nag_fft_multiple_real (c06fpc)

1. Purpose

nag_fft_multiple_real (c06fpc) computes the discrete Fourier transforms of m sequences, each containing n real data values.

2. Specification

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

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

3. Description

Given m sequences of n real data values x_j^p, for $j = 0, 1, \ldots, n-1$; $p = 1, 2, \ldots, m$, this function simultaneously calculates the Fourier transforms of all the sequences defined by

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j^p \exp(-2\pi ijk/n), \qquad \text{for } k = 0, 1, \ldots, n-1; p = 1, 2, \ldots, m.$$

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

The transformed values $\hat{z}_k^p$ are complex, but for each value of p the $\hat{z}_k^p$ form a Hermitian sequence (i.e., $\hat{z}_{n-k}^p$ is the complex conjugate of $\hat{z}_k^p$), so they are completely determined by mn real numbers. The first call of nag_fft_multiple_real must be preceded by a call to nag_fft_init_trig (c06gzc) to initialise the array **trig** with trigonometric coefficients according to the value of **n**.

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j^p \exp(+2\pi ijk/n).$$

To compute this form, this function should be followed by a call to nag_multiple_conjugate_hermitian (c06gqc) to form the complex conjugates of the $\hat{z}_k^p$.

The function uses a variant of the fast Fourier transform algorithm (Brigham 1974) known as the Stockham self-sorting algorithm, which is described in Temperton (1983). 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: $\mathbf{m} \geq 1$.

n

Input: the number of real values in each sequence, n.
Constraint: $\mathbf{n} \geq 1$.

x[m∗n]

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

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

Output: the m discrete Fourier transforms in Hermitian form, stored consecutively, overwriting the corresponding original sequences. If the n components of the discrete Fourier transform $\hat{z}_k^p$ are written as $a_k^p + ib_k^p$, then for $0 \leq k \leq n/2$, a_k^p is in array element $\mathbf{x}[(p-1)*n+k]$ and for $1 \leq k \leq (n-1)/2$, b_k^p is in array element $\mathbf{x}[(p-1)*n+n-k]$.

trig[2∗n]
Input: trigonometric coefficients as returned by a call of nag_fft_init_trig (c06gzc). nag_fft_multiple_real 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 initialised.

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.
Temperton C (1983) Fast Mixed-radix Real Fourier Transforms *J. Comput. Phys.* **52** 340–350.

7. See Also

nag_multiple_conjugate_hermitian (c06gqc)
nag_fft_init_trig (c06gzc)

8. Example

This program reads in sequences of real data values and prints their discrete Fourier transforms (as computed by nag_fft_multiple_real). The Fourier transforms are expanded into full complex form using nag_multiple_hermitian_to_complex (c06gsc) and printed. Inverse transforms are then calculated by calling nag_multiple_conjugate_hermitian (c06gqc) followed by nag_fft_multiple_hermitian (c06fqc) showing that the original sequences are restored.

8.1. Program Text

```
/* nag_fft_multiple_real(c06fpc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

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

#define MMAX  5
#define NMAX 20
```

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

  Vprintf("c06fpc Example Program Results\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  while (scanf("%ld%ld", &m, &n)!=EOF)
    if (m<=MMAX && n<=NMAX)
      {
        Vprintf("\n\nm = %2ld  n = %2ld\n", m, n);
        /* Read in data and print out. */
        for (j = 0; j<m; ++j)
          for (i = 0; i<n; ++i)
            Vscanf("%lf", &x[j*n + i]);
        Vprintf("\nOriginal data values\n\n");
        for (j = 0; j<m; ++j)
          {
            Vprintf("    ");
            for (i = 0; i<n; ++i)
              Vprintf("%10.4f%s", x[j*n + i],
                      (i%6==5 && i!=n-1 ? "\n     " : ""));
            Vprintf("\n");
          }
        c06gzc(n, trig, NAGERR_DEFAULT);  /* Initialise trig array */
        /* Calculate transforms */
        c06fpc(m, n, x, trig, NAGERR_DEFAULT);
        Vprintf("\nDiscrete Fourier transforms in Hermitian format\n\n");
        for (j = 0; j<m; ++j)
          {
            Vprintf("    ");
            for (i = 0; i<n; ++i)
              Vprintf("%10.4f%s", x[j*n + i],
                      (i%6==5 && i!=n-1 ? "\n     " : ""));
            Vprintf("\n");
          }
        /* Calculate full complex form of Hermitian result */
        c06gsc(m, n, x, u, v, NAGERR_DEFAULT);
        Vprintf("\nFourier transforms in full complex form\n\n");
        for (j = 0; j<m; ++j)
          {
            Vprintf("Real");
            for (i = 0; i<n; ++i)
              Vprintf("%10.4f%s", u[j*n + i],
                      (i%6==5 && i!=n-1 ? "\n     " : ""));
            Vprintf("\nImag");
            for (i = 0; i<n; ++i)
              Vprintf("%10.4f%s", v[j*n + i],
                      (i%6==5 && i!=n-1 ? "\n     " : ""));
            Vprintf("\n\n");
          }
        /* Calculate inverse transforms */
        /* Conjugate Hermitian sequences of transforms */
        c06gqc(m, n, x, NAGERR_DEFAULT);
        /* Transform to give inverse transforms */
        c06fqc(m, n, x, trig, NAGERR_DEFAULT);
        Vprintf ("\nOriginal data as restored by inverse transform\n\n");
        for (j = 0; j<m; ++j)
          {
            Vprintf("    ");
            for (i = 0; i<n; ++i)
              Vprintf("%10.4f%s", x[j*n + i],
                      (i%6==5 && i!=n-1 ? "\n     " : ""));
            Vprintf("\n");
          }
      }
    else
```

```
      Vfprintf(stderr,"\nInvalid value of m or n.\n");
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
c06fpc Example Program Data
     3     6
     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
```

8.3. Program Results

```
c06fpc Example Program Results


m =  3  n =  6

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 Fourier transforms in Hermitian format

        1.0737   -0.1041    0.1126   -0.1467   -0.3738   -0.0044
        1.3961   -0.0365    0.0780   -0.1521   -0.0607    0.4666
        1.1237    0.0914    0.3936    0.1530    0.3458   -0.0508

Fourier transforms in full complex form

Real    1.0737   -0.1041    0.1126   -0.1467    0.1126   -0.1041
Imag    0.0000   -0.0044   -0.3738    0.0000    0.3738    0.0044

Real    1.3961   -0.0365    0.0780   -0.1521    0.0780   -0.0365
Imag    0.0000    0.4666   -0.0607    0.0000    0.0607   -0.4666

Real    1.1237    0.0914    0.3936    0.1530    0.3936    0.0914
Imag    0.0000   -0.0508    0.3458    0.0000   -0.3458    0.0508


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
```