NAG Fortran Library Routine Document

C06FUF

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

C06FUF computes the two-dimensional discrete Fourier transform of a bivariate sequence of complex data values. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
SUBROUTINE C06FUF(M, N, X, Y, INIT, TRIGM, TRIGN, WORK, IFAIL)
INTEGER           M, N, IFAIL
real              X(M*N), Y(M*N), TRIGM(2*M), TRIGN(2*N), WORK(2*M*N)
CHARACTER*1       INIT
```

3 Description

This routine computes the two-dimensional discrete Fourier transform 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 discrete Fourier transform is here defined by:

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

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.)

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 in 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: M – INTEGER *Input*

On entry: the number of rows, m, of the arrays X and Y.

Constraint: M ≥ 1.

2: N – INTEGER *Input*

On entry: the number of columns, n, of the arrays X and Y.

Constraint: N ≥ 1.

3: X(M∗N) – ***real*** array *Input/Output*
4: Y(M∗N) – ***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 two-dimensional arrays of dimension $(0 : \mathrm{M}-1,\ 0 : \mathrm{N}-1)$, then $\mathrm{X}(j_1, j_2)$ and $\mathrm{Y}(j_1, j_2)$ must contain the real and imaginary parts of $z_{j_1 j_2}$.

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

5: 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 TRIGM and TRIGN, 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 m and n are supplied in the arrays TRIGM and TRIGN, having been calculated in a previous call to the routine.

If INIT contains 'R', (**R**estart), then the routine assumes that trigonometric coefficients for the particular values of m and n are supplied in the arrays TRIGM and TRIGN, but does not check that the routine has previously been called. This option allows the TRIGM and TRIGN 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 m and n are compatible with the arrays TRIGM and TRIGN.

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

6: TRIGM(2∗M) – ***real*** array *Input/Output*
7: TRIGN(2∗N) – ***real*** array *Input/Output*

On entry: if INIT = 'S' or 'R', TRIGM and TRIGN must contain the required coefficients calculated in a previous call of the routine. Otherwise TRIGM and TRIGN need not be set.

If $m = n$ the same array may be supplied for TRIGM and TRIGN.

On exit: TRIGM and TRIGN contain the required coefficients (computed by the routine if INIT = 'I').

8: WORK(2∗M∗N) – ***real*** array *Workspace*

9: 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, M < 1.

IFAIL = 2

On entry, N < 1.

IFAIL = 3

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

IFAIL = 4

Not used at this Mark.

IFAIL = 5

On entry, On entry, INIT = 'S' or 'R', but at least one of the arrays TRIGM and TRIGN is inconsistent with the current value of M or N.

IFAIL = 6

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 $mn \times \log(mn)$, but also depends on the factorization of the individual dimensions m and n. 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 bivariate sequence of complex data values and prints the two-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.

```
*     C06FUF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, MNMAX
      PARAMETER        (MMAX=96,NMAX=96,MNMAX=MMAX*NMAX)
*     .. Local Scalars ..
      INTEGER          IFAIL, M, N
*     .. Local Arrays ..
      real             TRIGM(2*MMAX), TRIGN(2*NMAX), WORK(2*MNMAX),
     +                 X(MNMAX), Y(MNMAX)
*     .. External Subroutines ..
      EXTERNAL         C06FUF, C06GCF, READXY, WRITXY
*     .. Executable Statements ..
      WRITE (NOUT,*) 'C06FUF Example Program Results'
*     Skip heading in data Ûle
      READ (NIN,*)
   20 READ (NIN,*,END=40) M, N
      IF (M*N.GE.1 .AND. M*N.LE.MNMAX) THEN
         CALL READXY(NIN,X,Y,M,N)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original data values'
```

