. In general, each m and each n may be different on different logical processors in the Library Grid. This routine does not assume that the local matrices A are part of any particular global data structure. It prints the local matrices in the row-major ordering of the grid.

In general, each matrix A can be printed on a different output unit NOUT defined on the logical processor which owns the matrix A but in practice NOUT will generally have the same value on all logical processors and the matrices will then be written to a single file.

Optionally a standard title which identifies the logical processor can be printed followed by a blank line before a matrix is printed. As an example, the title

```
Array from logical processor {5,3}
```

can be printed before the matrix on logical processor {5,3}. The matrices are output with a **maximum record length of 80**. Options are also available to print column numbers of the matrices.

If the root processor is not present on the current Library Grid then the logical processor {0,0} of the current two-dimensional grid takes over the role of the root processor.

2 Specification

```
 SUBROUTINE X04BUFP(ICNTXT, NOUT, M, N, A, LDA, FORMAT, TITOP,
1                   CNUMOP, ICOFF, W, LDW, IFAIL)
 COMPLEX*16         A(LDA,*), W(LDW,*)
 INTEGER            ICNTXT, NOUT, M, N, LDA, ICOFF, LDW, IFAIL
 CHARACTER*1        TITOP, CNUMOP
 CHARACTER*(*)      FORMAT
```

3 Usage

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m	–	the number of rows of a matrix on a logical processor.
n	–	the number of columns of a matrix on a logical processor.
$m_{\max}$	–	the maximum value of m taken over all logical processors (which have a matrix to be printed) except the root processor.
$n_{\max}$	–	the maximum value of n taken over all logical processors (which have a matrix to be printed) except the root processor.

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: IFAIL

Global output arguments: IFAIL

Note: the workspace argument W and the input arguments FORMAT, TITOP, and CNUMOP are referenced only on the root processor.

3.3 Distribution Strategy

All complex matrices are defined locally on each logical processor. The values of m and n (which define the dimensions of the matrices) do not have to be identical on every logical processor.

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

Note: the value of ICNTXT **must not** be changed.

2: NOUT — INTEGER *Local Input*

On entry: unit number for output. If NOUT is non-positive, no output of this matrix (including the standard title which specifies the processor) is done.

3: M — INTEGER *Local Input*

On entry: m, the number of rows of the local matrix. If m is non-positive, no output of this matrix (including the standard title which specifies the processor) is done.

4: N — INTEGER *Local Input*

On entry: n, the number of columns of the local matrix. If n is non-positive, no output of this matrix (including the standard title which specifies the processor) is done.

5: A(LDA,$*$) — COMPLEX*16 array *Local Input*

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

On entry: the local matrix A.

6: LDA — INTEGER *Local Input*

On entry: the size of the first dimension of the array A as declared in the (sub)program from which X04BUFP is called.

Constraint: LDA $\geq$ M if M $>$ 0, and N $>$ 0 and NOUT $>$ 0; otherwise LDA $\geq$ 1.

7: FORMAT — CHARACTER*(*) *Local Input*

On entry: a valid Fortran format code on the root processor. This may be any format code allowed on the root processor, whether it is standard Fortran or not. FORMAT is used to output elements of the matrices. It may or may not be enclosed in brackets. Examples of valid values for FORMAT are `'(F11.4)'`, `'1PE13.5'`, `'G14.5'`.

In addition, there is a special code which forces X04BUFP to choose its own format code:

> FORMAT = '*' means that X04BUFP will choose a format code such that numbers will be printed to as many significant digits as are necessary to distinguish between neighbouring machine numbers. Thus any two numbers that are stored with different internal representations should look different on output. Whether they do in fact look different will depend on the run-time library of the Fortran compiler in use.

FORMAT is not referenced on other logical processors.

Constraint: The character length of FORMAT must not be greater than 80.

8: TITOP — CHARACTER*1 *Local Input*

On entry: if TITOP = 'Y', then the standard title which identifies the logical processor is printed before the corresponding matrix is printed. Otherwise the standard title is not printed.

TITOP is referenced only by the root processor.

9: CNUMOP — CHARACTER*1 *Local Input*

On entry: indicates the type of labelling to be applied to columns of the matrices.

If CNUMOP = 'L' then the numbering of columns is specific to the matrix being printed. The columns are numbered starting at ICOFF + 1 and ending at ICOFF + n for each matrix.

If CNUMOP = 'G' then the columns are numbered consecutively in the order they are printed. The columns are numbered starting at 1 and ending at n_{total} where n_{total} is the sum of all (positive) values of n.

