nag_fft_hermitian (c06ebc)

1. Purpose

nag_fft_hermitian (c06ebc) calculates the discrete Fourier transform of a Hermitian sequence of n complex data values.

2. Specification

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

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

3. Description

Given a Hermitian sequence of n complex data values z_j (i.e., a sequence such that z_0 is real and z_{n-j} is the complex conjugate of z_j, for $j = 1, 2, \ldots, n-1$) this function calculates their discrete Fourier transform defined by

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

(Note the scale factor of $1/\sqrt{n}$ in this definition.) The transformed values $\hat{x}_k$ are purely real.

To compute the inverse discrete Fourier transform defined by

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

this function should be preceded by a call of nag_conjugate_hermitian (c06gbc) to form the complex conjugates of the z_j.

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: the sequence to be transformed stored in Hermitian form. If the data values z_j are written as $x_j + iy_j$, then for $0 \le j \le n/2$, x_j is contained in $\mathbf{x}[j]$, and for $1 \le j \le (n-1)/2$, y_j is contained in $\mathbf{x}[n-j]$. It is not necessary for other elements of the sequence to be explicitly stored. (See also the Example Program.)
Output: the components of the discrete Fourier transform $\hat{x}_k$. $\hat{x}_k$ is stored in $\mathbf{x}[k]$, 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_complex (c06ecc)
nag_conjugate_hermitian (c06gbc)

8. Example

This program reads in a sequence of real data values which is assumed to be a Hermitian sequence of complex data values stored in Hermitian form. The input sequence is expanded into a full complex sequence and printed alongside the original sequence. The discrete Fourier transform (as computed by nag_fft_hermitian) is printed out.

The program then performs an inverse transform using nag_fft_real (c06eac) and nag_conjugate_hermitian (c06gbc), and prints the sequence so obtained alongside the original data values.

8.1. Program Text

```
 *
 * Mark 1, 1990.
 */

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

#define NMAX 20

main()
{
  Integer j, n, n2, nj;
  double u[NMAX], v[NMAX], x[NMAX], xx[NMAX];

  Vprintf("c06ebc 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", &x[j]);
            xx[j] = x[j];
          }
        /* Calculate full complex form of Hermitian sequence */
        u[0] = x[0];
```

```
        v[0] = 0.0;
        n2 = (n-1)/2;
        for (j = 1; j<=n2; j++)
          {
            nj = n - j;
            u[j] = x[j];
            u[nj] = x[j];
            v[j] = x[nj];
            v[nj] = -x[nj];
          }
        if (n % 2==0)
          {
            u[n2+1] = x[n2+1];
            v[n2+1] = 0.0;
          }
        Vprintf("\nOriginal and corresponding complex sequence\n");
        Vprintf("\n         Data       Real       Imag \n\n");
        for (j = 0; j<n; j++)
          Vprintf("%3ld %10.5f %10.5f %10.5f\n", j, x[j], u[j], v[j]);
        /* Calculate transform */
        c06ebc(n, x, NAGERR_DEFAULT);
        Vprintf("\nComponents of discrete Fourier transform\n\n");
        for (j = 0; j<n; j++)
          Vprintf("%3ld %10.5f\n", j, x[j]);
        /* Calculate inverse transform */
        c06eac(n, x, NAGERR_DEFAULT);
        c06gbc(n, x, NAGERR_DEFAULT);
        Vprintf("\nOriginal sequence as restored by inverse transform\n");
        Vprintf("\n       Original   Restored\n\n");
        for (j = 0; j<n; j++)
          Vprintf("%3ld %10.5f %10.5f\n", j, xx[j], x[j]);
      }
    else
      {
        Vfprintf(stderr,"Invalid value of n\n");
        exit(EXIT_FAILURE);
      }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
c06ebc Example Program Data
    7
  0.34907
  0.54890
  0.74776
  0.94459
  1.13850
  1.32850
  1.51370
```

8.3. Program Results

```
c06ebc Example Program Results

Original and corresponding complex sequence

         Data       Real       Imag

  0    0.34907    0.34907    0.00000
  1    0.54890    0.54890    1.51370
  2    0.74776    0.74776    1.32850
  3    0.94459    0.94459    1.13850
  4    1.13850    0.94459   -1.13850
  5    1.32850    0.74776   -1.32850
  6    1.51370    0.54890   -1.51370
```

```
Components of discrete Fourier transform

  0    1.82616
  1    1.86862
  2   -0.01750
  3    0.50200
  4   -0.59873
  5   -0.03144
  6   -2.62557

Original sequence as restored by inverse transform

       Original   Restored

  0    0.34907    0.34907
  1    0.54890    0.54890
  2    0.74776    0.74776
  3    0.94459    0.94459
  4    1.13850    1.13850
  5    1.32850    1.32850
  6    1.51370    1.51370
```