C06FUFP

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

C06FUFP computes the two-dimensional Discrete Fourier Transform (DFT) of a bivariate sequence of complex data values $z_{j_1 j_2}$, where $j_1 = 0, 1, \ldots, m-1$, $j_2 = 0, 1, \ldots, n-1$.

The DFT is here defined by:

$$\hat{z}_{k_1 k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} z_{j_1 j_2} \times \exp\left(\pm 2\pi i \left(\frac{j_1 k_1}{m} + \frac{j_2 k_2}{n}\right)\right),$$

where $k_1 = 0, 1, \ldots, m-1$, $k_2 = 0, 1, \ldots, n-1$.

(Note the scale factor of $\frac{1}{\sqrt{mn}}$ in this definition.) The **minus** sign is taken in the argument of exponential within the summation when the **forward** transform is required, and the **plus** sign is taken when the **backward** transform is required (see argument DIRECT). The above bivariate sequence and its DFT can also be expressed in the matrix forms:

$$Z = [z_{j_1 j_2}] = X + iY; \quad \hat{Z} = [\hat{z}_{j_1 j_2}] = \hat{X} + i\hat{Y};$$

where X, Y and $\hat{X}$, $\hat{Y}$ contain the real and the imaginary parts of Z and $\hat{Z}$.

Optionally, the trigonometric coefficients generated by a previous call to C06FUFP for a problem of the same dimensions ($m \times n$) can be reused, thereby avoiding their repeated calculation.

2 Specification

```
   SUBROUTINE C06FUFP(ICNTXT, M, N, X, Y, INIT, TRIG, DIRECT, TRANS,
  1                   MX, WORK, IFAIL)
   DOUBLE PRECISION   X(*), Y(*), TRIG(2*(N+M)), WORK(*)
   INTEGER            ICNTXT, M, N, MX, IFAIL
   CHARACTER*1        INIT, DIRECT, TRANS
```

3 Usage

3.1 Definitions

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

p	–	the total number of processors in the Library Grid.
M_b	–	the nominal number of rows of X, Y, $\hat{X}$ and $\hat{Y}$, or of $\hat{X}^T$ and $\hat{Y}^T$, held locally on a logical processor (see Section 3.3).
M_x	–	the actual number of rows of X, Y, $\hat{X}$ and $\hat{Y}$, or of $\hat{X}^T$ and $\hat{Y}^T$, held locally on a logical processor, where $0 \leq M_x \leq M_b$ (see argument MX).
$\lceil x \rceil$	–	the ceiling function of x, which gives the smallest integer which is not less than x.

3.2 Global and Local Arguments

The 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, INIT, TRIG, DIRECT, TRANS, IFAIL

Global output arguments: TRIG, IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

The matrices X and Y are distributed in row-block fashion, that is rows of X and Y are allocated to logical processors on the two-dimensional grid row by row (i.e., in row major ordering of the grid), starting from the $\{0,0\}$ logical processor. Each processor, that contains rows of X and Y, contains $M_b = \lceil m/p \rceil$ consecutive rows, except the last processor that actually contains data, for which the number of rows may be less than M_b. This processor holds mod(m,M_b) rows if mod(m,M_b) $\neq 0$ and M_b rows otherwise.

This routine provides a facility which allows users to store the transforms in their transpose form. This avoids one of the two global transpose operations performed in the two-dimensional FFT routine (see Section 6.2), and hence reduces the execution time of the routine as all the communication is performed in the global transpose step. However, this will result in having the Fourier transform in a **different order** with respect to the original transform. Hence, if TRANS = 'Y' then the Fourier transforms are stored in the same data distribution as the original sequence. If TRANS = 'N' then the Fourier transform are stored in transposed form, that is the transformed matrices $\hat{X}^T$ and $\hat{Y}^T$ are allocated analogously to the input matrices, with n playing the role of m and vice versa.

3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data in row-block form.

The following routines may be used in conjunction with C06FUFP:

Real matrix generation: F01ZMFP

Real matrix output: X04BFFP

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: M — INTEGER *Global Input*

On entry: m, the first dimension of the bivariate sequence to be transformed.

Constraint: M ≥ 1.

3: N — INTEGER *Global Input*

On entry: n, the second dimension of the bivariate sequence to be transformed.

Constraint: N ≥ 1.

4: X($*$) — DOUBLE PRECISION array *Local Input/Local Output*

Note: the dimension of the array X must be at least max($\lceil m/p \rceil \times n, m \times \lceil n/p \rceil$).

