C06MXFP

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

C06MXFP computes the three-dimensional Discrete Fourier Transform (DFT) of a trivariate sequence of complex data values $z_{j_1 j_2 j_3}$, where $j_1 = 0, 1, \ldots, l-1$, $j_2 = 0, 1, \ldots, m-1$, and $j_3 = 0, 1, \ldots, n-1$.

The DFT is here defined by:

$$\hat{z}_{k_1 k_2 k_3} = \frac{1}{\sqrt{lmn}} \sum_{j_1=0}^{l-1} \sum_{j_2=0}^{m-1} \sum_{j_3=0}^{n-1} z_{j_1 j_2 j_3} \times \exp\left(\pm 2\pi i \left(\frac{j_1 k_1}{l} + \frac{j_2 k_2}{m} + \frac{j_3 k_3}{n}\right)\right),$$

where $k_1 = 0, 1, \ldots, l-1$, $k_2 = 0, 1, \ldots, m-1$, and $k_3 = 0, 1, \ldots, n-1$.

(Note the scale factor of $\frac{1}{\sqrt{lmn}}$ 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 trivariate sequence and its DFT can also be expressed as three-dimensional arrays of the form:

$$Z = [z_{j_1 j_2 j_3}] = X + iY, \quad \hat{Z} = [\hat{z}_{j_1 j_2 j_3}] = \hat{X} + i\hat{Y}$$

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 C06MXFP for a problem of the same dimensions ($l \times m \times n$) can be reused, thereby avoiding their repeated calculation.

2 Specification

```
   SUBROUTINE C06MXFP(ICNTXT, L, M, N, X, Y, INIT, TRIG, DIRECT,
  1                   TRANS, LX, WORK, IFAIL)
   DOUBLE PRECISION   X(*), Y(*), TRIG(2*(L+M+N)), WORK(*)
   INTEGER            ICNTXT, L, M, N, LX, IFAIL
   CHARACTER*1        INIT, DIRECT, TRANS
```

3 Usage

3.1 Definitions

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

n_p	–	the number of columns in the Library Grid.
L_b	–	the maximum number of jk-planes 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).
L_x	–	the actual number of jk-planes of X, Y, $\hat{X}$ and $\hat{Y}$, or of $\hat{X}^T$ and $\hat{Y}^T$ held locally on a logical processor, where $0 \leq L_x \leq L_b$.
$\lceil x \rceil$	–	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: L, M, N, INIT, TRIG, DIRECT, TRANS, IFAIL

Global output arguments: TRIG, IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

The arrays X and Y are distributed in three-dimensional i-block form, that is blocks of consecutive jk-planes of X and Y, obtained by partitioning X and Y along the first dimension i, 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 an i-block of X and Y, contains $L_b = \lceil l/p \rceil$ consecutive jk-planes, except the last processor that actually contains data, for which the number of jk-planes may be less than L_b. This processor holds mod(l,L_b) jk-planes if mod(l,L_b) $\neq 0$ and L_b jk-planes 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 three-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 output data are stored in transposed order, that is the three-dimensional arrays $\hat{X}^T$ and $\hat{Y}^T$, obtained exchanging the first and the third dimension of $\hat{X}$ and $\hat{Y}$, are allocated analogously to the input arrays, with n playing the role of l and vice versa.

3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data in matrix and three-dimensional block form. The following routines may be used in conjunction with C06MXFP:

Real three-dimensional block generation:	F01ZHFP
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: L — INTEGER *Global Input*

On entry: l, the first dimension of the trivariate sequence to be transformed.

Constraint: L ≥ 1.

3: M — INTEGER *Global Input*

On entry: m, the second dimension of the trivariate sequence to be transformed.

Constraint: M ≥ 1.

4: N — INTEGER *Global Input*

On entry: n, the third dimension of the trivariate sequence to be transformed.

Constraint: N ≥ 1.

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

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

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

On entry: the local parts of X which define the real parts of the complex trivariate sequence Z stored in three-dimensional i-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 three-dimensional i-block distribution.