If CNUMOP $\neq$ 'L' or 'G' then the columns are not numbered.

CNUMOP is referenced only by the root processor.

10: ICOFF — INTEGER *Local Input*

On entry: offset value for column numbering if CNUMOP = 'L'.

11: W(LDW,$*$) — COMPLEX*16 array *Local Workspace*

Note: the size of the second dimension of the array W must be at least max(1,$n_{\max}$). This array is referenced only on the root processor.

12: LDW — INTEGER *Local Input*

On entry: the size of the first dimension of the array W as declared in the (sub)program from which X04BUFP is called.

Constraint: on the root processor LDW must be at least max(1,$m_{\max}$).

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 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 was invalid. 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.

5.2 Any Error Checking Mode

IFAIL = 1

On entry, variable FORMAT is more than 80 characters long.

IFAIL = 2

The code supplied in FORMAT cannot be used to output a number. FORMAT probably has too wide a field width or contains an illegal edit descriptor.

6 Further Comments

6.1 Algorithmic Detail

None.

6.2 Parallelism Detail

Each matrix is brought to the root processor and is then printed. This is performed sequentially using the row-major ordering of the grid.

7 References

None.

8 Example

To find the eigenvalues and eigenvectors of a 14 by 14 Hermitian matrix A using the routine F02FRFP, and to print the eigenvalues and the eigenvectors on the root processor. The element a_{ij} of the matrix A is taken as $a_{ij} = i + j + 1$ if $i = j$, $a_{ij} = i + j\sqrt{-1}$ if $i > j$ and $a_{i,j} = i - j\sqrt{-1}$ if $i < j$. The routine F01ZRFP is used to generate the matrix A on a 2×2 logical processor grid. The number of columns of the matrix A on each logical processor, N_x, is equal to 4 on logical processors {0,0}, {0,1}, and {1,0}. On the final logical processor {1,1}, $N_x = 2$. The number of eigenvectors on each logical processor is given by N_x.

The routines X04BFFP and X04BUFP are used to print the eigenvalues and eigenvectors, respectively. All rsults are printed to standard output.

See also Section 8 of the document for F02FRFP for a simple example of use.

