

nag_fft_real (c06eac)

1. Purpose

nag_fft_real (c06eac) calculates the discrete Fourier transform of a sequence of n real data values.

2. Specification

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

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

3. Description

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

$$\hat{z}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \exp\left(-i \frac{2\pi j k}{n}\right), \quad \text{for } k = 0, 1, \dots, n-1.$$

(Note the scale factor of $1/\sqrt{n}$ in this definition.) The transformed values \hat{z}_k are complex, but they form a Hermitian sequence (i.e., \hat{z}_{n-k} is the complex conjugate of \hat{z}_k), so they are completely determined by n real numbers.

The function **nag_multiple_hermitian_to_complex (c06gsc)** may be used to convert a Hermitian sequence to the corresponding complex sequence.

To compute the inverse discrete Fourier transform defined by

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

this function should be followed by a call of **nag_conjugate_hermitian (c06gbc)** to form the complex conjugates of 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 \mathbf{n} must not exceed 19, and the total number of prime factors of \mathbf{n} , counting repetitions, must not exceed 20.

x[n]

Input: $\mathbf{x}[j]$ must contain x_j , for $j = 0, 1, \dots, n-1$.

Output: the discrete Fourier transform stored in Hermitian form. If the components of the transform \hat{z}_k are written as $a_k + ib_k$, then for $0 \leq k \leq n/2$, a_k is contained in $\mathbf{x}[k]$, and for $1 \leq k \leq (n-1)/2$, b_k is contained in $\mathbf{x}[n-k]$. Elements of the sequence which are not explicitly stored are given by $a_{n-k} = a_k$, $b_{n-k} = -b_k$, $b_o = 0$ and, if n is even, $b_{n/2} = 0$. (See also the Example Program.)

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 \mathbf{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** = $\langle value \rangle$.

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_conjugate_hermitian (c06gbc)

nag_multiple_hermitian_to_complex (c06gsc)

8. Example

This program reads in a sequence of real data values, and prints their discrete Fourier transform (as computed by nag_fft_real), after expanding it from Hermitian form into a full complex sequence.

It then performs an inverse transform using nag_conjugate_hermitian (c06gbc) and nag_fft_hermitian (c06ebc), and prints the sequence so obtained alongside the original data values.

8.1. Program Text

```
/* nag_fft_real(c06eac) 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, n2, nj;
    double a[NMAX], b[NMAX], x[NMAX], xx[NMAX];

    Vprintf("c06eac 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 transform */
    c06eac(n, x, NAGERR_DEFAULT);
    /* Calculate full complex form of Hermitian result */
    a[0] = x[0];
    b[0] = 0.0;
    n2 = (n-1)/2;
    for (j = 1; j<=n2; j++)
    {
        nj = n - j;
        a[j] = x[j];
        a[nj] = x[j];
        b[j] = x[nj];
        b[nj] = -x[nj];
    }
    if (n % 2==0)
    {
        a[n2+1] = x[n2+1];
        b[n2+1] = 0.0;
    }
    Vprintf("\nComponents of discrete Fourier transform\n");
    Vprintf("\n          Real          Imag \n\n");
    for (j = 0; j<n; j++)
        Vprintf("%3ld %10.5f %10.5f\n", j, a[j], b[j]);
    /* Calculate inverse transform */
    c06gbc(n, x, NAGERR_DEFAULT);
    c06ebc(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

c06eac Example Program Data

```

7
0.34907
0.54890
0.74776
0.94459
1.13850
1.32850
1.51370

```

8.3. Program Results

c06eac Example Program Results

Components of discrete Fourier transform

	Real	Imag
0	2.48361	0.00000
1	-0.26599	0.53090
2	-0.25768	0.20298
3	-0.25636	0.05806
4	-0.25636	-0.05806
5	-0.25768	-0.20298
6	-0.26599	-0.53090

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