If TRANS = 'N', X contains the real parts of the transform in transposed form, with the first and the third dimension interchanged.

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

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

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

On entry: the local parts of Y which define the imaginary parts of the complex trivariate sequence Z stored in three-dimensional i-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 three-dimensional i-block distribution.

If TRANS = 'N', Y contains the imaginary parts of the transform in transposed form, with the first and the third dimension interchanged.

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

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

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

Note: users are advised not to change 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.

On exit: TRIG contains the trigonometric coefficients, computed by the routine if INIT = 'I'. If the routine has been called with INIT = 'I', then TRIG(1:2∗L) contains the trigonometric coefficients related to L, TRIG(2∗L+1:2∗L+2∗M) contains the trigonometric coefficients related to M, and TRIG(2∗L+2∗M+1:2∗L+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.

9: 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'.

10: 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 blocks held locally in the first and the third dimension will be different from the input.

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

11: LX — INTEGER *Local Output*

On exit:

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

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

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

13: 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 three-dimensional DFT of a trivariate sequence $Z = (z_{j_1 j_2 j_3})$ of length $l\times m\times n$ is obtained performing $l\times m$ one-dimensional transforms of length n, along the first dimension of the array Z, $l\times n$ one-dimensional transforms of length m, along the second dimension of the array Z, and $m\times n$ one-dimensional transforms of length l, along the third dimension of the array Z. 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 three-dimensional i-block distribution of the three-dimensional array Z, as described in Section 3. The FFT is computed by performing the following four basic steps:

(1) **FFT on planes:** each processor performs $L_x \times m$ one-dimensional transforms of length n, on all local vectors in the direction corresponding to n, and performs about $L_x \times n$ one-dimensional transforms of length m, on all local vectors in the direction corresponding to m.
(2) **Transpose:** transposition is performed on the first and third dimension of the global three-dimensional matrix Z, obtained from the previous transforms.
(3) **FFT on transpose:** same as Step 1, but by the direction of l, on its local vectors in the direction corresponding to l.
(4) **Transpose:** to have the three-dimensional array $\hat{Z}$ in the same distribution as Z, it is necessary to perform a transposition in the first and third dimension of the resulting matrix. This can be achieved by setting TRANS to 'Y'. However, if TRANS is set to 'N', this step is not performed by C06MXFP.

This algorithm is a generalization to the three-dimensional case of the algorithm implemented in the two-dimensional FFT routine C06FUFP 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 C06MXFP(ICNTXT,L,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,LX,WORK,
     *             IFAIL)
*
*     Some Initializations
      .
      .

      DIRECT = 'B'
      TRANS = 'N'
*
*     Compute backward transform
*
      CALL C06MXFP(ICNTXT,N,M,L,X,Y,INIT,TRIG,DIRECT,TRANS,LX,WORK,
     *             IFAIL)
```

The above two calls to C06MXFP avoids two transposition steps which will result into significant gain in performance.

6.3 Accuracy

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

$$\frac{\|\tilde{z} - \hat{z}\|}{\|\hat{z}\|} \leq 1.06\sqrt{lmn}\left(\sum_{i=1}^{r}(2l_i)^{\frac{3}{2}} + \sum_{j=1}^{s}(2m_j)^{\frac{3}{2}} + \sum_{k=1}^{t}(2n_k)^{\frac{3}{2}}\right)\varepsilon$$

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

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

$l = l_1 \times l_2 \times \ldots \times l_r$, $m = m_1 \times m_2 \times \ldots \times m_s$ and $n = n_1 \times n_2 \times \ldots \times n_t$ are the factorizations of l, m and n used by the FFT algorithm, and ε is the machine precision. The input data are 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 4 and 6 in l, m and n, then all the factors of 2 and 3 and finally all the other prime 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 otained performing a direct transform followed by an inverse transform and comparing the results with the original sequence (in exact arithmetic they would be identical).

6.4 Computational Costs

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

$$Z = X + iY = (x_{j_1 j_2 j_3}) + i(y_{j_1 j_2 j_3}), \qquad \begin{array}{l} j_1 = 0, 1, \ldots, l-1, \\ j_2 = 0, 1, \ldots, m-1, \\ j_3 = 0, 1, \ldots, n-1, \end{array}$$

with

$$x_{j_1 j_2 j_3} = (j_1 + 6)(j_2 + 1) + j_3, \qquad y_{j_1 j_2 j_3} = (j_1 + 1) + (j_2 + 1)(j_3 + 4),$$

and l =4, $m = 3$ and $n = 2$.

The routine F01ZHFP is used to generate the three-dimensional arrays X and Y on a 2 by 2 logical processor grid. The actual number of jk-planes of X and Y on each logical processor, L_x, is equal to 1. Note that, in the routines GMATX and GMATY used by F01ZHFP, the indices I, J and K, corresponding to j_1, j_2 and j_3, start from 0 rather than from 1.

The routine C06MXFP is used to compute the forward and the backward DFT. The actual number of jk-planes of the transformed arrays on each logical processor is the same as for the input data.

The routine X04BFFP is used to print the input data, the forward DFT and the backward DFT. Note that ij-planes, rather than jk-planes, are printed.

8.1 Example Text

