# nag_fft_multiple_cosine (c06hbc)

## 1. Purpose

**nag_fft_multiple_cosine (c06hbc)** computes the discrete Fourier cosine transforms of $m$ sequences of real data values.

## 2. Specification

```
#include <nag.h>
#include <nagc06.h>

void nag_fft_multiple_cosine(Integer m, Integer n, double x[],
     double trig[], NagError *fail)
```

## 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 function simultaneously calculates the Fourier cosine transforms of all the sequences defined by:

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

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

The Fourier cosine transform defined above is its own inverse, and two consecutive calls of this function with the same data will restore the original data (but see Section 6.1).

The transform calculated by this function can be used to solve Poisson's equation when the solution is specified at both left and right boundaries (Swarztrauber 1977).

The function uses a variant of the fast Fourier transform (FFT) algorithm (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.

## 4. Parameters

**m**

Input: the number of sequences to be transformed, $m$.
Constraint: $\mathbf{m} \geq \mathbf{1}$.

**n**

Input: one less than the number of real values in each sequence, i.e., the number of values in each sequence is $n+1$.
Constraint: $\mathbf{n} \geq \mathbf{1}$.

**x[m∗(n+1)]**

Input: the $m$ data sequences stored in **x** consecutively. If the $n+1$ data values of the $p$th sequence to be transformed are denoted by $x_j^p$, for $j = 0, 1, \ldots, n$; $p = 1, 2, \ldots, m$, then the first $m(n+1)$ elements of the array **x** must contain the values

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

Output: the $m$ Fourier cosine transforms stored consecutively, overwriting the corresponding original sequence.

**trig[2∗n]**

Input: trigonometric coefficients as returned by a call of nag_fft_init_trig (c06gzc). nag_fft_multiple_cosine makes a simple check to ensure that **trig** has been initialised and that the initialisation is compatible with the value of **n**.

**fail**
The NAG error parameter, see the Essential Introduction to the NAG C Library.

## 5. Error Indications and Warnings

**NE_INT_ARG_LT**
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.
On entry, **n** must not be less than 1: **n** = ⟨*value*⟩.

**NE_C06_NOT_TRIG**
Value of **n** and **trig** array are incompatible or **trig** array not initialized.

**NE_ALLOC_FAIL**
Memory allocation failed.

## 6. Further Comments

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

### 6.1. 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).

### 6.2. 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 Review* **19** (3) 490–501.
Swarztrauber, P.N. (1982) Vectorizing the FFT's *Parallel Computations* G Rodrigue (ed) Academic Press pp 51–83.
Temperton C (1983) Fast Mixed-radix Real Fourier Transforms *J. Comput. Phys.* **52** 340–350.

## 7. See Also

nag_fft_init_trig (c06gzc)

## 8. Example

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

### 8.1. Program Text

```
/* nag_fft_multiple_cosine(c06hbc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagc06.h>

#define MMAX 5
#define NMAX 20
#define X(I,J) x[(I)*row_len + (J)]
```

```
main()
{
  double trig[2*NMAX], x[(NMAX+1)*MMAX];
  Integer i, j, m, n, row_len;

  Vprintf("c06hbc Example Program Results\n");
  Vscanf(" %*[^\n]");    /* Skip heading in data file */
  while (scanf("%ld %ld", &m, &n) != EOF)
    if (m <= MMAX && n <= NMAX)
      {
        row_len = n + 1;
        Vscanf(" %*[^\n]"); /* Skip text in data file */
        Vscanf(" %*[^\n]");
        for (i = 0; i < m; ++i)
          for (j = 0; j < row_len; ++j)
            Vscanf("%lf", &X(i,j));
        Vprintf("\nOriginal data values\n\n");
        for (i = 0; i < m; ++i)
          {
            for (j = 0; j < row_len; ++j)
              Vprintf(" %10.4f%s", X(i,j),
                      (j%7==6 && j!=row_len-1 ? "\n" : ""));
            Vprintf("\n");
          }

        c06gzc(n, trig, NAGERR_DEFAULT);  /* Initialise trig array */
        c06hbc(m, n, x, trig, NAGERR_DEFAULT);  /* Compute transform */
        Vprintf("\nDiscrete Fourier cosine transforms\n\n");
        for (i = 0; i < m; ++i)
          {
            for (j = 0; j < row_len; ++j)
              Vprintf(" %10.4f%s", X(i,j),
                      (j%7==6 && j!=row_len-1 ? "\n" : ""));
            Vprintf("\n");
          }
        /* Compute inverse transform */
        c06hbc(m, n, x, trig, NAGERR_DEFAULT);
        Vprintf("\nOriginal data as restored by inverse transform\n\n");
        for (i = 0; i < m; ++i)
          {
            for (j = 0; j < row_len; ++j)
              Vprintf(" %10.4f%s", X(i,j),
                      (j%7==6 && j!=row_len-1 ? "\n" : ""));
            Vprintf("\n");
          }
      }
    else Vfprintf(stderr,"\nInvalid value of m or n.\n");
  exit(EXIT_SUCCESS);
}
```

### 8.2. Program Data

```
c06hbc Example Program Data
3  6 : Number of sequences, m,   (number of values in each sequence)-1, n
Real data sequences
 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
```

### 8.3. Program Results

```
c06hbc 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
```