# NAG Library Routine Document

# C06HBF

**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

C06HBF computes the discrete Fourier cosine transforms of $m$ sequences of real data values. This routine is designed to be particularly efficient on vector processors.

## 2 Specification

```
SUBROUTINE C06HBF (M, N, X, INIT, TRIG, WORK, IFAIL)

INTEGER              M, N, IFAIL
REAL (KIND=nag_wp) X(M*(N+1)), TRIG(2*N), WORK(M*N)
CHARACTER(1)         INIT
```

## 3 Description

Given $m$ sequences of $n+1$ real data values $x_j^p$, for $j=0,1,\ldots,n$ and $p=1,2,\ldots,m$, C06HBF simultaneously calculates the Fourier cosine transforms of all the sequences defined by

$$\hat{x}_k^p = \sqrt{\frac{2}{n}}\left\{\frac{1}{2}x_0^p + \sum_{j=1}^{n-1} x_j^p \times \cos\left(jk\frac{\pi}{n}\right) + \frac{1}{2}-1^k x_n^p\right\}, \quad k=0,1,\ldots,n \text{ and } p=1,2,\ldots,m.$$

(Note the scale factor $\sqrt{\frac{2}{n}}$ in this definition.)

The Fourier cosine transform is its own inverse and two calls of this routine with the same data 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 both left and right boundaries (see Swarztrauber (1977)). (See the C06 Chapter Introduction.)

The routine uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, described in Temperton (1983), together with pre- and post-processing stages described in Swarztrauber (1982). 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

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

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

Swarztrauber P N (1982) Vectorizing the FFT's *Parallel Computation* (ed G Rodrique) 51–83 Academic Press

Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

## 5 Parameters

1: M – INTEGER *Input*

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

*Constraint*: $\text{M} \geq 1$.

2: N – INTEGER *Input*

*On entry*: one less than the number of real values in each sequence, i.e., the number of values in each sequence is $n+1$.

*Constraint*: $\text{N} \geq 1$.

3: $\text{X}(\text{M} \times (\text{N}+1))$ – REAL (KIND=nag_wp) array *Input/Output*

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

$$x_0^1, x_0^2, \ldots, x_0^m, x_1^1, x_1^2, \ldots, x_1^m, \ldots, x_n^1, x_n^2, \ldots, x_n^m.$$

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

$$\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^m, \hat{x}_1^1, \hat{x}_1^2, \ldots, \hat{x}_1^m, \ldots, \hat{x}_n^1, \hat{x}_n^2, \ldots, \hat{x}_n^m.$$

4: INIT – CHARACTER(1) *Input*

*On entry*: indicates whether trigonometric coefficients are to be calculated.

INIT = 'I'
: Calculate the required trigonometric coefficients for the given value of $n$, and store in the array TRIG.

INIT = 'S' or 'R'
: The required trigonometric coefficients are assumed to have been calculated and stored in the array TRIG in a prior call to one of C06HAF, C06HBF, C06HCF or C06HDF. The routine performs a simple check that the current value of $n$ is consistent with the values stored in TRIG.

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

5: $\text{TRIG}(2 \times \text{N})$ – REAL (KIND=nag_wp) array *Input/Output*

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

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

6: $\text{WORK}(\text{M} \times \text{N})$ – REAL (KIND=nag_wp) array *Workspace*

7: 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*: 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, $\mathrm{M} < 1$.

IFAIL $= 2$

On entry, $\mathrm{N} < 1$.

IFAIL $= 3$

On entry, INIT $\neq$ 'I', '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 $= 6$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

## 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 Parallelism and Performance

C06HBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06HBF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

The time taken by C06HBF is approximately proportional to $nm\log(n)$, but also depends on the factors of $n$. C06HBF 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.

## 10 Example

This example reads in sequences of real data values and prints their Fourier cosine transforms (as computed by C06HBF). It then calls C06HBF again and prints the results which may be compared with the original sequence.

### 10.1 Program Text

```
    Program c06hbfe

!     C06HBF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: c06hbf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ieof, ifail, j, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: trig(:), work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06HBF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) m, n
        If (ieof<0) Exit loop

        Allocate (trig(2*n),work(m*n),x(m*(n+1)))
        Do j = 1, m
          Read (nin,*)(x(i*m+j),i=0,n)
        End Do
        Write (nout,*)
        Write (nout,*) 'Original data values'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
!       -- Compute transform
        Call c06hbf(m,n,x,'Initial',trig,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Discrete Fourier cosine transforms'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do

!       -- Compute inverse transform
        Call c06hbf(m,n,x,'Subsequent',trig,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Original data as restored by inverse transform'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do
        Deallocate (trig,work,x)
      End Do loop

99999 Format (6X,7F10.4)
    End Program c06hbfe
```

## 10.2 Program Data

```
C06HBF Example Program Data
 3  6                                                     : m, n
 0.3854  0.6772  0.1138  0.6751  0.6362  0.1424  0.9562
 0.5417  0.2983  0.1181  0.7255  0.8638  0.8723  0.4936
 0.9172  0.0644  0.6037  0.6430  0.0428  0.4815  0.2057  : x
```

## 10.3 Program Results

```
 C06HBF Example Program Results

 Original data values

          0.3854    0.6772    0.1138    0.6751    0.6362    0.1424    0.9562
          0.5417    0.2983    0.1181    0.7255    0.8638    0.8723    0.4936
          0.9172    0.0644    0.6037    0.6430    0.0428    0.4815    0.2057

 Discrete Fourier cosine transforms

          1.6833   -0.0482    0.0176    0.1368    0.3240   -0.5830   -0.0427
          1.9605   -0.4884   -0.0655    0.4444    0.0964    0.0856   -0.2289
          1.3838    0.1588   -0.0761   -0.1184    0.3512    0.5759    0.0110

 Original data as restored by inverse transform

          0.3854    0.6772    0.1138    0.6751    0.6362    0.1424    0.9562
          0.5417    0.2983    0.1181    0.7255    0.8638    0.8723    0.4936
          0.9172    0.0644    0.6037    0.6430    0.0428    0.4815    0.2057
```