```
*     C06MXFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          L, M, N
      PARAMETER        (L=4,M=3,N=2)
      INTEGER          MG, NG
      PARAMETER        (MG=2,NG=2)
      INTEGER          LDMAX, LMAX, MMAX, NMAX, MAXDIM, LWORK
      PARAMETER        (LDMAX=3,LMAX=10,MMAX=10,NMAX=10,
     +                 MAXDIM=LDMAX*MMAX*NMAX,LWORK=2*MAXDIM)
*     .. Local Scalars ..
      INTEGER          ICNTXT, ICOFF, IFAIL, IM, K, LD1, LD2, LX, MP,
     +                 MYPCOL, MYPROW, NP, P
      LOGICAL          ROOT
      CHARACTER        DIRECT, INIT, TRANS
      CHARACTER*80     CNUMOP, FORMT, TITOP
*     .. Local Arrays ..
      DOUBLE PRECISION TRIG(2*LMAX+2*MMAX+2*NMAX), WORK(LWORK),
     +                 X(MAXDIM), Y(MAXDIM)
*     .. External Functions ..
      INTEGER          Z01CFFP
      LOGICAL          Z01ACFP
      EXTERNAL         Z01CFFP, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         C06MXFP, F01ZHFP, GMATX, GMATY, X04BFFP, Z01AAFP,
     +                 Z01ABFP, Z01ZAFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'C06MXFP Example Program Results'
         WRITE (NOUT,*)
         WRITE (NOUT,99999)
         WRITE (NOUT,99998) L, M, N
         WRITE (NOUT,*)
         WRITE (NOUT,*)
      END IF
*
*     Define 2d processor grid
*
      MP = MG
      NP = NG
*
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      CALL Z01ZAFP(ICNTXT,MP,NP,MYPROW,MYPCOL)
      P = MP*NP
      IM = MYPROW*NP + MYPCOL
*
*     Compute leading dimensions of X and Y, LD1 and LD2, as required
*     by F01ZHFP. Since X and Y are formally 1-d, LD1 is the actual
*     number of jk-planes of X and Y held by the logical processor,
*     that is LX, and LD2 is equal to M.
*
```

```
      LD1 = Z01CFFP(P,L,IM)
      LD2 = M
*
*     Generate arrays the X and Y.
*
      IFAIL = 0
      CALL F01ZHFP(ICNTXT,GMATX,L,M,N,X,LD1,LD2,LX,IFAIL)
      IFAIL = 0
      CALL F01ZHFP(ICNTXT,GMATY,L,M,N,Y,LD1,LD2,LX,IFAIL)
*
*     Print generated data
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Generated data'
         WRITE (NOUT,*)
      END IF
*
      FORMT = 'F11.3'
      TITOP = 'Y'
      CNUMOP = 'L'
      ICOFF = 0
*
*     Print ij-planes of array X
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X (Real):'
      END IF
      DO 20 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,X(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
   20 CONTINUE
*
*     Print ij-planes of array Y
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Y (Imag):'
      END IF
      DO 40 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,Y(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
   40 CONTINUE
*
*     Compute forward transform
```

