

NAG Toolbox

nag_sum_fft_complex_1d_sep (c06fc)

1 Purpose

nag_sum_fft_complex_1d_sep (c06fc) calculates the discrete Fourier transform of a sequence of n complex data values (using a work array for extra speed).

2 Syntax

```
[x, y, ifail] = nag_sum_fft_complex_1d_sep(x, y, 'n', n)
```

```
[x, y, ifail] = c06fc(x, y, 'n', n)
```

3 Description

Given a sequence of n complex data values z_j , for $j = 0, 1, \dots, n-1$, nag_sum_fft_complex_1d_sep (c06fc) calculates their discrete Fourier transform defined by

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

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

To compute the inverse discrete Fourier transform defined by

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

this function should be preceded and followed by the complex conjugation of the data values and the transform (by negating the imaginary parts stored in y).

nag_sum_fft_complex_1d_sep (c06fc) uses the fast Fourier transform (FFT) algorithm (see Brigham (1974)). There are some restrictions on the value of n (see Section 5).

4 References

Brigham E O (1974) *The Fast Fourier Transform* Prentice–Hall

5 Parameters

5.1 Compulsory Input Parameters

1: **x(n)** – REAL (KIND=nag_wp) array

If **x** is declared with bounds $(0 : n - 1)$ in the function from which nag_sum_fft_complex_1d_sep (c06fc) is called, then **x(j)** must contain x_j , the real part of z_j , for $j = 0, 1, \dots, n - 1$.

2: **y(n)** – REAL (KIND=nag_wp) array

If **y** is declared with bounds $(0 : n - 1)$ in the function from which nag_sum_fft_complex_1d_sep (c06fc) is called, then **y(j)** must contain y_j , the imaginary part of z_j , for $j = 0, 1, \dots, n - 1$.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)
n, the number of data values. 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.

Constraint: **n** > 1.

5.3 Output Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array

The real parts a_k of the components of the discrete Fourier transform. If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_complex_1d_sep (c06fc) is called, then for $0 \leq k \leq n - 1$, a_k is contained in **x**(k).

2: **y**(**n**) – REAL (KIND=nag_wp) array

The imaginary parts b_k of the components of the discrete Fourier transform. If **y** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_complex_1d_sep (c06fc) is called, then for $0 \leq k \leq n - 1$, b_k is contained in **y**(k).

3: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

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

ifail = 2

n has more than 20 prime factors.

ifail = 3

On entry, **n** ≤ 1.

ifail = 4

An unexpected error has occurred in an internal call. Check all function calls and array dimensions. Seek expert help.

ifail = -99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = -399

Your licence key may have expired or may not have been installed correctly.

ifail = -999

Dynamic memory allocation failed.

7 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).

8 Further Comments

The time taken is approximately proportional to $n \times \log(n)$, but also depends on the factorization of n . `nag_sum_fft_complex_1d_sep` (c06fc) is faster if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

9 Example

This example reads in a sequence of complex data values and prints their discrete Fourier transform (as computed by `nag_sum_fft_complex_1d_sep` (c06fc)). It then performs an inverse transform using `nag_sum_fft_complex_1d_sep` (c06fc), and prints the sequence so obtained alongside the original data values.

9.1 Program Text

```
function c06fc_example

fprintf('c06fc example results\n\n');

z = [0.34907 - 0.37168*i;
     0.54890 - 0.35669*i;
     0.74776 - 0.31175*i;
     0.94459 - 0.23702*i;
     1.13850 - 0.13274*i;
     1.32850 + 0.00074*i;
     1.51370 + 0.16298*i];
x = real(z);
y = imag(z);

[ztr, zti, ifail] = c06fc(x, y);
ztrans = ztr + i*zti;
disp('Components of discrete Fourier transform')
disp(ztrans);

[xres, yres, ifail] = c06fc(ztr, -zti);
zres = xres-i*yres;
zout = [z zres];

fprintf('Original sequence as restored by inverse transform\n');
fprintf('      Original          Restored\n');
disp(zout);
```

9.2 Program Results

```
c06fc example results

Components of discrete Fourier transform
 2.4836 - 0.4710i
-0.5518 + 0.4968i
-0.3671 + 0.0976i
-0.2877 - 0.0586i
-0.2251 - 0.1748i
-0.1483 - 0.3084i
 0.0198 - 0.5650i

Original sequence as restored by inverse transform
      Original          Restored
 0.3491 - 0.3717i    0.3491 - 0.3717i
 0.5489 - 0.3567i    0.5489 - 0.3567i
```

0.7478 - 0.3118i	0.7478 - 0.3117i
0.9446 - 0.2370i	0.9446 - 0.2370i
1.1385 - 0.1327i	1.1385 - 0.1327i
1.3285 + 0.0007i	1.3285 + 0.0007i
1.5137 + 0.1630i	1.5137 + 0.1630i
