NAG Fortran Library Routine Document

C06FXF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

C06FXF computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
 SUBROUTINE C06FXF(N1, N2, N3, X, Y, INIT, TRIGN1, TRIGN2, TRIGN3, WORK,
1                  IFAIL)
 INTEGER           N1, N2, N3, IFAIL
 real              X(N1*N2*N3), Y(N1*N2*N3), TRIGN1(2*N1), TRIGN2(2*N2),
1                  TRIGN3(2*N3), WORK(2*N1*N2*N3)
 CHARACTER*1       INIT
```

3 Description

This routine computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values $z_{j_1 j_2 j_3}$, where $j_1 = 0, 1, \ldots, n_1 - 1$, $j_2 = 0, 1, \ldots, n_2 - 1$, $j_3 = 0, 1, \ldots, n_3 - 1$.

The discrete Fourier transform is here defined by:

$$\hat{z}_{k_1 k_2 k_3} = \frac{1}{\sqrt{n_1 n_2 n_3}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} z_{j_1 j_2 j_3} \times \exp\left(-2\pi i \left(\frac{j_1 k_1}{n_1} + \frac{j_2 k_2}{n_2} + \frac{j_3 k_3}{n_3}\right)\right),$$

where $k_1 = 0, 1, \ldots, n_1 - 1$, $k_2 = 0, 1, \ldots, n_2 - 1$, $k_3 = 0, 1, \ldots, n_3 - 1$.

(Note the scale factor of $\frac{1}{\sqrt{n_1 n_2 n_3}}$ in this definition.)

To compute the inverse discrete Fourier transform, defined with $\exp(+2\pi i(\ldots))$ in the above formula instead of $\exp(-2\pi i(\ldots))$, this routine should be preceded and followed by calls of C06GCF to form the complex conjugates of the data values and the transform.

This routine calls C06FRF to perform multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm (Brigham (1974)). It is designed to be particularly efficient on vector processors.

4 References

Brigham E O (1974) *The Fast Fourier Transform* Prentice-Hall

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

5 Parameters

1: N1 – INTEGER *Input*

On entry: the first dimension of the transform, n_1.

Constraint: N1 ≥ 1.

2: N2 – INTEGER *Input*

On entry: the second dimension of the transform, n_2.

Constraint: N2 ≥ 1.

3: N3 – INTEGER *Input*

On entry: the third dimension of the transform, n_3.

Constraint: N3 ≥ 1.

4: X(N1∗N2∗N3) – ***real*** array *Input/Output*
5: Y(N1∗N2∗N3) – ***real*** array *Input/Output*

On entry: the real and imaginary parts of the complex data values must be stored in arrays X and Y respectively. If X and Y are regarded as three-dimensional arrays of dimension $(0 : \mathrm{N1} - 1, 0 : \mathrm{N2} - 1, 0 : \mathrm{N3} - 1)$, then $\mathrm{X}(j_1, j_2, j_3)$ and $\mathrm{Y}(j_1, j_2, j_3)$ must contain the real and imaginary parts of $z_{j_1 j_2 j_3}$.

On exit: the real and imaginary parts respectively of the corresponding elements of the computed transform.

6: INIT – CHARACTER*1 *Input*

On entry: if the trigonometric coefficients required to compute the transforms are to be calculated by the routine and stored in the arrays TRIGN1, TRIGN2 and TRIGN3, 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 n_1, n_2 and n_3 are supplied in the arrays TRIGN1, TRIGN2 and TRIGN3, having been calculated in a previous call to the routine.

If INIT = 'R', (**R**estart), then the routine assumes that trigonometric coefficients for the specified values of n_1, n_2 and n_3 are supplied in the arrays TRIGN1, TRIGN2 and TRIGN3, but does not check that the routine has previously been called. This option allows the TRIGN1, TRIGN2 and TRIGN3 arrays to be stored in an external file, read in and re-used without the need for a call with INIT equal to 'I'. The routine carries out a simple test to check that the current values of n_1, n_2 and n_3 are compatible with the arrays TRIGN1, TRIGN2 and TRIGN3.

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

7: TRIGN1(2∗N1) – ***real*** array *Input/Output*
8: TRIGN2(2∗N2) – ***real*** array *Input/Output*
9: TRIGN3(2∗N3) – ***real*** array *Input/Output*

On entry: if INIT = 'S' or 'R', TRIGN1, TRIGN2 and TRIGN3 must contain the required coefficients calculated in a previous call of the routine. Otherwise TRIGN1, TRIGN2 and TRIGN3 need not be set. If $n_i = n_j$ the same array may be supplied for TRIGNi and TRIGNj, for $i, j = 1$, 2, 3.