```
*
      INIT = 'I'
      DIRECT = 'F'
      TRANS = 'Y'
      IFAIL = 0
      CALL C06MXFP(ICNTXT,L,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,LX,WORK,
     +             IFAIL)
*
*     Print results
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Components of Discrete Fourier Transform'
         WRITE (NOUT,*)
      END IF
*
*     Print ij-planes of array X
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X (Real):'
      END IF
      DO 60 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,X(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
   60 CONTINUE
*
*     Print ij-planes of array Y
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Y (Imag):'
      END IF
      DO 80 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,Y(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
   80 CONTINUE
*
*     Compute backward transform
*
      INIT = 'S'
      DIRECT = 'B'
      TRANS = 'Y'
      IFAIL = 0
```

```
      CALL C06MXFP(ICNTXT,L,M,N,X,Y,INIT,TRIG,DIRECT,TRANS,LX,WORK,
     +             IFAIL)
*
*     Print results
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original sequence as restored by backward',
     +     ' Transform'
         WRITE (NOUT,*)
      END IF
*
*     Print ij-planes of array X
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'X (Real):'
      END IF
      DO 100 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,X(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
  100 CONTINUE
*
*     Print ij-planes of array Y
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Y (Imag):'
      END IF
      DO 120 K = 0, N - 1
         IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'ij-plane ', K + 1, ' from each',
     +        ' logical processor'
            WRITE (NOUT,*)
         END IF
         IFAIL = 0
         CALL X04BFFP(ICNTXT,NOUT,LX,M,Y(1+K*LX*M),LX,FORMT,TITOP,
     +                CNUMOP,ICOFF,WORK,LX,IFAIL)
  120 CONTINUE
*
*     Undefine 2d processor grid
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
*
99999 FORMAT (/'    L    M    N')
99998 FORMAT (3I5)
      END
```

```
*
      SUBROUTINE GMATX(I1,I2,M,N,X,LD1,LD2)
*
*     GMATX generates the block X(I1:I2, 1:M, 1:N) of the array X
*     such that
*                          X(I,J,K) = (I+5)*J+K-1,
*     where I=1,...,L, J=1,...,M, K=1,...,N.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, LD1, LD2, M, N
*     .. Array Arguments ..
      DOUBLE PRECISION X(LD1,LD2,*)
*     .. Local Scalars ..
      INTEGER          I, J, K, L
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      DO 60 K = 1, N
         DO 40 J = 1, M
            L = 1
            DO 20 I = I1, I2
               X(L,J,K) = DBLE((I+5)*J+K-1)
               L = L + 1
   20       CONTINUE
   40    CONTINUE
   60 CONTINUE
*
*     End of GMATX.
*
      RETURN
      END
*
      SUBROUTINE GMATY(I1,I2,M,N,Y,LD1,LD2)
*
*     GMATY generates the block Y(I1:I2, 1:M, 1:N) of the array Y
*     such that
*                          Y(I,J,K) = I+J*(K+3),
*     where I=1,...,L, J=1,...,M, K=1,...,N.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, LD1, LD2, M, N
*     .. Array Arguments ..
      DOUBLE PRECISION Y(LD1,LD2,*)
*     .. Local Scalars ..
      INTEGER          I, J, K, L
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      DO 60 K = 1, N
         DO 40 J = 1, M
            L = 1
            DO 20 I = I1, I2
               Y(L,J,K) = DBLE(I+J*(K+3))
               L = L + 1
   20       CONTINUE
   40    CONTINUE
   60 CONTINUE
*
```

```
*     End of GMATY.
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
C06MXFP Example Program Results


   L    M    N
   4    3    2


Generated data

X (Real):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      6.000     12.000     18.000

  Array from logical processor    0,   1

          1          2          3
      7.000     14.000     21.000

  Array from logical processor    1,   0

          1          2          3
      8.000     16.000     24.000

  Array from logical processor    1,   1

          1          2          3
      9.000     18.000     27.000


ij-plane            2 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      7.000     13.000     19.000