8.1 Example Text

```
*     X04BUFP Example Program Text
*     NAG Parallel Library Release 3 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=14)
      INTEGER          MG, NG
      PARAMETER        (MG=2,NG=2)
      INTEGER          LDA, TDA, LDD
      PARAMETER        (LDA=N+N+2,TDA=(N/(MG*NG))+2,LDD=2)
      CHARACTER*20     FORMT1, FORMT2
```

```
      PARAMETER        (FORMT1='F7.3',FORMT2='F12.4')
*     .. Local Scalars ..
      INTEGER          ICNTXT, ICOFF, IFAIL, J, MP, NP, NX
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
*     .. Local Arrays ..
      COMPLEX*16       A(0:LDA-1,0:TDA-1)
      DOUBLE PRECISION D(0:1,0:TDA-1)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         F01ZWFP, F02FRFP, GMATA, X04BFFP, X04BUFP,
     +                 Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X04BUFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
*     Define the 2D processor grid
*
      MP = MG
      NP = NG
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      IFAIL = 0
*
*     Generation of the matrix A
*
      CALL F01ZWFP(ICNTXT,GMATA,N,N,A(1,1),LDA,NX,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Matrix'
         WRITE (NOUT,*)
         TITOP = 'Y'
         CNUMOP = 'G'
      END IF
      ICOFF = 0
      IFAIL = 0
*
      CALL X04BUFP(ICNTXT,NOUT,N,NX,A(1,1),LDA,FORMT1,TITOP,CNUMOP,
     +             ICOFF,A(N+1,1),LDA,IFAIL)
*
      IFAIL = 0
*
*     Solution of the Hermitian eigenvalue problem
*
      CALL F02FRFP(ICNTXT,N,A,LDA,NX,IFAIL)
*
*     Print the eigenvalues on the root processor
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Eigenvalues'
         WRITE (NOUT,*)
```

```
         TITOP = 'N'
      END IF
      ICOFF = 0
      IFAIL = 0
      DO 20 J = 1, NX
         D(0,J) = A(0,J)
   20 CONTINUE
*
      CALL X04BFFP(ICNTXT,NOUT,1,NX,D(0,1),LDD,FORMT2,TITOP,CNUMOP,
     +             ICOFF,D(1,1),LDD,IFAIL)
*
*     Print the eigenvectors on the root processor
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Eigenvectors of the matrix'
         WRITE (NOUT,*)
         CNUMOP = 'G'
      END IF
      IFAIL = 0
*
      CALL X04BUFP(ICNTXT,NOUT,N,NX,A(1,1),LDA,FORMT1,TITOP,CNUMOP,
     +             ICOFF,A(N+1,1),LDA,IFAIL)
*
*     Undefine the 2D processor grid
*
      IFAIL = 0
*
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GMATA(M,J1,JL,AL,LDAL)
*
*     GMATA generates the block A( 1: M, J1: JL ) of the
*     matrix A such that
*
*     a(i,j) = i+j+1  if i.eq.j
*     a(i,j) = cmplx(i, j)  if i.gt.j
*     a(i,j) = cmplx(i,-j)  if i.lt.j
*
*     in the array AL.
*
*     .. Scalar Arguments ..
      INTEGER          J1, JL, 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, JL
         DO 20 I = 1, M
            IF (I.GT.J) THEN
               AL(I,L) = DCMPLX(I,J)
            ELSE IF (I.LT.J) THEN
```

```
               AL(I,L) = DCMPLX(I,-J)
            ELSE
               AL(I,L) = I + J + 1
            END IF
   20    CONTINUE
         L = L + 1
   40 CONTINUE
*
*     End of GMATA.
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
 X04BUFP Example Program Results

 Matrix

   Array from logical processor    0,   0

                  1                 2                 3                 4
 (  3.000,  0.000) (  1.000, -2.000) (  1.000, -3.000) (  1.000, -4.000)
 (  2.000,  1.000) (  5.000,  0.000) (  2.000, -3.000) (  2.000, -4.000)
 (  3.000,  1.000) (  3.000,  2.000) (  7.000,  0.000) (  3.000, -4.000)
 (  4.000,  1.000) (  4.000,  2.000) (  4.000,  3.000) (  9.000,  0.000)
 (  5.000,  1.000) (  5.000,  2.000) (  5.000,  3.000) (  5.000,  4.000)
 (  6.000,  1.000) (  6.000,  2.000) (  6.000,  3.000) (  6.000,  4.000)
 (  7.000,  1.000) (  7.000,  2.000) (  7.000,  3.000) (  7.000,  4.000)