Distribution: array X is formally defined as a vector. However, you may find it more convenient to consider X as a two-dimensional array X(0:M_x−1, 0:N−1), that is X(i,j) corresponds to X($1 + i + j \times M_x$). If TRANS = 'N', on exit X is considered as X(0:$M_x - 1$, 0:M−1). The array X is not referenced if $M_x = 0$.

On entry: the local parts of X which define the real parts of the complex bivariate sequence Z stored in row block distribution (see Section 3.3).

On exit: the real part of the Fourier transform.

If TRANS = 'Y', the real parts of the corresponding elements of the computed transform stored in the same row block distribution.

If TRANS = 'N', X contains the real parts of the transform in transposed form.

5: Y(∗) — DOUBLE PRECISION array *Local Input/Local Output*

Note: the dimension of the array Y must be at least $\max(\lceil m/p \rceil \times n, m \times \lceil n/p \rceil)$.

Distribution: array Y is formally defined as a vector. However, you may find it more convenient to consider Y as a two-dimensional array Y(0:M_x−1, 0:N−1), that is Y(i, j) corresponds to Y($1 + i + j \times M_x$). If TRANS = 'N', on exit Y is considered as Y(0:M_x−1, 0:M−1). The array Y is not referenced if $M_x = 0$.

On entry: the local parts of Y which define the imaginary parts of the complex bivariate sequence Z stored in row block distribution (see Section 3.3).

On exit: the imaginary part of the Fourier transform.

If TRANS = 'Y', the imaginary parts of the corresponding elements of the computed transform stored in the same row block distribution.
If TRANS = 'N', Y contains the imaginary parts of the transform in transposed form.

6: INIT — CHARACTER*1 *Global Input*

On entry: if the trigonometric coefficients required to compute the transforms are to be calculated by the routine and stored in the array TRIG, then INIT must be set equal to 'I' (**I**nitial call).

If INIT = 'S' (**S**ubsequent call), then the routine assumes that trigonometric coefficients for the specified values of m and n are supplied in the array TRIG, having been calculated in a previous call to the routine.

Constraint: INIT = 'I' or 'S'.

7: TRIG(2∗(M+N)) — DOUBLE PRECISION array *Global Input/Global Output*

Note: users are advised not to change the elements of the array TRIG.

On entry: if INIT = 'S', TRIG must contain the required trigonometric coefficients calculated in a previous call to the routine. Otherwise TRIG does not need to be set.

On exit: TRIG contains the required coefficients, computed by the routine if INIT = 'I'. If the routine has been called with INIT = 'I', then TRIG(1 : 2∗M) contains the trigonometric coefficients related to M and TRIG(2∗M+1 : 2∗M+2∗N) contains the trigonometric coefficients related to N. If the routine has been called with INIT = 'S', then the trigonometric coefficients are the same as on entry to the routine.

8: DIRECT — CHARACTER*1 *Global Input*

On entry: the direction of transform to be computed.

If DIRECT = 'F', then **F**orward transform as defined in Section 1 will be computed.
If DIRECT = 'B', then **B**ackward transform as defined in Section 1 will be computed.

Constraint: DIRECT = 'F' or 'B'.

9: TRANS — CHARACTER*1 *Global Input*

On entry: specifies how the output arrays X and Y are to be locally stored.

If TRANS = 'Y', then on exit, X and Y are locally stored in an identical manner to the input arrays.
If TRANS = 'N', then on exit, X and Y are locally stored in transposed form, i.e., the number of rows stored locally will be different from the number on input.

Constraint: TRANS = 'Y' or 'N'.

10: MX — INTEGER *Local Output*

On exit:

If TRANS = 'Y', M_x, the actual number of rows of X and Y (and of $\hat{X}$ and $\hat{Y}$), held on the logical processor.
If TRANS = 'N', M_x, the actual number of rows of $\hat{X}^T$ and $\hat{Y}^T$.

11: WORK(∗) — DOUBLE PRECISION array *Local Workspace*

Note: the dimension of the array WORK must be at least $2\times\max(\lceil m/p\rceil \times n, m \times \lceil n/p\rceil)$.

12: 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).

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.

6 Further Comments

6.1 Algorithmic Detail

The two-dimensional DFT of a bivariate sequence $Z = [z_{j_1 j_2}]$ of length $m \times n$ is obtained performing m one-dimensional transforms of length n, on the rows of the matrix Z, and n one-dimensional transforms of length m, on the rows of the transposed resulting matrix, i.e. on the columns of the resulting matrix. The algorithm used to perform the one-dimensional transforms is the *Stockham self-sorting algorithm*, described in Temperton [4].

