

## NAG Toolbox

### nag\_sum\_fft\_realherm\_1d\_multi\_row (c06pp)

#### 1 Purpose

nag\_sum\_fft\_realherm\_1d\_multi\_row (c06pp) computes the discrete Fourier transforms of  $m$  sequences, each containing  $n$  real data values or a Hermitian complex sequence stored in a complex storage format.

#### 2 Syntax

```
[x, ifail] = nag_sum_fft_realherm_1d_multi_row(direct, m, n, x)
[x, ifail] = c06pp(direct, m, n, x)
```

#### 3 Description

Given  $m$  sequences of  $n$  real data values  $x_j^p$ , for  $j = 0, 1, \dots, n-1$  and  $p = 1, 2, \dots, m$ , nag\_sum\_fft\_realherm\_1d\_multi\_row (c06pp) simultaneously calculates the Fourier transforms of all the sequences defined by

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

The transformed values  $\hat{z}_k^p$  are complex, but for each value of  $p$  the  $\hat{z}_k^p$  form a Hermitian sequence (i.e.,  $\hat{z}_{n-k}^p$  is the complex conjugate of  $\hat{z}_k^p$ ), so they are completely determined by  $mn$  real numbers (since  $\hat{z}_0^p$  is real, as is  $\hat{z}_{n/2}^p$  for  $n$  even).

Alternatively, given  $m$  Hermitian sequences of  $n$  complex data values  $z_j^p$ , this function simultaneously calculates their inverse (**backward**) discrete Fourier transforms defined by

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

The transformed values  $\hat{x}_k^p$  are real.

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

A call of nag\_sum\_fft\_realherm\_1d\_multi\_row (c06pp) with **direct** = 'F' followed by a call with **direct** = 'B' will restore the original data.

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 coding is provided for the factors 2, 3, 4 and 5.

#### 4 References

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

Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

## 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: **m** – INTEGER

$m$ , the number of sequences to be transformed.

*Constraint:*  $m \geq 1$ .

- 3: **n** – INTEGER

$n$ , the number of real or complex values in each sequence.

*Constraint:*  $n \geq 1$ .

- 4: **x**( $m \times (n + 2)$ ) – REAL (KIND=nag\_wp) array

The data must be stored in **x** 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 the array. In other words, if the data values of the  $p$ th sequence to be transformed are denoted by  $x_j^p$ , for  $j = 0, 1, \dots, n - 1$ , then:

if **direct** = 'F', **x**( $j \times m + p$ ) must contain  $x_j^p$ , for  $j = 0, 1, \dots, n - 1$  and  $p = 1, 2, \dots, m$ ;

if **direct** = 'B', **x**( $2 \times k \times m + p$ ) and **x**( $(2 \times k + 1) \times m + p$ ) must contain the real and imaginary parts respectively of  $\hat{z}_k^p$ , for  $k = 0, 1, \dots, n/2$  and  $p = 1, 2, \dots, m$ . (Note that for the sequence  $\hat{z}_k^p$  to be Hermitian, the imaginary part of  $\hat{z}_0^p$ , and of  $\hat{z}_{n/2}^p$  for  $n$  even, must be zero.)

### 5.2 Optional Input Parameters

None.

### 5.3 Output Parameters

- 1: **x**( $m \times (n + 2)$ ) – REAL (KIND=nag\_wp) array

if **direct** = 'F' and **x** is declared with bounds  $(1 : m, 0 : n + 1)$  then **x**( $p, 2 \times k$ ) and **x**( $p, 2 \times k + 1$ ) will contain the real and imaginary parts respectively of  $\hat{z}_k^p$ , for  $k = 0, 1, \dots, n/2$  and  $p = 1, 2, \dots, m$ ;

if **direct** = 'B' and **x** is declared with bounds  $(1 : m, 0 : n + 1)$  then **x**( $p, j$ ) will contain  $x_j^p$ , for  $j = 0, 1, \dots, n - 1$  and  $p = 1, 2, \dots, m$ .

- 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,  $m < 1$ .

**ifail** = 2

On entry, **n** < 1.

**ifail** = 3

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 by `nag_sum_fft_realherm_1d_multi_row` (c06pp) is approximately proportional to  $nm \log(n)$ , but also depends on the factors of  $n$ . `nag_sum_fft_realherm_1d_multi_row` (c06pp) 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 real data values and prints their discrete Fourier transforms (as computed by `nag_sum_fft_realherm_1d_multi_row` (c06pp) with **direct** = 'F'), after expanding them from complex Hermitian form into a full complex sequences. Inverse transforms are then calculated by calling `nag_sum_fft_realherm_1d_multi_row` (c06pp) with **direct** = 'B' showing that the original sequences are restored.

### 9.1 Program Text

```
function c06pp_example

fprintf('c06pp example results\n\n');

% 3 real sequences stored as rows
m = nag_int(3);
n = nag_int(6);
x = [0.3854    0.6772    0.1138    0.6751    0.6362    0.1424    0    0;
     0.5417    0.2983    0.1181    0.7255    0.8638    0.8723    0    0;
     0.9172    0.0644    0.6037    0.6430    0.0428    0.4815    0    0];
disp('Original data values:');
disp(x(:,1:n));

% Transform to get Hermitian sequences
direct = 'F';
[xt, ifail] = c06pp(direct, m, n, x);
zt = xt(:,1:2:n+1) + i*xt(:,2:2:n+2);
```

```

title = 'Discrete Fourier transforms in complex Hermitian format:';
[ifail] = x04da('General','Non-unit', zt, title);

for j = 1:m
    zt(j,1:n) = nag_herm2complex(n,xt(j,:));
end
title = 'Discrete Fourier transforms in full complex format:';
disp(' ');
[ifail] = x04da('General','Non-unit', zt, title);

% Restore data by back transform
direct = 'B';
[xr, ifail] = c06pp(direct, m, n, xt);
disp(' ');
disp('Original data as restored by inverse transform:');
disp(xr(:,1:n));

function [z] = nag_herm2complex(n,x);
    z(1) = complex(x(1));
    for j = 2:floor(double(n)/2) + 1
        z(j) = x(2*j-1) + i*x(2*j);
        z(n-j+2) = x(2*j-1) - i*x(2*j);
    end

```

## 9.2 Program Results

c06pp example results

Original data values:

0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815

Discrete Fourier transforms in complex Hermitian format:

	1	2	3	4
1	1.0737 0.0000	-0.1041 -0.0044	0.1126 -0.3738	-0.1467 0.0000
2	1.3961 0.0000	-0.0365 0.4666	0.0780 -0.0607	-0.1521 0.0000
3	1.1237 0.0000	0.0914 -0.0508	0.3936 0.3458	0.1530 0.0000

Discrete Fourier transforms in full complex format:

	1	2	3	4	5	6
1	1.0737 0.0000	-0.1041 -0.0044	0.1126 -0.3738	-0.1467 0.0000	0.1126 0.3738	-0.1041 0.0044
2	1.3961 0.0000	-0.0365 0.4666	0.0780 -0.0607	-0.1521 0.0000	0.0780 0.0607	-0.0365 -0.4666
3	1.1237 0.0000	0.0914 -0.0508	0.3936 0.3458	0.1530 0.0000	0.3936 -0.3458	0.0914 0.0508

Original data as restored by inverse transform:

0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815

---