 (  8.000,  1.000) (  8.000,  2.000) (  8.000,  3.000) (  8.000,  4.000)
 (  9.000,  1.000) (  9.000,  2.000) (  9.000,  3.000) (  9.000,  4.000)
 ( 10.000,  1.000) ( 10.000,  2.000) ( 10.000,  3.000) ( 10.000,  4.000)
 ( 11.000,  1.000) ( 11.000,  2.000) ( 11.000,  3.000) ( 11.000,  4.000)
 ( 12.000,  1.000) ( 12.000,  2.000) ( 12.000,  3.000) ( 12.000,  4.000)
 ( 13.000,  1.000) ( 13.000,  2.000) ( 13.000,  3.000) ( 13.000,  4.000)
 ( 14.000,  1.000) ( 14.000,  2.000) ( 14.000,  3.000) ( 14.000,  4.000)

   Array from logical processor    0,   1

                  5                 6                 7                 8
 (  1.000, -5.000) (  1.000, -6.000) (  1.000, -7.000) (  1.000, -8.000)
 (  2.000, -5.000) (  2.000, -6.000) (  2.000, -7.000) (  2.000, -8.000)
 (  3.000, -5.000) (  3.000, -6.000) (  3.000, -7.000) (  3.000, -8.000)
 (  4.000, -5.000) (  4.000, -6.000) (  4.000, -7.000) (  4.000, -8.000)
 ( 11.000,  0.000) (  5.000, -6.000) (  5.000, -7.000) (  5.000, -8.000)
 (  6.000,  5.000) ( 13.000,  0.000) (  6.000, -7.000) (  6.000, -8.000)
 (  7.000,  5.000) (  7.000,  6.000) ( 15.000,  0.000) (  7.000, -8.000)
 (  8.000,  5.000) (  8.000,  6.000) (  8.000,  7.000) ( 17.000,  0.000)
 (  9.000,  5.000) (  9.000,  6.000) (  9.000,  7.000) (  9.000,  8.000)
 ( 10.000,  5.000) ( 10.000,  6.000) ( 10.000,  7.000) ( 10.000,  8.000)
 ( 11.000,  5.000) ( 11.000,  6.000) ( 11.000,  7.000) ( 11.000,  8.000)
 ( 12.000,  5.000) ( 12.000,  6.000) ( 12.000,  7.000) ( 12.000,  8.000)
 ( 13.000,  5.000) ( 13.000,  6.000) ( 13.000,  7.000) ( 13.000,  8.000)
 ( 14.000,  5.000) ( 14.000,  6.000) ( 14.000,  7.000) ( 14.000,  8.000)
```

```
 Array from logical processor    1,   0

                9                10                11                12
(  1.000, -9.000) (  1.000,-10.000) (  1.000,-11.000) (  1.000,-12.000)
(  2.000, -9.000) (  2.000,-10.000) (  2.000,-11.000) (  2.000,-12.000)
(  3.000, -9.000) (  3.000,-10.000) (  3.000,-11.000) (  3.000,-12.000)
(  4.000, -9.000) (  4.000,-10.000) (  4.000,-11.000) (  4.000,-12.000)
(  5.000, -9.000) (  5.000,-10.000) (  5.000,-11.000) (  5.000,-12.000)
(  6.000, -9.000) (  6.000,-10.000) (  6.000,-11.000) (  6.000,-12.000)
(  7.000, -9.000) (  7.000,-10.000) (  7.000,-11.000) (  7.000,-12.000)
(  8.000, -9.000) (  8.000,-10.000) (  8.000,-11.000) (  8.000,-12.000)
( 19.000,  0.000) (  9.000,-10.000) (  9.000,-11.000) (  9.000,-12.000)
( 10.000,  9.000) ( 21.000,  0.000) ( 10.000,-11.000) ( 10.000,-12.000)
( 11.000,  9.000) ( 11.000, 10.000) ( 23.000,  0.000) ( 11.000,-12.000)
( 12.000,  9.000) ( 12.000, 10.000) ( 12.000, 11.000) ( 25.000,  0.000)
( 13.000,  9.000) ( 13.000, 10.000) ( 13.000, 11.000) ( 13.000, 12.000)
( 14.000,  9.000) ( 14.000, 10.000) ( 14.000, 11.000) ( 14.000, 12.000)

 Array from logical processor    1,   1

               13                14
(  1.000,-13.000) (  1.000,-14.000)
(  2.000,-13.000) (  2.000,-14.000)
(  3.000,-13.000) (  3.000,-14.000)
(  4.000,-13.000) (  4.000,-14.000)
(  5.000,-13.000) (  5.000,-14.000)
(  6.000,-13.000) (  6.000,-14.000)
(  7.000,-13.000) (  7.000,-14.000)
(  8.000,-13.000) (  8.000,-14.000)
(  9.000,-13.000) (  9.000,-14.000)
( 10.000,-13.000) ( 10.000,-14.000)
( 11.000,-13.000) ( 11.000,-14.000)
( 12.000,-13.000) ( 12.000,-14.000)
( 27.000,  0.000) ( 13.000,-14.000)
( 14.000, 13.000) ( 29.000,  0.000)

Eigenvalues

          1          2          3          4
   -49.7708   -11.1399    -2.3971     1.2580

          5          6          7          8
     2.8328     4.1983     5.8893     7.8568

          9         10         11         12
    10.0688    13.2859    17.0928    23.0333

         13         14
    35.8288   160.5275

Eigenvectors of the matrix