6.2 Parallelism Detail

The parallel algorithm is based on a row block distribution of the matrix Z, as described in Section 3.3. The FFT is computed by performing the following four basic steps.

(1) **FFT on rows:** Each processor performs M_x one-dimensional transforms of length n, on the rows assigned to it.
(2) **Matrix Transpose:** Because column elements of the sequence as a whole are not stored in contiguous memory location, row/column transposition of matrix Z is necessary prior to computing FFT on columns.

(3) **FFT on columns:** Same as Step 1, but by columns.
(4) **Matrix Transpose:** To have the matrix $\hat{Z}$ in the same distribution as Z, it is necessary to perform a transposition of the matrix computed in Step 3. This can be achieved by setting TRANS to 'Y'. However, if TRANS is set to 'N', this step is not performed by C06FUFP.

More details of this algorithm are given in Carracciuolo *et al.* [1].

Note that avoiding the last transposition leads to a significant reduction of the execution time. This choice is recommended when a forward transform followed by a backward transform has to be computed, since the second transform can be computed directly from a matrix in transposed form, i.e.,:

```
*     Some Initializations
      .
      .
      DIRECT = 'F'
      TRANS = 'N'
*
*     Compute forward transform
*
      CALL C06FUFP(ICNTXT,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,MX,WORK,IFAIL)
*
*     Some Initializations
      .
      .

      DIRECT = 'B'
      TRANS = 'N'
*
*     Compute backward transform
*
      CALL C06FUFP(ICNTXT,N,M,X,Y,INIT,TRIG,DIRECT,TRANS,MX,WORK,IFAIL)
```

The above two calls to C06FUFP avoids two transposition steps which will result in a significant gain in performance.

6.3 Accuracy

An upper bound of the roundoff error in the computed DFT is given by:

$$\frac{\|\tilde{z}-\hat{z}\|}{\|\hat{z}\|} \leq 1.06\sqrt{mn}\left(\sum_{i=1}^{r}(2m_i)^{\frac{3}{2}}+\sum_{j=1}^{s}(2n_j)^{\frac{3}{2}}\right)\varepsilon$$

where $\hat{z}$ is the **exact** DFT of z, $\tilde{z}$ is the **computed** one, $\|u\|$ is the following norm:

$$\|u\| = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} \mid u_{ij} \mid^2\right)^{\frac{1}{2}},$$

$m = m_1 \times m_2 \times \ldots \times m_r$ and $n = n_1 \times n_2 \times \ldots \times n_s$ are the factorizations of n and m used by the FFT algorithm, and ε is the machine precision. The input data is assumed to be exactly representable within the limits of machine precision and the roundoff errors in computing trigonometric coefficients are not considered.

However, experiments have shown that the actual errors are usually lower than the above error bound. For details on the roundoff error analysis see Gentleman and Sande [2] and Ramos [3].

The routine computes first all the factors of 6 and 4 in m and n, then all the factors of 2 and 3 and finally all the other factors, but does not give these factors in output. The value of ε can be computed using the routine X02AJF.

Some indication of accuracy can be obtained performing a direct transform followed by an inverse transform and comparing the results with the original sequence (in exact arithmetic they should be identical).

6.4 Computational Costs

The number of floating-point operations per processor is approximately proportional to $\frac{m \times n}{p}(\log n + \log m)$, but also depends on the factorization of the individual dimensions m and n.

The number of floating-point values that each processor exchanges with each other is approximately $4 \times \frac{m \times n}{p^2}$ if TRANS = 'Y', and approximately $2 \times \frac{m \times n}{p^2}$ if TRANS = 'N'.

7 References

[1] Carracciuolo L, de Bono I, de Cesare M L, di Serafino D and Perla F (1997) Development of a Parallel Two-Dimensional Mixed-Radix FFT Routine *Tech. Rep. TR-97-8* Center for Research on Parallel Computing and Supercomputers (CPS) – CNR, Naples, Italy

[2] Gentleman W M and Sande G (1966) Fast Fourier Transform – For Fun and Profit *AFIPS Proceedings 1996 Fall Joint Conference* **29** Spartan Books 563–578

[3] Ramos G U (1971) Roundoff Error Analysis of the Fast Fourier Transform *Math. Comp.* **25** 757–768

[4] Temperton C (1983) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

8 Example

To compute the forward and the backward DFT of a bivariate sequence Z, where Z is given by

$$Z = X + \sqrt{-1}Y$$

and

$$X = [\cos(2\pi ij/n)], \quad Y = [\sin(2\pi ij/n)]; \quad \text{for} \quad i = 1, \ldots, m \text{ and } j = 1, \ldots, n,$$

The routine F01ZMFP is used to generate the matrices X and Y on a 2 by 2 Library Grid. The actual number of rows of the matrices X and Y on each logical processor, M_x, is equal to 2 on the logical processors {0,0}, {0,1} and {1,0}, and is equal to 1 on the logical processor {1,1}.

