

NAG Toolbox

nag_sum_fft_complex_1d_multi_rfmt (c06fr)

1 Purpose

nag_sum_fft_complex_1d_multi_rfmt (c06fr) computes the discrete Fourier transforms of m sequences, each containing n complex data values. This function is designed to be particularly efficient on vector processors.

Note: This function is scheduled to be withdrawn, please see c06fr in Advice on Replacement Calls for Withdrawn/Superseded Routines..

2 Syntax

```
[x, y, trig, ifail] = nag_sum_fft_complex_1d_multi_rfmt(m, n, x, y, init, trig)
[x, y, trig, ifail] = c06fr(m, n, x, y, init, trig)
```

3 Description

Given m sequences of n complex data values z_j^p , for $j = 0, 1, \dots, n - 1$ and $p = 1, 2, \dots, m$, nag_sum_fft_complex_1d_multi_rfmt (c06fr) simultaneously calculates the Fourier transforms of all the sequences defined by

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

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

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

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

To compute this form, this function should be preceded and followed by a call of nag_sum_conjugate_complex_sep (c06gc) to form the complex conjugates of the z_j^p and the \hat{z}_k^p .

The function 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). Special code is provided for the factors 2, 3, 4, 5 and 6. This function is designed to be particularly efficient on vector processors, and it becomes especially fast as m , the number of transforms to be computed in parallel, increases.

4 References

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: **m** – INTEGER

m , the number of sequences to be transformed.

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

2: **n** – INTEGER

n , the number of complex values in each sequence.

Constraint: $n \geq 1$.

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

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

The real and imaginary parts of the complex data must be stored in **x** and **y** respectively as if in a two-dimensional array of dimension $(1 : m, 0 : n - 1)$; each of the m sequences is stored in a **row** of each array. In other words, if the real parts of the p th sequence to be transformed are denoted by x_j^p , for $j = 0, 1, \dots, n - 1$, then the mn elements of the array **x** must contain the values

$$x_0^1, x_0^2, \dots, x_0^m, x_1^1, x_1^2, \dots, x_1^m, \dots, x_{n-1}^1, x_{n-1}^2, \dots, x_{n-1}^m.$$

5: **init** – CHARACTER(1)

Indicates whether trigonometric coefficients are to be calculated.

init = 'I'

Calculate the required trigonometric coefficients for the given value of n , and store in the array **trig**.

init = 'S' or 'R'

The required trigonometric coefficients are assumed to have been calculated and stored in the array **trig** in a prior call to one of `nag_sum_fft_real_1d_multi_rfmt` (c06fp), `nag_sum_fft_hermitian_1d_multi_rfmt` (c06fq) or `nag_sum_fft_complex_1d_multi_rfmt` (c06fr). The function performs a simple check that the current value of n is consistent with the values stored in **trig**.

Constraint: **init** = 'I', 'S' or 'R'.

6: **trig(2 × n)** – REAL (KIND=nag_wp) array

If **init** = 'S' or 'R', **trig** must contain the required trigonometric coefficients that have been previously calculated. Otherwise **trig** need not be set.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

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

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

x and **y** store the real and imaginary parts of the complex transforms.

3: **trig(2 × n)** – REAL (KIND=nag_wp) array

Contains the required coefficients (computed by the function if **init** = 'I').

4: **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, **m** < 1.

ifail = 2

On entry, **n** < 1.

ifail = 3

On entry, **init** ≠ 'T', 'S' or 'R'.

ifail = 4

Not used at this Mark.

ifail = 5

On entry, **init** = 'S' or 'R', but the array **trig** and the current value of **n** are inconsistent.

ifail = 6

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 by `nag_sum_fft_complex_1d_multi_rfmt` (c06fr) is approximately proportional to $n \log(n)$, but also depends on the factors of n . `nag_sum_fft_complex_1d_multi_rfmt` (c06fr) is fastest if the only prime factors of n are 2, 3 and 5, and is particularly slow if n is a large prime, or has large prime factors.

9 Example

This example reads in sequences of complex data values and prints their discrete Fourier transforms (as computed by `nag_sum_fft_complex_1d_multi_rfmt` (c06fr)). Inverse transforms are then calculated using `nag_sum_fft_complex_1d_multi_rfmt` (c06fr) and `nag_sum_conjugate_complex_sep` (c06gc) and printed out, showing that the original sequences are restored.

9.1 Program Text

```

function c06fr_example

fprintf('c06fr example results\n\n');

% 3 complex sequences, real and imaginary parts stored as rows
% in separate arrays.
m = nag_int(3);
n = nag_int(6);
zr = [0.3854 0.6772 0.1138 0.6751 0.6362 0.1424;
       0.9172 0.0644 0.6037 0.6430 0.0428 0.4815;
       0.1156 0.0685 0.2060 0.8630 0.6967 0.2792];
zi = [0.5417 0.2983 0.1181 0.7255 0.8638 0.8723;
       0.9089 0.3118 0.3465 0.6198 0.2668 0.1614;
       0.6214 0.8681 0.7060 0.8652 0.9190 0.3355];

z = zr + i*zi;
title = 'Original sequences:';
[ifail] = x04da('General','Non-unit', z, title);

% Transform
init = 'Initial';
trig = zeros(2*n,1);
[ztr, zti, trig, ifail] = c06fr(m, n, zr, zi, init, trig);

zt = ztr + i*zti;
disp(' ');
title = 'Discrete Fourier Transforms:';
[ifail] = x04da('General','Non-unit', zt, title);

% Restore by transform with pre- and post-conjugation
zti = -zti;
init = 'Subsequent';
[zrr, zri, trig, ifail] = c06fr(m, n, ztr, zti, init, trig);
zr = zrr - i*zri;

disp(' ');
title = 'Original data as restored by inverse transform';
[ifail] = x04da('General','Non-unit', zr, title);

```

9.2 Program Results

c06fr example results

Original sequences:

	1	2	3	4	5	6
1	0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
	0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
2	0.9172	0.0644	0.6037	0.6430	0.0428	0.4815
	0.9089	0.3118	0.3465	0.6198	0.2668	0.1614
3	0.1156	0.0685	0.2060	0.8630	0.6967	0.2792
	0.6214	0.8681	0.7060	0.8652	0.9190	0.3355

Discrete Fourier Transforms:

	1	2	3	4	5	6
1	1.0737	-0.5706	0.1733	-0.1467	0.0518	0.3625
	1.3961	-0.0409	-0.2958	-0.1521	0.4517	-0.0321
2	1.1237	0.1728	0.4185	0.1530	0.3686	0.0101
	1.0677	0.0386	0.7481	0.1752	0.0565	0.1403
3	0.9100	-0.3054	0.4079	-0.0785	-0.1193	-0.5314
	1.7617	0.0624	-0.0695	0.0725	0.1285	-0.4335

Original data as restored by inverse transform

	1	2	3	4	5	6
1	0.3854	0.6772	0.1138	0.6751	0.6362	0.1424

	0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
2	0.9172	0.0644	0.6037	0.6430	0.0428	0.4815
	0.9089	0.3118	0.3465	0.6198	0.2668	0.1614
3	0.1156	0.0685	0.2060	0.8630	0.6967	0.2792
	0.6214	0.8681	0.7060	0.8652	0.9190	0.3355