  Array from logical processor    0,   1

          1          2          3
      8.000     15.000     22.000
```

```
  Array from logical processor    1,   0

          1          2          3
      9.000     17.000     25.000

  Array from logical processor    1,   1

          1          2          3
     10.000     19.000     28.000


Y (Imag):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      5.000      9.000     13.000

  Array from logical processor    0,   1

          1          2          3
      6.000     10.000     14.000

  Array from logical processor    1,   0

          1          2          3
      7.000     11.000     15.000

  Array from logical processor    1,   1

          1          2          3
      8.000     12.000     16.000


ij-plane            2 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      6.000     11.000     16.000

  Array from logical processor    0,   1

          1          2          3
      7.000     12.000     17.000

  Array from logical processor    1,   0

          1          2          3
      8.000     13.000     18.000

  Array from logical processor    1,   1

          1          2          3
      9.000     14.000     19.000
```

```
Components of Discrete Fourier Transform

X (Real):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0

          1          2          3
     75.934    -24.735    -12.007

  Array from logical processor    0,   1

          1          2          3
     -7.348      0.518      1.932

  Array from logical processor    1,   0

          1          2          3
     -4.899      1.225      1.225

  Array from logical processor    1,   1

          1          2          3
     -2.449      1.932      0.518


ij-plane            2 from each logical processor

  Array from logical processor    0,   0

          1          2          3
     -2.449      0.707     -0.707

  Array from logical processor    0,   1

          1          2          3
      0.000      0.000      0.000

  Array from logical processor    1,   0

          1          2          3
      0.000      0.000      0.000

  Array from logical processor    1,   1

          1          2          3
      0.000      0.000      0.000


Y (Imag):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0
```

```
           1          2          3
      56.338     -0.416    -21.629

  Array from logical processor    0,   1

           1          2          3
       2.449     -1.932     -0.518

  Array from logical processor    1,   0

           1          2          3
      -2.449     -0.707      0.707

  Array from logical processor    1,   1

           1          2          3
      -7.348      0.518      1.932


ij-plane            2 from each logical processor

  Array from logical processor    0,   0

           1          2          3
      -4.899      1.225      1.225

  Array from logical processor    0,   1

           1          2          3
       0.000      0.000      0.000

  Array from logical processor    1,   0

           1          2          3
       0.000      0.000      0.000

  Array from logical processor    1,   1

           1          2          3
       0.000      0.000      0.000



Original sequence as restored by backward Transform

X (Real):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0

           1          2          3
       6.000     12.000     18.000

  Array from logical processor    0,   1

           1          2          3
       7.000     14.000     21.000
```

```
  Array from logical processor    1,   0

          1          2          3
      8.000     16.000     24.000

  Array from logical processor    1,   1

          1          2          3
      9.000     18.000     27.000


ij-plane            2 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      7.000     13.000     19.000

  Array from logical processor    0,   1

          1          2          3
      8.000     15.000     22.000

  Array from logical processor    1,   0

          1          2          3
      9.000     17.000     25.000

  Array from logical processor    1,   1

          1          2          3
     10.000     19.000     28.000


Y (Imag):

ij-plane            1 from each logical processor

  Array from logical processor    0,   0

          1          2          3
      5.000      9.000     13.000

  Array from logical processor    0,   1

          1          2          3
      6.000     10.000     14.000

  Array from logical processor    1,   0

          1          2          3
      7.000     11.000     15.000

  Array from logical processor    1,   1

          1          2          3
      8.000     12.000     16.000
```

```
 ij-plane            2 from each logical processor

   Array from logical processor    0,   0

            1          2          3
        6.000     11.000     16.000

   Array from logical processor    0,   1

            1          2          3
        7.000     12.000     17.000

   Array from logical processor    1,   0

            1          2          3
        8.000     13.000     18.000

   Array from logical processor    1,   1

            1          2          3
        9.000     14.000     19.000
```