NAG Toolbox

nag_sum_fft_complex_multid_1d_sep (c06ff)

1 Purpose

nag_sum_fft_complex_multid_1d_sep (c06ff) computes the discrete Fourier transform of one variable in a multivariate sequence of complex data values.

2 Syntax

```
[x, y, ifail] = nag_sum_fft_complex_multid_1d_sep(l, nd, x, y, 'ndim', ndim, 'n', n)
[x, y, ifail] = c06ff(l, nd, x, y, 'ndim', ndim, 'n', n)
```

3 Description

nag_sum_fft_complex_multid_1d_sep (c06ff) computes the discrete Fourier transform of one variable (the lth say) in a multivariate sequence of complex data values $z_{j_1 j_2 \ldots j_m}$, for $j1 = 0, 1, \ldots, n_1 - 1$ and $j2 = 0, 1, \ldots, n_2 - 1$, and so on. Thus the individual dimensions are $n_1, n_2, \ldots, n_m$, and the total number of data values is $n = n_1 \times n_2 \times \cdots \times n_m$.

The function computes n/n_l one-dimensional transforms defined by

$$\hat{z}_{j_1 \ldots k_l \ldots j_m} = \frac{1}{\sqrt{n_l}} \sum_{j_l=0}^{n_l-1} z_{j_1 \ldots j_l \ldots j_m} \times \exp\left(-\frac{2\pi i j_l k_l}{n_l}\right),$$

where $k_l = 0, 1, \ldots, n_l - 1$.

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

To compute the inverse discrete Fourier transforms, defined with $\exp\left(+\frac{2\pi i j_l k_l}{n_l}\right)$ in the above formula instead of $\exp\left(-\frac{2\pi i j_l k_l}{n_l}\right)$, 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).

The data values must be supplied in a pair of one-dimensional arrays (real and imaginary parts separately), in accordance with the Fortran convention for storing multidimensional data (i.e., with the first subscript j_1 varying most rapidly).

This function calls nag_sum_fft_complex_1d_sep (c06fc) to perform one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm in Brigham (1974), and hence there are some restrictions on the values of n_l (see Section 5).

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **l** – INTEGER

l, the index of the variable (or dimension) on which the discrete Fourier transform is to be performed.

Constraint: $1 \leq \mathbf{l} \leq \mathbf{ndim}$.

2: **nd**(**ndim**) – INTEGER array

nd(i) must contain n_i (the dimension of the ith variable) , for $i = 1, 2, \ldots, m$. The largest prime factor of **nd**(l) must not exceed 19, and the total number of prime factors of **nd**(l), counting repetitions, must not exceed 20.

Constraint: **nd**(i) ≥ 1, for $i = 1, 2, \ldots,$ **ndim**.

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

x($1 + j_1 + n_1 j_2 + n_1 n_2 j_3 + \ldots$) must contain the real part of the complex data value $z_{j_1 j_2 \ldots j_m}$, for $0 \leq j_1 \leq n_1 - 1, 0 \leq j_2 \leq n_2 - 1, \ldots$; i.e., the values are stored in consecutive elements of the array according to the Fortran convention for storing multidimensional arrays.

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

The imaginary parts of the complex data values, stored in the same way as the real parts in the array **x**.

5.2 Optional Input Parameters

1: **ndim** – INTEGER

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

m, the number of dimensions (or variables) in the multivariate data.

Constraint: **ndim** ≥ 1.

2: **n** – INTEGER

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

n, the total number of data values.

Constraint: **n** $=$ **nd**(1) $\times$ **nd**(2) $\times \cdots \times$ **nd**(**ndim**).

5.3 Output Parameters

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

The real parts of the corresponding elements of the computed transform.

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

The imaginary parts of the corresponding elements of the computed transform.

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 $= 1$

On entry, **ndim** < 1.

ifail $= 2$

On entry, **n** $\neq$ **nd**(1) $\times$ **nd**(2) $\times \cdots \times$ **nd**(**ndim**).

ifail $= 3$

On entry, **l** < 1 or **l** $>$ **ndim**.

ifail $= 10 \times l + 1$

At least one of the prime factors of **nd**(l) is greater than 19.

ifail $= 10 \times l + 2$

nd(l) has more than 20 prime factors.

ifail $= 10 \times l + 3$

On entry, **nd**$(l) < 1$.

ifail $= 10 \times l + 4$

On entry, $lwork < 3 \times$ **nd**(l).

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 $n \times \log n_l$, but also depends on the factorization of n_l. nag_sum_fft_complex_multid_1d_sep (c06ff) is faster if the only prime factors of n_l are 2, 3 or 5; and fastest of all if n_l is a power of 2.

9 Example

This example reads in a bivariate sequence of complex data values and prints the discrete Fourier transform of the second variable. It then performs an inverse transform and prints the sequence so obtained, which may be compared with the original data values.

9.1 Program Text

```
function c06ff_example

fprintf('c06ff example results\n\n');

x = [ 1.000     0.999     0.987     0.936     0.802;
      0.994     0.989     0.963     0.891     0.731;
      0.903     0.885     0.823     0.694     0.467];
y = [ 0.000    -0.040    -0.159    -0.352    -0.597;
     -0.111    -0.151    -0.268    -0.454    -0.682
     -0.430    -0.466    -0.568    -0.720    -0.884];
nd = nag_int(size(x));
l  = nag_int(2);

% transform, then inverse transform to restore data
[xt, yt, ifail] = c06ff(l, nd, x, y);
[xr, yr, ifail] = c06ff(l, nd, xt, -yt);

% Display as complex matrices
```

```
z = x + i*y;
zt = reshape(xt+i*yt,nd);
zr = reshape(xr-i*yr,nd);

matrix = 'general';
diag = ' ';
usefrm = 'Above';
format = 'F9.3';
labrow = 'None';
labcol = 'None';
ncols  = nag_int(80);
indent = nag_int(0);

title  = 'Original data:';
[ifail] = x04db(...
    matrix, diag, z, usefrm, format, title, labrow, labcol, ncols, indent);
disp(' ');
title = 'Discrete Fourier transform of variable 2:';
[ifail] = x04db(...
    matrix, diag, zt, usefrm, format, title, labrow, labcol, ncols, indent);
disp(' ');
title = 'Original sequence as restored by inverse transform:';
[ifail] = x04db(...
    matrix, diag, zr, usefrm, format, title, labrow, labcol, ncols, indent);
```

9.2 Program Results

```
     c06ff example results

 Original data:
      1.000    0.999    0.987    0.936    0.802
      0.000   -0.040   -0.159   -0.352   -0.597

      0.994    0.989    0.963    0.891    0.731
     -0.111   -0.151   -0.268   -0.454   -0.682

      0.903    0.885    0.823    0.694    0.467
     -0.430   -0.466   -0.568   -0.720   -0.884

 Discrete Fourier transform of variable 2:
      2.113    0.288    0.126   -0.003   -0.287
     -0.513   -0.000    0.130    0.190    0.194

      2.043    0.286    0.139    0.018   -0.263
     -0.745   -0.032    0.115    0.189    0.225

      1.687    0.260    0.170    0.079   -0.176
     -1.372   -0.125    0.063    0.173    0.299

 Original sequence as restored by inverse transform:
      1.000    0.999    0.987    0.936    0.802
      0.000   -0.040   -0.159   -0.352   -0.597

      0.994    0.989    0.963    0.891    0.731
     -0.111   -0.151   -0.268   -0.454   -0.682

      0.903    0.885    0.823    0.694    0.467
     -0.430   -0.466   -0.568   -0.720   -0.884
```