```
         CALL WRITXY(NOUT,X,Y,M,N)
         IFAIL = 0
*
*         Compute transform
         CALL C06FUF(M,N,X,Y,'Initial',TRIGM,TRIGN,WORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Components of discrete Fourier transform'
         CALL WRITXY(NOUT,X,Y,M,N)
*
*         Compute inverse transform
         CALL C06GCF(Y,M*N,IFAIL)
         CALL C06FUF(M,N,X,Y,'Subsequent',TRIGM,TRIGN,WORK,IFAIL)
         CALL C06GCF(Y,M*N,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     'Original sequence as restored by inverse transform'
         CALL WRITXY(NOUT,X,Y,M,N)
         GO TO 20
      ELSE
         WRITE (NOUT,*) ' ** Invalid value of M or N'
      END IF
   40 STOP
      END
*
      SUBROUTINE READXY(NIN,X,Y,N1,N2)
*     Read 2-dimensional complex data
*     .. Scalar Arguments ..
      INTEGER           N1, N2, NIN
*     .. Array Arguments ..
      real              X(N1,N2), Y(N1,N2)
*     .. Local Scalars ..
      INTEGER           I, J
*     .. Executable Statements ..
      DO 20 I = 1, N1
         READ (NIN,*) (X(I,J),J=1,N2)
         READ (NIN,*) (Y(I,J),J=1,N2)
   20 CONTINUE
      RETURN
      END
*
      SUBROUTINE WRITXY(NOUT,X,Y,N1,N2)
*     Print 2-dimensional complex data
*     .. Scalar Arguments ..
      INTEGER           N1, N2, NOUT
*     .. Array Arguments ..
      real              X(N1,N2), Y(N1,N2)
*     .. Local Scalars ..
      INTEGER           I, J
*     .. Executable Statements ..
      DO 20 I = 1, N1
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Real ', (X(I,J),J=1,N2)
         WRITE (NOUT,99999) 'Imag ', (Y(I,J),J=1,N2)
   20 CONTINUE
      RETURN
*
99999 FORMAT (1X,A,7F10.3,/(6X,7F10.3))
      END
```

9.2 Program Data

```
C06FUF Example Program Data
3  5  : Number of rows, M, and columns, N, in X and Y
     1.000     0.999     0.987     0.936     0.802  :  X(0,J), J=0,...,N-1
     0.000    -0.040    -0.159    -0.352    -0.597  :  Y(0,J), J=0,...,N-1
     0.994     0.989     0.963     0.891     0.731  :  X(1,J), J=0,...,N-1
    -0.111    -0.151    -0.268    -0.454    -0.682  :  Y(1,J), J=0,...,N-1
     0.903     0.885     0.823     0.694     0.467  :  X(2,J), J=0,...,N-1
```

```
   -0.430    -0.466    -0.568    -0.720    -0.884  :  Y(2,J), J=0,...,N-1
```

9.3 Program Results

```
 C06FUF Example Program Results

 Original data values

 Real      1.000     0.999     0.987     0.936     0.802
 Imag      0.000    -0.040    -0.159    -0.352    -0.597

 Real      0.994     0.989     0.963     0.891     0.731
 Imag     -0.111    -0.151    -0.268    -0.454    -0.682

 Real      0.903     0.885     0.823     0.694     0.467
 Imag     -0.430    -0.466    -0.568    -0.720    -0.884

 Components of discrete Fourier transform

 Real      3.373     0.481     0.251     0.054    -0.419
 Imag     -1.519    -0.091     0.178     0.319     0.415

 Real      0.457     0.055     0.009    -0.022    -0.076
 Imag      0.137     0.032     0.039     0.036     0.004

 Real     -0.170    -0.037    -0.042    -0.038    -0.002
 Imag      0.493     0.058     0.008    -0.025    -0.083

 Original sequence as restored by inverse transform

 Real      1.000     0.999     0.987     0.936     0.802
 Imag     -0.000    -0.040    -0.159    -0.352    -0.597

 Real      0.994     0.989     0.963     0.891     0.731
 Imag     -0.111    -0.151    -0.268    -0.454    -0.682

 Real      0.903     0.885     0.823     0.694     0.467
 Imag     -0.430    -0.466    -0.568    -0.720    -0.884
```