                1                2                3                4
(  0.301,  0.092) (  0.311,  0.074) ( -0.056,  0.174) (  0.409,  0.000)
(  0.315,  0.068) (  0.293,  0.036) (  0.533,  0.000) ( -0.192, -0.216)
(  0.318,  0.037) (  0.181, -0.005) ( -0.038, -0.362) ( -0.219,  0.237)
(  0.304,  0.000) ( -0.038, -0.099) ( -0.227,  0.002) (  0.205,  0.222)
(  0.272, -0.039) ( -0.176, -0.275) ( -0.077,  0.098) (  0.190, -0.185)
```

```
(  0.217, -0.079) ( -0.053, -0.182) ( -0.055,  0.115) ( -0.199, -0.177)
(  0.138, -0.116) ( -0.201,  0.235) (  0.080,  0.175) ( -0.198,  0.195)
(  0.035, -0.148) ( -0.248,  0.096) (  0.224, -0.026) (  0.162,  0.208)
( -0.081, -0.171) (  0.192,  0.322) (  0.003, -0.240) (  0.189, -0.137)
( -0.193, -0.182) (  0.059,  0.114) ( -0.261, -0.029) ( -0.135, -0.177)
( -0.275, -0.173) (  0.375,  0.000) ( -0.075,  0.267) ( -0.183,  0.129)
( -0.300, -0.130) ( -0.021, -0.277) (  0.238,  0.116) (  0.104,  0.184)
( -0.253, -0.037) ( -0.039, -0.078) (  0.128, -0.201) (  0.166, -0.087)
( -0.145,  0.111) ( -0.245, -0.119) ( -0.177, -0.132) ( -0.086, -0.153)

               5                6                7                8
(  0.578,  0.000) ( -0.164,  0.280) ( -0.075, -0.193) ( -0.122,  0.076)
( -0.333,  0.006) ( -0.290, -0.240) (  0.180, -0.180) ( -0.148, -0.061)
(  0.208, -0.063) (  0.458,  0.000) (  0.224,  0.332) (  0.057, -0.275)
( -0.292, -0.177) ( -0.304, -0.116) ( -0.360,  0.019) (  0.389,  0.000)
( -0.173,  0.342) (  0.219,  0.138) (  0.370,  0.000) ( -0.171,  0.239)
(  0.267,  0.155) ( -0.048, -0.295) ( -0.270, -0.024) (  0.052, -0.366)
(  0.093, -0.173) ( -0.212,  0.156) (  0.246,  0.136) ( -0.030,  0.294)
( -0.110, -0.036) (  0.207,  0.100) ( -0.095, -0.250) (  0.033, -0.264)
( -0.007,  0.052) (  0.018, -0.195) ( -0.132,  0.219) ( -0.137,  0.263)
( -0.018, -0.026) ( -0.134,  0.052) (  0.202, -0.004) (  0.261, -0.085)
( -0.079,  0.068) (  0.119,  0.071) ( -0.121, -0.110) ( -0.179, -0.056)
(  0.083,  0.121) (  0.029, -0.159) ( -0.018,  0.186) (  0.145,  0.142)
(  0.137, -0.090) ( -0.164,  0.009) (  0.184, -0.058) (  0.033, -0.228)
( -0.107, -0.150) (  0.053,  0.159) ( -0.124, -0.144) ( -0.206,  0.088)

               9               10               11               12
(  0.124,  0.064) (  0.129,  0.073) ( -0.081,  0.069) ( -0.001, -0.089)
(  0.028,  0.049) (  0.139,  0.107) ( -0.108,  0.064) (  0.014, -0.102)
( -0.142, -0.066) (  0.085,  0.162) ( -0.140,  0.015) (  0.056, -0.101)
( -0.161, -0.316) ( -0.082,  0.162) ( -0.135, -0.091) (  0.119, -0.066)
(  0.155, -0.251) ( -0.259, -0.041) ( -0.023, -0.216) (  0.177,  0.027)
( -0.023,  0.286) ( -0.101, -0.324) (  0.202, -0.208) (  0.172,  0.174)
( -0.315, -0.087) (  0.234, -0.061) (  0.299,  0.057) (  0.039,  0.299)
(  0.300,  0.150) ( -0.177,  0.304) ( -0.035,  0.251) ( -0.186,  0.242)
( -0.174,  0.067) ( -0.212, -0.227) ( -0.297, -0.141) ( -0.257, -0.064)
(  0.384,  0.000) (  0.259,  0.208) (  0.186, -0.291) (  0.057, -0.296)
( -0.161, -0.194) ( -0.267,  0.002) (  0.077,  0.301) (  0.366,  0.000)
(  0.126,  0.050) (  0.338,  0.000) ( -0.307, -0.147) ( -0.011,  0.339)
( -0.160, -0.235) ( -0.254, -0.056) (  0.330,  0.000) ( -0.344, -0.137)
( -0.142,  0.271) (  0.233,  0.001) ( -0.259,  0.152) (  0.237, -0.264)

              13               14
(  0.057, -0.055) (  0.114, -0.127)
(  0.067, -0.049) (  0.103, -0.141)
(  0.076, -0.021) (  0.097, -0.156)
(  0.076,  0.032) (  0.097, -0.171)
(  0.056,  0.106) (  0.103, -0.185)
(  0.003,  0.190) (  0.115, -0.196)
( -0.091,  0.257) (  0.134, -0.204)
( -0.216,  0.263) (  0.159, -0.206)
( -0.322,  0.166) (  0.190, -0.201)
( -0.320, -0.032) (  0.227, -0.187)
( -0.132, -0.233) (  0.267, -0.161)
(  0.196, -0.257) (  0.309, -0.122)
(  0.413,  0.000) (  0.349, -0.069)
(  0.226,  0.320) (  0.384,  0.000)
```