NAG Library Routine Document

C06FFF

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

C06FFF computes the discrete Fourier transform of one variable in a multivariate sequence of complex data values.

2 Specification

```
SUBROUTINE C06FFF (NDIM, L, ND, N, X, Y, WORK, LWORK, IFAIL)

INTEGER            NDIM, L, ND(NDIM), N, LWORK, IFAIL
REAL (KIND=nag_wp) X(N), Y(N), WORK(LWORK)
```

3 Description

C06FFF computes the discrete Fourier transform of one variable (the lth say) in a multivariate sequence of complex data values $z_{j_1 j_2 \ldots j_m}$, for $j_1 = 0, 1, \ldots, n_1 - 1$ and $j_2 = 0, 1, \ldots, n_2 - 1$, and so on. Thus the individual dimensions are $n_1, n_2, \ldots, n_m$, and the total number of data values is $n = n_1 \times n_2 \times \cdots \times n_m$.

The routine computes n/n_l one-dimensional transforms defined by

$$\hat{z}_{j_1 \ldots k_l \ldots j_m} = \frac{1}{\sqrt{n_l}} \sum_{j_l=0}^{n_l-1} z_{j_1 \ldots j_l \ldots j_m} \times \exp\left(-\frac{2\pi i j_l k_l}{n_l}\right),$$

where $k_l = 0, 1, \ldots, n_l - 1$.

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

To compute the inverse discrete Fourier transforms, defined with $\exp\left(+\frac{2\pi i j_l k_l}{n_l}\right)$ in the above formula instead of $\exp\left(-\frac{2\pi i j_l k_l}{n_l}\right)$, this routine should be preceded and followed by the complex conjugation of the data values and the transform (by negating the imaginary parts stored in y).

The data values must be supplied in a pair of one-dimensional arrays (real and imaginary parts separately), in accordance with the Fortran convention for storing multidimensional data (i.e., with the first subscript j_1 varying most rapidly).

This routine calls C06FCF to perform one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm in Brigham (1974), and hence there are some restrictions on the values of n_l (see Section 5).

4 References

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

5 Parameters

1: NDIM – INTEGER *Input*

On entry: m, the number of dimensions (or variables) in the multivariate data.

Constraint: NDIM ≥ 1.

2: L – INTEGER *Input*

On entry: l, the index of the variable (or dimension) on which the discrete Fourier transform is to be performed.

Constraint: $1 \leq \mathrm{L} \leq \mathrm{NDIM}$.

3: ND(NDIM) – INTEGER array *Input*

On entry: ND(i) must contain n_i (the dimension of the ith variable) , for $i = 1, 2, \ldots, m$. The largest prime factor of ND(l) must not exceed 19, and the total number of prime factors of ND(l), counting repetitions, must not exceed 20.

Constraint: $\mathrm{ND}(i) \geq 1$, for $i = 1, 2, \ldots, \mathrm{NDIM}$.

4: N – INTEGER *Input*

On entry: n, the total number of data values.

Constraint: $\mathrm{N} = \mathrm{ND}(1) \times \mathrm{ND}(2) \times \cdots \times \mathrm{ND}(\mathrm{NDIM})$.

5: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: $\mathrm{X}(1 + j_1 + n_1 j_2 + n_1 n_2 j_3 + \ldots)$ must contain the real part of the complex data value $z_{j_1 j_2 \cdots j_m}$, for $0 \leq j_1 \leq n_1 - 1, 0 \leq j_2 \leq n_2 - 1, \ldots$; i.e., the values are stored in consecutive elements of the array according to the Fortran convention for storing multidimensional arrays.

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

6: Y(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the imaginary parts of the complex data values, stored in the same way as the real parts in the array X.

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

7: WORK(LWORK) – REAL (KIND=nag_wp) array *Workspace*

8: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which C06FFF is called.

Constraint: $\mathrm{LWORK} \geq 3 \times \mathrm{ND}(\mathrm{L})$.

9: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

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, if you are 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.**

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

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, NDIM < 1.

IFAIL $= 2$

On entry, N $\neq$ ND(1) $\times$ ND(2) $\times \cdots \times$ ND(NDIM).

IFAIL $= 3$

On entry, L < 1 or L $>$ NDIM.

IFAIL $= 10 \times l + 1$

At least one of the prime factors of ND(l) is greater than 19.

