nag_fft_complex (c06ecc)

1. Purpose

nag_fft_complex (c06ecc) calculates the discrete Fourier transform of a sequence of n complex data values.

2. Specification

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

void nag_fft_complex(Integer n, double x[], double y[], NagError *fail)
```

3. Description

Given a sequence of n complex data values z_j, for $j = 0, 1, \ldots, n-1$, this function calculates their discrete Fourier transform defined by

$$\hat{z}_k = a_k + ib_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \exp\left(-i\frac{2\pi jk}{n}\right), \qquad \text{for } k = 0, 1, \ldots, n-1.$$

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

To compute the inverse discrete Fourier transform defined by

$$\hat{w}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \exp\left(+i\frac{2\pi jk}{n}\right), \qquad \text{for } k = 0, 1, \ldots, n-1,$$

this function should be preceded and followed by calls of nag_conjugate_complex (c06gcc) to form the complex conjugates of the z_j and the $\hat{z}_k$.

The function uses the Fast Fourier Transform algorithm (Brigham 1974). There are some restrictions on the value of n (see Section 4).

4. Parameters

n

Input: the number of data values, n.
Constraint: $\mathbf{n} > 1$. The largest prime factor of **n** must not exceed 19, and the total number of prime factors of **n**, counting repetitions, must not exceed 20.

x[n]

Input: $\mathbf{x}[j]$ must contain x_j, the real part of z_j, for $j = 0, 1, \ldots, n-1$.
Output: the real parts a_k of the components of the discrete Fourier transform. a_k is contained in $\mathbf{x}[k]$, for $k = 0, 1, \ldots, n-1$.

y[n]

Input: $\mathbf{y}[j]$ must contain y_j, the imaginary part of z_j, for $j = 0, 1, \ldots, n-1$.
Output: the imaginary parts b_k of the components of the discrete Fourier transform. b_k is contained in $\mathbf{y}[k]$, for $k = 0, 1, \ldots, n-1$.

fail

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

5. Error Indications and Warnings

NE_C06_FACTOR_GT

At least one of the prime factors of **n** is greater than 19.

NE_C06_TOO_MANY_FACTORS

n has more than 20 prime factors.

NE_INT_ARG_LE

On entry, **n** must not be less than or equal to 1: **n** = ⟨*value*⟩.

6. Further Comments

The time taken by the function is approximately proportional to $n \log n$, but also depends on the factorization of n. The function is somewhat faster than average if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

On the other hand, the function is particularly slow if n has several unpaired prime factors, i.e., if the 'square-free' part of n has several 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.

7. See Also

nag_fft_hermitian (c06ebc)
nag_conjugate_complex (c06gcc)

8. Example

This program reads in a sequence of complex data values and prints their discrete Fourier transform.

It then performs an inverse transform using nag_conjugate_complex (c06gcc) and nag_fft_complex, and prints the sequence so obtained alongside the original data values.

8.1. Program Text

```
/* nag_fft_complex(c06ecc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

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

#define NMAX 20

main()
{
  Integer j, n ;
  double x[NMAX], y[NMAX], xx[NMAX], yy[NMAX];

  Vprintf("c06ecc Example Program Results\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  while (scanf("%ld", &n)!=EOF)
    if (n>1 && n<=NMAX)
      {
        for (j = 0; j<n; ++j)
          {
            Vscanf("%lf%lf", &x[j], &y[j]);
            xx[j] = x[j];
            yy[j] = y[j];
          }
        /* Compute transform */
        c06ecc(n, x, y, NAGERR_DEFAULT);
```

```
        Vprintf("\nComponents of discrete Fourier transform\n\n");
        Vprintf("         Real       Imag\n\n");
        for (j = 0; j<n; ++j)
          Vprintf("%3ld %10.5f %10.5f\n", j, x[j], y[j]);
        /* Compute inverse transform */
        /* Conjugate the transform */
        c06gcc(n, y, NAGERR_DEFAULT);
        /* Transform */
        c06ecc(n, x, y, NAGERR_DEFAULT);
        /* Conjugate to give inverse transform */
        c06gcc(n, y, NAGERR_DEFAULT);
        Vprintf("\nOriginal sequence as restored by inverse transform\n");
        Vprintf("\n             Original                 Restored\n");
        Vprintf("         Real       Imag           Real       Imag\n\n");
        for (j = 0; j<n; ++j)
          Vprintf("%3ld %10.5f %10.5f     %10.5f %10.5f\n",
                  j, xx[j], yy[j], x[j], y[j]);
      }
    else
      {
        Vfprintf(stderr,"\n Invalid value of n\n");
        exit(EXIT_FAILURE);
      }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
c06ecc Example Program Data
    7
  0.34907  -0.37168
  0.54890  -0.35669
  0.74776  -0.31175
  0.94459  -0.23702
  1.13850  -0.13274
  1.32850   0.00074
  1.51370   0.16298
```

8.3. Program Results

```
c06ecc Example Program Results

Components of discrete Fourier transform

         Real       Imag

  0    2.48361   -0.47100
  1   -0.55180    0.49684
  2   -0.36711    0.09756
  3   -0.28767   -0.05865
  4   -0.22506   -0.17477
  5   -0.14825   -0.30840
  6    0.01983   -0.56496

Original sequence as restored by inverse transform

             Original                 Restored
         Real       Imag           Real       Imag

  0    0.34907   -0.37168        0.34907   -0.37168
  1    0.54890   -0.35669        0.54890   -0.35669
  2    0.74776   -0.31175        0.74776   -0.31175
  3    0.94459   -0.23702        0.94459   -0.23702
  4    1.13850   -0.13274        1.13850   -0.13274
  5    1.32850    0.00074        1.32850    0.00074
  6    1.51370    0.16298        1.51370    0.16298
```