On exit: TRIGN1, TRIGN2 and TRIGN3 contain the required coefficients (computed by the routine if INIT = 'I').

10: WORK(2∗N1∗N2∗N3) – ***real*** array *Workspace*

11: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the

value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N1 < 1.

IFAIL $= 2$

On entry, N2 < 1.

IFAIL $= 3$

On entry, N3 < 1.

IFAIL $= 4$

On entry, INIT is not one of 'I', 'S' or 'R'.

IFAIL $= 5$

Not used at this Mark.

IFAIL $= 6$

On entry, INIT $=$ 'S' or 'R', but at least one of the arrays TRIGN1, TRIGN2 and TRIGN3 is inconsistent with the current value of N1, N2 or N3.

IFAIL $= 7$

7 Accuracy

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

8 Further Comments

The time taken by the routine is approximately proportional to $n_1 n_2 n_3 \times \log(n_1 n_2 n_3)$, but also depends on the factorization of the individual dimensions n_1, n_2 and n_3. The routine is somewhat faster than average if their only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2.

9 Example

This program reads in a trivariate sequence of complex data values and prints the three-dimensional Fourier transform. It then performs an inverse transform and prints the sequence so obtained, which may be compared to the original data values.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     C06FXF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          N1MAX, N2MAX, N3MAX, NMAX
      PARAMETER        (N1MAX=16,N2MAX=16,N3MAX=16,
     +                 NMAX=N1MAX*N2MAX*N3MAX)
*     .. Local Scalars ..
      INTEGER          IFAIL, N, N1, N2, N3
*     .. Local Arrays ..
      real             TRIGN1(2*N1MAX), TRIGN2(2*N2MAX),
     +                 TRIGN3(2*N3MAX), WORK(2*NMAX), X(NMAX), Y(NMAX)
*     .. External Subroutines ..
      EXTERNAL         C06FXF, C06GCF, READXY, WRITXY
*     .. Executable Statements ..
      WRITE (NOUT,*) 'C06FXF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
   20 READ (NIN,*,END=40) N1, N2, N3
      N = N1*N2*N3
      IF (N.GE.1 .AND. N.LE.NMAX) THEN
         CALL READXY(NIN,X,Y,N1,N2,N3)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original data values'
         CALL WRITXY(NOUT,X,Y,N1,N2,N3)
         IFAIL = 0
*
*        Compute transform
         CALL C06FXF(N1,N2,N3,X,Y,'Initial',TRIGN1,TRIGN2,TRIGN3,WORK,
     +              IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Components of discrete Fourier transform'
         CALL WRITXY(NOUT,X,Y,N1,N2,N3)
*
*        Compute inverse transform
         CALL C06GCF(Y,N,IFAIL)
         CALL C06FXF(N1,N2,N3,X,Y,'Subsequent',TRIGN1,TRIGN2,TRIGN3,
     +               WORK,IFAIL)
         CALL C06GCF(Y,N,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     'Original sequence as restored by inverse transform'
         CALL WRITXY(NOUT,X,Y,N1,N2,N3)
         GO TO 20
      ELSE
         WRITE (NOUT,*) ' ** Invalid value of n1, n2 or n3'
      END IF
   40 STOP
      END
*
      SUBROUTINE READXY(NIN,X,Y,N1,N2,N3)
*     Read 3-dimensional complex data
*     .. Scalar Arguments ..
      INTEGER           N1, N2, N3, NIN
*     .. Array Arguments ..
      real              X(N1,N2,N3), Y(N1,N2,N3)
*     .. Local Scalars ..
      INTEGER           I, J, K
*     .. Executable Statements ..
      DO 40 I = 1, N1
         DO 20 J = 1, N2
            READ (NIN,*) (X(I,J,K),K=1,N3)
```

```
            READ (NIN,*) (Y(I,J,K),K=1,N3)
   20    CONTINUE
   40 CONTINUE
      RETURN
      END
*
      SUBROUTINE WRITXY(NOUT,X,Y,N1,N2,N3)
*     Print 3-dimensional complex data
*     .. Scalar Arguments ..