The routine C06FUFP is used to compute the forward and the backward DFT. The actual number of rows of the transformed matrices on each logical processor is the same as for the input data.

The routine X04BFFP is used to output the generated data and the computed transforms.

8.1 Example Text

```
*     C06FUFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          LD, MMAX, NMAX
      PARAMETER        (LD=4,MMAX=15,NMAX=15)
      INTEGER          LWORK
      PARAMETER        (LWORK=2*LD*NMAX)
*     .. Local Scalars ..
      INTEGER          ICNTXT, ICOFF, IFAIL, IM, LDXY, M, MP, MX,
     +                 MYPCOL, MYPROW, N, NP, NUMPROCS
      LOGICAL          ROOT
      CHARACTER        DIRECT, INIT, TRANS
      CHARACTER*80     CNUMOP, FORMAT, TITOP
*     .. Local Arrays ..
      DOUBLE PRECISION TRIG(2*MMAX+2*NMAX), WORK(LWORK), X(LD*NMAX),
     +                 Y(LD*NMAX)
*     .. External Functions ..
```

```
      INTEGER          Z01CFFP
      LOGICAL          Z01ACFP
      EXTERNAL         Z01CFFP, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         C06FUFP, F01ZMFP, GMATX, GMATY, X04BFFP, Z01AAFP,
     +                 Z01ABFP, Z01ZAFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'C06FUFP Example Program Results'
*
*     Set up 2x2 Library Grid
*
      MP = 2
      NP = 2
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Obtain other Library Grid information
*     Calculate my processor id = IM
      CALL Z01ZAFP(ICNTXT,MP,NP,MYPROW,MYPCOL)
      NUMPROCS = MP*NP
      IM = MYPROW*NP + MYPCOL
*
*     Initialize problem size less than MMAX, NMAX
*
      M = 7
      N = 4
*
*     Z01CFFP compute the leading dimension of X and Y, LDXY,
*     for use by F01ZMFP.
*
      LDXY = Z01CFFP(NUMPROCS,M,IM)
*
*     Generate vecotrs X and Y
*
      IFAIL = 0
      CALL F01ZMFP(ICNTXT,GMATX,M,N,X,LDXY,MX,IFAIL)
*
      IFAIL = 0
      CALL F01ZMFP(ICNTXT,GMATY,M,N,Y,LDXY,MX,IFAIL)
*
*     Print generated data
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Generated data'
         WRITE (NOUT,*)
      END IF
*
      FORMAT = 'F7.3'
      TITOP = 'Y'
      CNUMOP = 'N'
      ICOFF = 0
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X: Real'
         WRITE (NOUT,*)
      END IF
```

```
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,X,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Y: Imag'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,Y,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
*     Compute forward transform
*
      TRANS = 'Y'
      INIT = 'I'
      DIRECT = 'F'
      IFAIL = 0
*
      CALL C06FUFP(ICNTXT,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,MX,WORK,IFAIL)
*
*     Print results
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Components of Discrete Fourier Transform'
         WRITE (NOUT,*)
      END IF
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X: Real'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,X,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Y: Imag'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,Y,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
*     Compute backward transform
*
      INIT = 'S'
      DIRECT = 'B'
      TRANS = 'Y'
      IFAIL = 0
*
      CALL C06FUFP(ICNTXT,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,MX,WORK,IFAIL)
*
```

```
*     Print results
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original sequence as restored by Backward',
     +     ' Transform'
         WRITE (NOUT,*)
      END IF
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X: Real'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,X,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Y: Imag'
         WRITE (NOUT,*)
      END IF
      IFAIL = 0
*
      CALL X04BFFP(ICNTXT,NOUT,MX,N,Y,LDXY,FORMAT,TITOP,CNUMOP,ICOFF,
     +             WORK,LDXY,IFAIL)
*
*     Leave processor grid
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
      STOP
