

## C06HDF – 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

C06HDF computes the discrete quarter-wave Fourier cosine transforms of  $m$  sequences of real data values. This routine is designed to be particularly efficient on vector processors.

## 2 Specification

```
SUBROUTINE C06HDF(DIRECT, M, N, X, INIT, TRIG, WORK, IFAIL)
  INTEGER          M, N, IFAIL
  real            X(M*N), TRIG(2*N), WORK(M*N)
  CHARACTER*1      DIRECT, INIT
```

## 3 Description

Given  $m$  sequences of  $n$  real data values  $x_j^p$ , for  $j = 0, 1, \dots, n-1$ ;  $p = 1, 2, \dots, m$ , this routine simultaneously calculates the quarter-wave Fourier cosine transforms of all the sequences defined by:

$$\hat{x}_k^p = \frac{1}{\sqrt{n}} \left\{ \frac{1}{2} x_0^p + \sum_{j=1}^{n-1} x_j^p \times \cos \left( j(2k-1) \frac{\pi}{2n} \right) \right\}, \quad \text{if DIRECT = 'F' or 'f'}$$

or its inverse

$$x_k^p = \frac{2}{\sqrt{n}} \sum_{j=0}^{n-1} \hat{x}_j^p \times \cos \left( (2j-1)k \frac{\pi}{2n} \right), \quad \text{if DIRECT = 'B' or 'b'}$$

for  $k = 0, 1, \dots, n-1$ ;  $p = 1, 2, \dots, m$ .

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

A call of the routine with DIRECT = 'F' followed by a call with DIRECT = 'B' will restore the original data.

The transform calculated by this routine can be used to solve Poisson's equation when the derivative of the solution is specified at the left boundary, and the solution is specified at the right boundary (Swarztrauber [2]). (See the Chapter Introduction.)

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, described in Temperton [4], together with pre- and post-processing stages described in Swarztrauber [3]. Special coding is provided for the factors 2, 3, 4, 5 and 6. This routine is designed to be particularly efficient on vector processors, and it becomes especially fast as  $m$ , the number of transforms to be computed in parallel, increases.

## 4 References

- [1] Brigham E O (1973) *The Fast Fourier Transform* Prentice-Hall
- [2] Swarztrauber P N (1977) The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle *SIAM Rev.* **19** (3) 490–501
- [3] Swarztrauber P N (1982) Vectorizing the FFT's *Parallel Computation* (ed G Rodrigue) Academic Press 51–83
- [4] Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

## 5 Parameters

- 1: DIRECT — CHARACTER\*1 Input

*On entry:* if the **F**orward transform as defined in Section 3 is to be computed, then DIRECT must be set equal to 'F'. If the **B**ackward transform is to be computed, that is the inverse, then DIRECT must be set equal to 'B'.

*Constraint:* DIRECT = 'F' or 'B'.

- 2: M — INTEGER Input

*On entry:* the number of sequences to be transformed,  $m$ .

*Constraint:*  $M \geq 1$ .

- 3: N — INTEGER Input

*On entry:* the number of real values in each sequence,  $n$ .

*Constraint:*  $N \geq 1$ .

- 4: X(M\*N) — *real* array Input/Output

*On entry:* the data must be stored in X as if in a two-dimensional array of dimension (1:M,0:N-1); each of the  $m$  sequences is stored in a **row** of the array. In other words, if the data values of the  $p$ th sequence to be transformed are denoted by  $x_j^p$ , for  $j = 0, 1, \dots, n-1$ ;  $p = 1, 2, \dots, m$ , then the  $mn$  elements of the array X must contain the values

$$x_0^1, x_0^2, \dots, x_0^m, x_1^1, x_1^2, \dots, x_1^m, \dots, x_{n-1}^1, x_{n-1}^2, \dots, x_{n-1}^m.$$

*On exit:* the  $m$  quarter-wave cosine transforms stored as if in a two-dimensional array of dimension (1:M,0:N-1). Each of the  $m$  transforms is stored in a **row** of the array, overwriting the corresponding original sequence. If the  $n$  components of the  $p$ th quarter-wave cosine transform are denoted by  $\hat{x}_k^p$ , for  $k = 0, 1, \dots, n-1$ ;  $p = 1, 2, \dots, m$ , then the  $mn$  elements of the array X contain the values

$$\hat{x}_0^1, \hat{x}_0^2, \dots, \hat{x}_0^m, \hat{x}_1^1, \hat{x}_1^2, \dots, \hat{x}_1^m, \dots, \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \dots, \hat{x}_{n-1}^m.$$

- 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 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 value of  $n$  are supplied in the array TRIG, having been calculated in a previous call to one of C06HAF, C06HBF, C06HCF or C06HDF.