      INTEGER           N1, N2, N3, NOUT
*     .. Array Arguments ..
      real              X(N1,N2,N3), Y(N1,N2,N3)
*     .. Local Scalars ..
      INTEGER           I, J, K
*     .. Executable Statements ..
      DO 40 I = 1, N1
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'z(i,j,k) for i =', I
         DO 20 J = 1, N2
            WRITE (NOUT,*)
            WRITE (NOUT,99999) 'Real ', (X(I,J,K),K=1,N3)
            WRITE (NOUT,99999) 'Imag ', (Y(I,J,K),K=1,N3)
   20    CONTINUE
   40 CONTINUE
      RETURN
*
99999 FORMAT (1X,A,7F10.3,/(6X,7F10.3))
99998 FORMAT (1X,A,I6)
      END
```

9.2 Program Data

```
C06FXF Example Program Data
2 3 4  : values of N1, N2, N3
     1.000     0.999     0.987     0.936   :  X(0,0,J), J=0,...,N3-1
     0.000    -0.040    -0.159    -0.352   :  Y(0,0,J), J=0,...,N3-1
     0.994     0.989     0.963     0.891   :  X(0,1,J), J=0,...,N3-1
    -0.111    -0.151    -0.268    -0.454   :  Y(0,1,J), J=0,...,N3-1
     0.903     0.885     0.823     0.694   :  X(0,2,J), J=0,...,N3-1
    -0.430    -0.466    -0.568    -0.720   :  Y(0,2,J), J=0,...,N3-1
     0.500     0.499     0.487     0.436   :  X(1,0,J), J=0,...,N3-1
     0.500     0.040     0.159     0.352   :  Y(1,0,J), J=0,...,N3-1
     0.494     0.489     0.463     0.391   :  X(1,1,J), J=0,...,N3-1
     0.111     0.151     0.268     0.454   :  Y(1,1,J), J=0,...,N3-1
     0.403     0.385     0.323     0.194   :  X(1,2,J), J=0,...,N3-1
     0.430     0.466     0.568     0.720   :  Y(1,2,J), J=0,...,N3-1
```

9.3 Program Results

```
 C06FXF Example Program Results

 Original data values

 z(i,j,k) for i =     1

 Real      1.000     0.999     0.987     0.936
 Imag      0.000    -0.040    -0.159    -0.352

 Real      0.994     0.989     0.963     0.891
 Imag     -0.111    -0.151    -0.268    -0.454

 Real      0.903     0.885     0.823     0.694
 Imag     -0.430    -0.466    -0.568    -0.720

 z(i,j,k) for i =     2

 Real      0.500     0.499     0.487     0.436
 Imag      0.500     0.040     0.159     0.352

 Real      0.494     0.489     0.463     0.391
```

```
 Imag      0.111     0.151     0.268     0.454

 Real      0.403     0.385     0.323     0.194
 Imag      0.430     0.466     0.568     0.720

 Components of discrete Fourier transform

 z(i,j,k) for i =     1

 Real      3.292     0.051     0.113     0.051
 Imag      0.102    -0.042     0.102     0.246

 Real      0.143     0.016    -0.024    -0.050
 Imag     -0.086     0.153     0.127     0.086

 Real      0.143    -0.050    -0.024     0.016
 Imag      0.290     0.118     0.077     0.051

 z(i,j,k) for i =     2

 Real      1.225     0.355    -0.000    -0.355
 Imag     -1.620     0.083     0.162     0.083

 Real      0.424     0.020     0.013    -0.007
 Imag      0.320    -0.115    -0.091    -0.080

 Real     -0.424     0.007    -0.013    -0.020
 Imag      0.320    -0.080    -0.091    -0.115

 Original sequence as restored by inverse transform

 z(i,j,k) for i =     1

 Real      1.000     0.999     0.987     0.936
 Imag      0.000    -0.040    -0.159    -0.352

 Real      0.994     0.989     0.963     0.891
 Imag     -0.111    -0.151    -0.268    -0.454

 Real      0.903     0.885     0.823     0.694
 Imag     -0.430    -0.466    -0.568    -0.720

 z(i,j,k) for i =     2

 Real      0.500     0.499     0.487     0.436
 Imag      0.500     0.040     0.159     0.352

 Real      0.494     0.489     0.463     0.391
 Imag      0.111     0.151     0.268     0.454

 Real      0.403     0.385     0.323     0.194
 Imag      0.430     0.466     0.568     0.720
```