

## C06PFF – NAG Fortran Library Routine Document

**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

C06PFF computes the discrete Fourier transform of one variable in a multivariate sequence of complex data values.

## 2 Specification

```
SUBROUTINE C06PFF(DIRECT, NDIM, L, ND, N, X, WORK, LWORK, IFAIL)
CHARACTER*1      DIRECT
INTEGER         NDIM, L, ND(NDIM), N, LWORK, IFAIL
complex       X(N), WORK(LWORK)
```

## 3 Description

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

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

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

where  $k_l = 0, 1, \dots, n_l - 1$ . The plus or minus sign in the argument of the exponential terms in the above definition determine the direction of the transform: a minus sign defines the **forward** direction and a plus sign defines the **backward** direction.

(Note the scale factor of  $\frac{1}{\sqrt{n_l}}$  in this definition.) A call of the routine with DIRECT = 'F' followed by a call with DIRECT = 'B' will restore the original data.

The data values must be supplied in a one-dimensional complex array in accordance with the Fortran convention for storing multi-dimensional data (i.e., with the first subscript  $j_1$  varying most rapidly).

This routine calls C06PRF to perform one-dimensional discrete Fourier transforms. Hence, the routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [2].

## 4 References

- [1] Brigham E O (1973) *The Fast Fourier Transform* Prentice–Hall
- [2] Temperton C (1983) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

## 5 Parameters

- 1: DIRECT — CHARACTER\*1 *Input*  
*On entry:* if the **Forward** transform as defined in Section 3 is to be computed, then DIRECT must be set equal to 'F'. If the **Backward** transform is to be computed then DIRECT must be set equal to 'B'.  
*Constraint:* DIRECT = 'F' or 'B'.
- 2: NDIM — INTEGER *Input*  
*On entry:* the number of dimensions (or variables) in the multivariate data,  $m$ .  
*Constraint:* NDIM  $\geq$  1.

- 3:** L — INTEGER *Input*  
*On entry:* the index of the variable (or dimension) on which the discrete Fourier transform is to be performed,  $l$ .  
*Constraint:*  $1 \leq L \leq \text{NDIM}$ .
- 4:** ND(NDIM) — INTEGER array *Input*  
*On entry:* ND( $i$ ) must contain  $n_i$  (the dimension of the  $i$ th variable), for  $i = 1, 2, \dots, m$ . The total number of prime factors of ND( $l$ ), counting repetitions, must not exceed 30.  
*Constraint:* ND( $i$ )  $\geq 1$  for all  $i$ .
- 5:** N — INTEGER *Input*  
*On entry:* the total number of data values,  $n$ .  
*Constraint:*  $N = \text{ND}(1) \times \text{ND}(2) \times \dots \times \text{ND}(\text{NDIM})$ .
- 6:** X(N) — *complex* array *Input/Output*  
*On entry:* X( $1 + j_1 + n_1 j_2 + n_1 n_2 j_3 + \dots$ ) must contain the complex data value  $z_{j_1 j_2 \dots j_m}$ , for  $0 \leq j_1 < n_1$  and  $0 \leq j_2 < n_2, \dots$ ; i.e., the values are stored in consecutive elements of the array according to the Fortran convention for storing multi-dimensional arrays.  
*On exit:* the corresponding elements of the computed transform.
- 7:** WORK(LWORK) — *complex* array *Workspace*  
The workspace requirements as documented for this routine may be an overestimate in some implementations. For full details of the workspace required by this routine please refer to the Users' Note for your implementation.  
*On exit:* the real part of WORK(1) contains the minimum workspace required for the current value of N with this implementation.
- 8:** LWORK — INTEGER *Input*  
*On entry:* the dimension of the array WORK as declared in the (sub)program from which C06PFF is called.  
*Constraint:* LWORK  $\geq N + \text{ND}(L) + 15$ .
- 9:** IFAIL — INTEGER *Input/Output*  
*On entry:* IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.  
*On exit:* IFAIL = 0 unless the routine detects an error (see Section 6).

## 6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NDIM < 1.

IFAIL = 2

On entry, L < 1 or L > NDIM.

IFAIL = 3

On entry, DIRECT not equal to one of 'F' or 'B'.

IFAIL = 4

On entry, at least one of  $ND(i) < 1$  for some  $i$ .

IFAIL = 5

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

IFAIL = 6

On entry, LWORK is too small. The minimum amount of workspace required is returned in WORK(1).

IFAIL = 7

On entry, ND(L) has more than 30 prime factors.

IFAIL = 8

An unexpected error has occurred in an internal call. Check all subroutine calls and array dimensions. Seek expert help.

