# NAG Toolbox

# nag_sum_fft_complex_1d_multi_row (c06pr)

## 1 Purpose

nag_sum_fft_complex_1d_multi_row (c06pr) computes the discrete Fourier transforms of $m$ sequences, each containing $n$ complex data values.

## 2 Syntax

```
[x, ifail] = nag_sum_fft_complex_1d_multi_row(direct, m, n, x)
[x, ifail] = c06pr(direct, m, n, x)
```

## 3 Description

Given $m$ sequences of $n$ complex data values $z_j^p$, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, nag_sum_fft_complex_1d_multi_row (c06pr) simultaneously calculates the (**forward** or **backward**) discrete Fourier transforms of all the sequences defined by

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

(Note the scale factor $\frac{1}{\sqrt{n}}$ 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_multi_row (c06pr) 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 code 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) 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: **m** – INTEGER

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

*Constraint*: **m** $\geq 1$.

3: **n** – INTEGER

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

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

4: **x**($\mathbf{m} \times \mathbf{n}$) – COMPLEX (KIND=nag_wp) array

The complex data must be stored in **x** as if in a two-dimensional array of dimension $(1 : \mathbf{m}, 0 : \mathbf{n} - 1)$; each of the $m$ sequences is stored in a **row** of each array. In other words, if the elements of the $p$th sequence to be transformed are denoted by $z_j^p$, for $j = 0, 1, \ldots, n - 1$, then $\mathbf{x}(j \times \mathbf{m} + p)$ must contain $z_j^p$.

### 5.2 Optional Input Parameters

None.

### 5.3 Output Parameters

1: **x**($\mathbf{m} \times \mathbf{n}$) – COMPLEX (KIND=nag_wp) array

Stores the complex transforms.

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, $\mathbf{m} < 1$.

**ifail** $= 2$

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

**ifail** $= 3$

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

**ifail** $= 5$

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_row (c06pr) is approximately proportional to $nm\log(n)$, but also depends on the factors of $n$. nag_sum_fft_complex_1d_multi_row (c06pr) 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_row (c06pr) with **direct** = 'F'). Inverse transforms are then calculated using nag_sum_fft_complex_1d_multi_row (c06pr) with **direct** = 'B' and printed out, showing that the original sequences are restored.

## 9.1 Program Text

```
function c06pr_example

fprintf('c06pr example results\n\n');

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
direct = 'F';
[zt, ifail] = c06pr(direct, m, n, z);
disp(' ');
title = 'Discrete Fourier Transforms:';
[ifail] = x04da('General','Non-unit', zt, title);

% Restore by back-transform
direct = 'B';
[zr, ifail] = c06pr(direct, m, n, zt);
disp(' ');
title = 'Original data as restored by inverse transform';
[ifail] = x04da('General','Non-unit', zr, title);
```

## 9.2 Program Results

```
c06pr 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
```