If INIT contains 'R' (**R**estart), then the routine assumes that trigonometric coefficients for the particular value of  $n$  are supplied in the array TRIG, but does not check that C06HAF, C06HBF, C06HCF or C06HDF have previously been called. This option allows the TRIG array 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 value of  $n$  is consistent with the array TRIG.

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

- 6: TRIG(2\*N) — *real* array Input/Output

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

*On exit:* TRIG contains the required coefficients (computed by the routine if INIT = 'I').

- 7: WORK(M\*N) — *real* array Workspace

**8: 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**

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

Errors 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 'T', 'S' or 'R'.

IFAIL = 4

Not used at this Mark.

IFAIL = 5

On entry, INIT =, 'S' or 'R', but the array TRIG and the current value of N are inconsistent.

IFAIL = 6

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

IFAIL = 7

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  $nm \times \log n$ , but also depends on the factors of  $n$ . The routine is fastest if the only prime factors of  $n$  are 2, 3 and 5, and is particularly slow if  $n$  is a large prime, or has large prime factors.

**9 Example**

This program reads in sequences of real data values and prints their quarter-wave cosine transforms as computed by C06HDF with DIRECT = 'F'. It then calls the routine again with DIRECT = or 'B' and prints the results which may be compared with the original data.

## 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.

```

*      C06HDF Example Program Text
*      Mark 14 Revised.  NAG Copyright 1989.
*      .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX
      PARAMETER        (MMAX=5,NMAX=20)
*      .. Local Scalars ..
      INTEGER          I, IFAIL, J, M, N
*      .. Local Arrays ..
      real            TRIG(2*NMAX), WORK(MMAX*NMAX), X(NMAX*MMAX)
*      .. External Subroutines ..
      EXTERNAL         C06HDF
*      .. Executable Statements ..
      WRITE (NOUT,*) 'C06HDF Example Program Results'
*      Skip heading in data file
      READ (NIN,*)
20     READ (NIN,*,END=120) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
          DO 40 J = 1, M
              READ (NIN,*) (X(I*M+J),I=0,N-1)
40     CONTINUE
          WRITE (NOUT,*)
          WRITE (NOUT,*) 'Original data values'
          WRITE (NOUT,*)
          DO 60 J = 1, M
              WRITE (NOUT,99999) (X(I*M+J),I=0,N-1)
60     CONTINUE
          IFAIL = 0
*
*      -- Compute transform
          CALL C06HDF('Forward',M,N,X,'Initial',TRIG,WORK,IFAIL)
*
          WRITE (NOUT,*)
          WRITE (NOUT,*)
      +   'Discrete quarter-wave Fourier cosine transforms'
          WRITE (NOUT,*)
          DO 80 J = 1, M
              WRITE (NOUT,99999) (X(I*M+J),I=0,N-1)
80     CONTINUE
*
*      -- Compute inverse transform
          CALL C06HDF('Backward',M,N,X,'Subsequent',TRIG,WORK,IFAIL)
*
          WRITE (NOUT,*)
          WRITE (NOUT,*) 'Original data as restored by inverse transform'
          WRITE (NOUT,*)
          DO 100 J = 1, M
              WRITE (NOUT,99999) (X(I*M+J),I=0,N-1)
100    CONTINUE
          GO TO 20
      ELSE
          WRITE (NOUT,*) 'Invalid value of M or N'
      END IF

```

```

120 STOP
*
99999 FORMAT (6X,7F10.4)
END

```

## 9.2 Program Data

C06HDF Example Program Data

```

3 6 : Number of sequences, M, and number of values in each sequence, N
0.3854 0.6772 0.1138 0.6751 0.6362 0.1424 : X, sequence 1
0.5417 0.2983 0.1181 0.7255 0.8638 0.8723 : X, sequence 2
0.9172 0.0644 0.6037 0.6430 0.0428 0.4815 : X, sequence 3

```

## 9.3 Program Results

C06HDF Example Program Results

Original data values

0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815

Discrete quarter-wave Fourier cosine transforms

0.7257	-0.2216	0.1011	0.2355	-0.1406	-0.2282
0.7479	-0.6172	0.4112	0.0791	0.1331	-0.0906
0.6713	-0.1363	-0.0064	-0.0285	0.4758	0.1475

Original data as restored by inverse transform

0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815

---