## 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  $n \times \log n_l$ , but also depends on the factorization of  $n_l$ . The routine is somewhat faster than average 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 program 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

```

*      C06PFF Example Program Text.
*      Mark 19 Release. NAG Copyright 1999.
*      .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER       (NIN=5,NOUT=6)
      INTEGER          NDIM, NMAX, LWORK
      PARAMETER       (NDIM=2,NMAX=96,LWORK=2*NMAX+15)
*      .. Local Scalars ..
      INTEGER          IFAIL, L, N
*      .. Local Arrays ..
      complex        WORK(LWORK), X(NMAX)
      INTEGER          ND(NDIM)
*      .. External Subroutines ..
      EXTERNAL        C06PFF, READX, WRITX
*      .. Executable Statements ..
      WRITE (NOUT,*) 'C06PFF Example Program Results'
*      Skip heading in data file
      READ (NIN,*)
20  CONTINUE

```

```

READ (NIN,*,END=40) ND(1), ND(2), L
N = ND(1)*ND(2)
IF (N.GE.1 .AND. N.LE.NMAX) THEN
  CALL READX(NIN,X,ND(1),ND(2))
  WRITE (NOUT,*)
  WRITE (NOUT,*) 'Original data'
  CALL WRITX(NOUT,X,ND(1),ND(2))
  IFAIL = 0
*
*   Compute transform
  CALL C06PFF('F',NDIM,L,ND,N,X,WORK,LWORK,IFAIL)
*
  WRITE (NOUT,*)
  WRITE (NOUT,99999) 'Discrete Fourier transform of variable ', L
  CALL WRITX(NOUT,X,ND(1),ND(2))
*
*   Compute inverse transform
  CALL C06PFF('B',NDIM,L,ND,N,X,WORK,LWORK,IFAIL)
*
  WRITE (NOUT,*)
  WRITE (NOUT,*)
+   'Original sequence as restored by inverse transform'
  CALL WRITX(NOUT,X,ND(1),ND(2))
  GO TO 20
ELSE
  WRITE (NOUT,*) 'Invalid value of N'
END IF
40 CONTINUE
STOP
*
99999 FORMAT (1X,A,I1)
END
*
SUBROUTINE READX(NIN,X,N1,N2)
*   Read 2-dimensional complex data
*   .. Scalar Arguments ..
INTEGER          N1, N2, NIN
*   .. Array Arguments ..
complex         X(N1,N2)
*   .. Local Scalars ..
INTEGER          I, J
*   .. Executable Statements ..
DO 20 I = 1, N1
  READ (NIN,*) (X(I,J),J=1,N2)
20 CONTINUE
RETURN
END
*
SUBROUTINE WRITX(NOUT,X,N1,N2)
*   Print 2-dimensional complex data
*   .. Scalar Arguments ..
INTEGER          N1, N2, NOUT
*   .. Array Arguments ..
complex         X(N1,N2)
*   .. Local Scalars ..
INTEGER          I, J
*   .. Executable Statements ..
DO 20 I = 1, N1

```

```

        WRITE (NOUT,*)
        WRITE (NOUT,99999) (X(I,J),J=1,N2)
20 CONTINUE
    RETURN
*
99999 FORMAT (1X,7(:1x,'(',F6.3,',',F6.3,')'))
END

```

## 9.2 Program Data

C06PFF Example Program Data

```

3      5      2
(1.000,0.000)
(0.999,-0.040)
(0.987,-0.159)
(0.936,-0.352)
(0.802,-0.597)
(0.994,-0.111)
(0.989,-0.151)
(0.963,-0.268)
(0.891,-0.454)
(0.731,-0.682)
(0.903,-0.430)
(0.885,-0.466)
(0.823,-0.568)
(0.694,-0.720)
(0.467,-0.884)

```

## 9.3 Program Results

C06PFF Example Program Results

Original data

```

( 1.000, 0.000) ( 0.999,-0.040) ( 0.987,-0.159) ( 0.936,-0.352) ( 0.802,-0.597)
( 0.994,-0.111) ( 0.989,-0.151) ( 0.963,-0.268) ( 0.891,-0.454) ( 0.731,-0.682)
( 0.903,-0.430) ( 0.885,-0.466) ( 0.823,-0.568) ( 0.694,-0.720) ( 0.467,-0.884)

```

Discrete Fourier transform of variable 2

```

( 2.113,-0.513) ( 0.288, 0.000) ( 0.126, 0.130) (-0.003, 0.190) (-0.287, 0.194)
( 2.043,-0.745) ( 0.286,-0.032) ( 0.139, 0.115) ( 0.018, 0.189) (-0.263, 0.225)
( 1.687,-1.372) ( 0.260,-0.125) ( 0.170, 0.063) ( 0.079, 0.173) (-0.176, 0.299)

```

Original sequence as restored by inverse transform

```

( 1.000, 0.000) ( 0.999,-0.040) ( 0.987,-0.159) ( 0.936,-0.352) ( 0.802,-0.597)
( 0.994,-0.111) ( 0.989,-0.151) ( 0.963,-0.268) ( 0.891,-0.454) ( 0.731,-0.682)
( 0.903,-0.430) ( 0.885,-0.466) ( 0.823,-0.568) ( 0.694,-0.720) ( 0.467,-0.884)

```