NAG Toolbox

nag_sum_fft_complex_1d (c06pc)

1 Purpose

nag_sum_fft_complex_1d (c06pc) calculates the discrete Fourier transform of a sequence of n complex data values (using complex data type).

2 Syntax

```
[x, ifail] = nag_sum_fft_complex_1d(direct, x, 'n', n)
[x, ifail] = c06pc(direct, x, 'n', n)
```

3 Description

Given a sequence of n complex data values z_j, for $j = 0, 1, \ldots, n-1$, nag_sum_fft_complex_1d (c06pc) calculates their (**forward** or **backward**) discrete Fourier transform (DFT) defined by

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

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The minus sign is taken in the argument of the exponential within the summation when the forward transform is required, and the plus sign is taken when the backward transform is required.

A call of nag_sum_fft_complex_1d (c06pc) with **direct** = 'F' followed by a call with **direct** = 'B' will restore the original data.

nag_sum_fft_complex_1d (c06pc) uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, which is described in Temperton (1983). If n is a large prime number or if n contains large prime factors, then the Fourier transform is performed using Bluestein's algorithm (see Bluestein (1968)), which expresses the DFT as a convolution that in turn can be efficiently computed using FFTs of highly composite sizes.

4 References

Bluestein L I (1968) A linear filtering approach to the computation of the discrete Fourier transform *Northeast Electronics Research and Engineering Meeting Record 10* 218–219

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

Temperton C (1983) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

5 Parameters

5.1 Compulsory Input Parameters

1: **direct** – CHARACTER(1)

If the forward transform as defined in Section 3 is to be computed, then **direct** must be set equal to `F'.

If the backward transform is to be computed then **direct** must be set equal to `B'.

Constraint: **direct** = 'F' or 'B'.

2: **x**(**n**) – COMPLEX (KIND=nag_wp) array

If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_complex_1d (c06pc) is called, then $\mathbf{x}(j)$ must contain z_j, for $j = 0, 1, \ldots, n - 1$.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **x**.

n, the number of data values.

Constraint: $\mathbf{n} \geq 1$.

5.3 Output Parameters

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

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 (c06pc) is called, then $\hat{z}_k$ is contained in $\mathbf{x}(k)$, for $0 \leq k \leq n - 1$.

2: **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 $= 1$

On entry, $\mathbf{n} < 1$.

ifail $= 2$

On entry, **direct** $\neq$ 'F' or 'B'.

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 (c06pc) 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. When the Bluestein's FFT algorithm is in use, an additional complex workspace of size approximately $8n$ is allocated.

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 (c06pc) with **direct** = 'F'). It then performs an inverse transform using nag_sum_fft_complex_1d (c06pc) with **direct** = 'B', and prints the sequence so obtained alongside the original data values.

9.1 Program Text

```
     function c06pc_example

fprintf('c06pc example results\n\n');

direct = 'F';
x = [ 0.34907 - 0.37168i;
      0.54890 - 0.35669i;
      0.74776 - 0.31175i;
      0.94459 - 0.23702i;
      1.13850 - 0.13274i;
      1.32850 + 0.00074i;
      1.51370 + 0.16298i];

[xt, ifail] = c06pc(direct, x);

disp('Discrete Fourier Transform of x:');
disp(xt);

% Restore x by inverse transform
direct = 'B';
[xr, ifail] = c06pc(direct, xt);

fprintf('Original sequence as restored by inverse transform\n\n');
fprintf('       Original          Restored\n');
z = [x xr];
disp(z);
```

9.2 Program Results

```
     c06pc example results

Discrete Fourier Transform of x:
   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 - 0.3118i
   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
```
