NAG Fortran 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               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 $j = 0, 1, \ldots, n$; $p = 1, 2, \ldots, m$, this routine simultaneously calculates the Fourier cosine transforms of all the sequences defined by:

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

(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 (Swarztrauber (1977)). (See the C06 Chapter Introduction.)

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham (1974)) known as the Stockham self-sorting algorithm, described in Temperton (1983a), 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 (1983a) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

5 Parameters

1: M – INTEGER *Input*

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

Constraint: $\mathrm{M} \geq 1$.

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: $\mathrm{N} \geq 1$.

3: X(M*(N+1)) – ***real*** array *Input/Output*

On entry: the data must be stored in X as if in a two-dimensional array of dimension $(1:\mathrm{M},\ 0:\mathrm{N})$; each of the m sequences is stored in a **row** of the array. In other words, if the $(n+1)$ data values of the pth sequence to be transformed are denoted by x_j^p, for $j=0,1,\ldots,n$; $p=1,2,\ldots,m$, 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 $(1:\mathrm{M}, 0:\mathrm{N})$. 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 pth Fourier cosine transform are denoted by $\hat{x}_k^p$, for $k=0,1,\ldots,n$; $p=1,2,\ldots,m$, 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: 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'.

5: 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').

6: WORK(M*N) – ***real*** array *Workspace*

7: 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, INIT = 'S' or 'R', but the array TRIG and the current value of n are inconsistent.

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 $nm \times \log n$, 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 Fourier cosine transforms (as computed by C06HBF). It then calls the routine again and prints the results which may be compared with the original sequence.

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.

```
*     C06HBF 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+1)*MMAX)
*     .. External Subroutines ..
      EXTERNAL         C06HBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'C06HBF Example Program Results'
*     Skip heading in data Ûle
      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)
   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)
   60    CONTINUE
         IFAIL = 0
*
*         Compute transform
         CALL C06HBF(M,N,X,'Initial',TRIG,WORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Discrete Fourier cosine transforms'
         WRITE (NOUT,*)
         DO 80 J = 1, M
            WRITE (NOUT,99999) (X(I*M+J),I=0,N)
   80    CONTINUE
*
*         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 100 J = 1, M
            WRITE (NOUT,99999) (X(I*M+J),I=0,N)
  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

```
C06HBF Example Program Data
3  6 : Number of sequences, M,   (number of values in each sequence)-1, N
 0.3854  0.6772  0.1138  0.6751  0.6362  0.1424  0.9562 : X, sequence 1
 0.5417  0.2983  0.1181  0.7255  0.8638  0.8723  0.4936 : X, sequence 2
 0.9172  0.0644  0.6037  0.6430  0.0428  0.4815  0.2057 : X, sequence 3
```

9.3 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
```