      END
*
      SUBROUTINE GMATX(I1,IL,N,X,LDX)
*
*     GMATX generates the block X(I1:IL,1:N) of the M by N matrix
*     X with elements cos((i*j)*2*pi)
*
*     .. Scalar Arguments ..
      INTEGER          I1, IL, LDX, N
*     .. Array Arguments ..
      DOUBLE PRECISION X(LDX,*)
*     .. Local Scalars ..
      DOUBLE PRECISION PI2
      INTEGER          I, J, K
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC        COS, DBLE
*     .. Executable Statements ..
      PI2 = 2.0D0*X01AAF(0.0D0)/DBLE(N)
      K = 1
      DO 40 I = I1, IL
         DO 20 J = 1, N
            X(K,J) = COS(DBLE(I*J)*PI2)
   20    CONTINUE
```

```
            K = K + 1
   40 CONTINUE
*
*     End of GMATX
*
      RETURN
      END
*
      SUBROUTINE GMATY(I1,IL,N,Y,LDY)
*
*     GMATY generates the block Y(I1:IL,1:N) of the 7 by 4 matrix
*     Y with elemets  sin((i*j)*2*pi)
*
*     .. Scalar Arguments ..
      INTEGER          I1, IL, LDY, N
*     .. Array Arguments ..
      DOUBLE PRECISION Y(LDY,*)
*     .. Local Scalars ..
      DOUBLE PRECISION PI2
      INTEGER          I, J, K
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE, SIN
*     .. Executable Statements ..

      PI2 = 2.0D0*X01AAF(0.0D0)/DBLE(N)
      K = 1
      DO 40 I = I1, IL
         DO 20 J = 1, N
            Y(K,J) = SIN(DBLE(I*J)*PI2)
   20    CONTINUE
         K = K + 1
   40 CONTINUE
*
*     End of GMATY
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
C06FUFP Example Program Results

Generated data

X: Real

  Array from logical processor    0,   0

   0.000 -1.000  0.000  1.000
```

```
  -1.000  1.000 -1.000  1.000

  Array from logical processor    0,   1

   0.000 -1.000  0.000  1.000
   1.000  1.000  1.000  1.000

  Array from logical processor    1,   0

   0.000 -1.000  0.000  1.000
  -1.000  1.000 -1.000  1.000

  Array from logical processor    1,   1

   0.000 -1.000  0.000  1.000

Y: Imag

  Array from logical processor    0,   0

   1.000  0.000 -1.000  0.000
   0.000  0.000  0.000  0.000

  Array from logical processor    0,   1

  -1.000  0.000  1.000  0.000
   0.000  0.000  0.000  0.000

  Array from logical processor    1,   0

   1.000  0.000 -1.000  0.000
   0.000  0.000  0.000  0.000

  Array from logical processor    1,   1

  -1.000  0.000  1.000  0.000


Components of Discrete Fourier Transform

X: Real

  Array from logical processor    0,   0

   0.756  0.000 -1.512  0.000
  -0.681 -0.328 -0.303 -0.146

  Array from logical processor    0,   1

   0.471  0.591  0.849  1.065
  -0.168 -0.737  0.210  0.919

  Array from logical processor    1,   0

  -0.168  0.737  0.210 -0.919
   0.471 -0.591  0.849 -1.065

  Array from logical processor    1,   1
```

```
  -0.681  0.328 -0.303  0.146

Y: Imag

  Array from logical processor    0,   0

   0.000  1.512  0.000 -1.512
  -0.328  0.075 -0.146 -0.303

  Array from logical processor    0,   1

   0.591  1.227  1.065  0.849
  -0.737  0.588  0.919  0.210

  Array from logical processor    1,   0

   0.737  0.588 -0.919  0.210
  -0.591  1.227 -1.065  0.849

  Array from logical processor    1,   1

   0.328  0.075  0.146 -0.303


Original sequence as restored by Backward Transform

X: Real

  Array from logical processor    0,   0

   0.000 -1.000  0.000  1.000
  -1.000  1.000 -1.000  1.000

  Array from logical processor    0,   1

   0.000 -1.000  0.000  1.000
   1.000  1.000  1.000  1.000

  Array from logical processor    1,   0

   0.000 -1.000  0.000  1.000
  -1.000  1.000 -1.000  1.000

  Array from logical processor    1,   1

   0.000 -1.000  0.000  1.000

Y: Imag

  Array from logical processor    0,   0

   1.000  0.000 -1.000  0.000
   0.000  0.000  0.000  0.000

  Array from logical processor    0,   1

  -1.000  0.000  1.000  0.000
```

```
 0.000  0.000  0.000  0.000

Array from logical processor    1,   0

 1.000  0.000 -1.000  0.000
 0.000  0.000  0.000  0.000

Array from logical processor    1,   1

-1.000  0.000  1.000  0.000
```