IFAIL $= 10 \times l + 2$

ND(l) has more than 20 prime factors.

IFAIL $= 10 \times l + 3$

On entry, ND(l) < 1.

IFAIL $= 10 \times l + 4$

On entry, LWORK $< 3 \times$ ND(l).

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 is approximately proportional to $n \times \log n_l$, but also depends on the factorization of n_l. C06FFF is faster if the only prime factors of n_l are 2, 3 or 5; and fastest of all if n_l is a power of 2.

9 Example

This example reads in a bivariate sequence of complex data values and prints the discrete Fourier transform of the second variable. It then performs an inverse transform and prints the sequence so obtained, which may be compared with the original data values.

9.1 Program Text

```
!   C06FFF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module c06fffe_mod

!     C06FFF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
```

```
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: nin = 5, nout = 6
    Contains
      Subroutine readxy(nin,x,y,n1,n2)
!       Read 2-dimensional complex data

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: n1, n2, nin
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: x(n1,n2), y(n1,n2)
!       .. Local Scalars ..
        Integer                              :: i, j
!       .. Executable Statements ..
        Do i = 1, n1
          Read (nin,*)(x(i,j),j=1,n2)
          Read (nin,*)(y(i,j),j=1,n2)
        End Do
        Return
      End Subroutine readxy

      Subroutine writxy(nout,x,y,n1,n2)
!       Print 2-dimensional complex data

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: n1, n2, nout
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: x(n1,n2), y(n1,n2)
!       .. Local Scalars ..
        Integer                              :: i, j
!       .. Executable Statements ..
        Do 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)
        End Do
        Return

99999   Format (1X,A,7F10.3/(6X,7F10.3))
      End Subroutine writxy
    End Module c06fffe_mod

    Program c06fffe

!     C06FFF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c06fff, c06gcf, nag_wp
      Use c06fffe_mod, Only: nin, nout, readxy, writxy
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                              :: ieof, ifail, l, lwork, n, ndim
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: work(:), x(:), y(:)
      Integer, Allocatable                 :: nd(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: product
!     .. Executable Statements ..
      Write (nout,*) 'C06FFF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) ndim
        If (ieof<0) Exit loop

        Allocate (nd(ndim))

        Read (nin,*) nd(1:ndim), l

        n = product(nd(1:ndim))
        lwork = 3*nd(l)
        Allocate (x(n),y(n),work(lwork))

        Call readxy(nin,x,y,nd(1),nd(2))

        Write (nout,*)
        Write (nout,*) 'Original data'
        Call writxy(nout,x,y,nd(1),nd(2))

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft

!       Compute transform
        ifail = 0
        Call c06fff(ndim,l,nd,n,x,y,work,lwork,ifail)

        Write (nout,*)
        Write (nout,99999) 'Discrete Fourier transform of variable ', l
        Call writxy(nout,x,y,nd(1),nd(2))

!       Compute inverse transform
        Call c06gcf(y,n,ifail)
        Call c06fff(ndim,l,nd,n,x,y,work,lwork,ifail)
        Call c06gcf(y,n,ifail)

        Write (nout,*)
        Write (nout,*) 'Original sequence as restored by inverse transform'
        Call writxy(nout,x,y,nd(1),nd(2))
        Deallocate (x,y,work,nd)
      End Do loop

99999 Format (1X,A,I1)
    End Program c06fffe
```

9.2 Program Data

```
C06FFF Example Program Data
    2                                                     : ndim
    3    5    2                                           : nd(1), nd(2), l
     1.000     0.999     0.987     0.936     0.802
     0.000    -0.040    -0.159    -0.352    -0.597
     0.994     0.989     0.963     0.891     0.731
    -0.111    -0.151    -0.268    -0.454    -0.682
     0.903     0.885     0.823     0.694     0.467
    -0.430    -0.466    -0.568    -0.720    -0.884      : x, y
```

9.3 Program Results

```
 C06FFF Example Program Results

 Original data

 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

 Discrete Fourier transform of variable 2

 Real      2.113     0.288     0.126    -0.003    -0.287
 Imag     -0.513    -0.000     0.130     0.190     0.194

 Real      2.043     0.286     0.139     0.018    -0.263
 Imag     -0.745    -0.032     0.115     0.189     0.225

 Real      1.687     0.260     0.170     0.079    -0.176
```

```
Imag     -1.372    -0.125     0.063     0.173     0.299

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
```