

## NAG Toolbox

### **nag\_sum\_fft\_complex\_3d (c06px)**

## 1 Purpose

`nag_sum_fft_complex_3d (c06px)` computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values (using complex data type).

## 2 Syntax

```
[x, ifail] = nag_sum_fft_complex_3d(direct, n1, n2, n3, x)
[x, ifail] = c06px(direct, n1, n2, n3, x)
```

## 3 Description

`nag_sum_fft_complex_3d (c06px)` computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values  $z_{j_1 j_2 j_3}$ , for  $j_1 = 0, 1, \dots, n_1 - 1$ ,  $j_2 = 0, 1, \dots, n_2 - 1$  and  $j_3 = 0, 1, \dots, n_3 - 1$ .

The discrete Fourier transform is here defined by

$$\hat{z}_{k_1 k_2 k_3} = \frac{1}{\sqrt{n_1 n_2 n_3}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} z_{j_1 j_2 j_3} \times \exp\left(\pm 2\pi i \left(\frac{j_1 k_1}{n_1} + \frac{j_2 k_2}{n_2} + \frac{j_3 k_3}{n_3}\right)\right),$$

where  $k_1 = 0, 1, \dots, n_1 - 1$ ,  $k_2 = 0, 1, \dots, n_2 - 1$  and  $k_3 = 0, 1, \dots, n_3 - 1$ .

(Note the scale factor of  $\frac{1}{\sqrt{n_1 n_2 n_3}}$  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_3d (c06px)` with `direct = 'F'` followed by a call with `direct = 'B'` will restore the original data.

This function performs multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm (see Brigham (1974)).

## 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: **n1** – INTEGER  
 $n_1$ , the first dimension of the transform.  
*Constraint:*  $\mathbf{n1} \geq 1$ .
- 3: **n2** – INTEGER  
 $n_2$ , the second dimension of the transform.  
*Constraint:*  $\mathbf{n2} \geq 1$ .
- 4: **n3** – INTEGER  
 $n_3$ , the third dimension of the transform.  
*Constraint:*  $\mathbf{n3} \geq 1$ .
- 5: **x(n1 × n2 × n3)** – COMPLEX (KIND=nag\_wp) array

The complex data values. Data values are stored in **x** using column-major ordering for storing multidimensional arrays; that is,  $z_{j_1 j_2 j_3}$  is stored in **x**(1 +  $j_1 + n_1 j_2 + n_1 n_2 j_3$ ).

## 5.2 Optional Input Parameters

None.

## 5.3 Output Parameters

- 1: **x(n1 × n2 × n3)** – COMPLEX (KIND=nag\_wp) array  
The corresponding elements of the computed transform.
- 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,  $\mathbf{n1} < 1$ .

**ifail** = 2  
On entry,  $\mathbf{n2} < 1$ .

**ifail** = 3  
On entry,  $\mathbf{n3} < 1$ .

**ifail** = 4  
On entry, **direct** ≠ 'F' or 'B'.

**ifail** = 8  
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  $n_1 n_2 n_3 \times \log(n_1 n_2 n_3)$ , but also depends on the factorization of the individual dimensions  $n_1$ ,  $n_2$  and  $n_3$ . nag\_sum\_fft\_complex\_3d (c06px) is faster if the only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2.

## 9 Example

This example reads in a trivariate sequence of complex data values and prints the three-dimensional Fourier transform. It then performs an inverse transform and prints the sequence so obtained, which may be compared to the original data values.

### 9.1 Program Text

```
function c06px_example

fprintf('c06px example results\n\n');

n1 = nag_int(2);
n2 = nag_int(3);
n3 = nag_int(4);
x = cat(4,[1           0.994-0.111i  0.903-0.430i;
            0.500+0.500i  0.494+0.111i  0.403+0.430i], ...
[0.999-0.040i  0.989-0.151i  0.885-0.466i;
            0.499+0.040i  0.489+0.151i  0.385+0.466i], ...
[0.987-0.159i  0.963-0.268i  0.823-0.568i;
            0.487+0.159i  0.463+0.268i  0.323+0.568i], ...
[0.936-0.352i  0.891-0.454i  0.694-0.720i;
            0.436+0.352i  0.391+0.454i  0.194+0.720i]);

direct = 'F';
[xt, ifail] = c06px(direct, n1, n2, n3, x);
direct = 'B';
[xr, ifail] = c06px(direct, n1, n2, n3, xt);

disp('Original data:');
disp(x);
disp('');
disp('Components of discrete Fourier transform:');
disp(xt);
disp('');
disp('Original sequence as restored by inverse transform:');
disp(xr);
```

### 9.2 Program Results

c06px example results

Original data:

(:,:,1,1) =

```

1.0000 + 0.0000i  0.9940 - 0.1110i  0.9030 - 0.4300i
0.5000 + 0.5000i  0.4940 + 0.1110i  0.4030 + 0.4300i

(:,:,1,2) =
0.9990 - 0.0400i  0.9890 - 0.1510i  0.8850 - 0.4660i
0.4990 + 0.0400i  0.4890 + 0.1510i  0.3850 + 0.4660i

(:,:,1,3) =
0.9870 - 0.1590i  0.9630 - 0.2680i  0.8230 - 0.5680i
0.4870 + 0.1590i  0.4630 + 0.2680i  0.3230 + 0.5680i

(:,:,1,4) =
0.9360 - 0.3520i  0.8910 - 0.4540i  0.6940 - 0.7200i
0.4360 + 0.3520i  0.3910 + 0.4540i  0.1940 + 0.7200i

```

Components of discrete Fourier transform:

```

(:,:,1,1) =
3.2921 + 0.1021i  0.1433 - 0.0860i  0.1433 + 0.2902i
1.2247 - 1.6203i  0.4243 + 0.3197i  -0.4243 + 0.3197i

(:,:,1,2) =
0.0506 - 0.0416i  0.0155 + 0.1527i  -0.0502 + 0.1180i
0.3548 + 0.0833i  0.0204 - 0.1147i  0.0070 - 0.0800i

(:,:,1,3) =
0.1127 + 0.1021i  -0.0245 + 0.1268i  -0.0245 + 0.0773i
0.0000 + 0.1621i  0.0134 - 0.0914i  -0.0134 - 0.0914i

(:,:,1,4) =
0.0506 + 0.2458i  -0.0502 + 0.0861i  0.0155 + 0.0515i
-0.3548 + 0.0833i  -0.0070 - 0.0800i  -0.0204 - 0.1147i

```

Original sequence as restored by inverse transform:

```

(:,:,1,1) =
1.0000 - 0.0000i  0.9940 - 0.1110i  0.9030 - 0.4300i
0.5000 + 0.5000i  0.4940 + 0.1110i  0.4030 + 0.4300i

(:,:,1,2) =
0.9990 - 0.0400i  0.9890 - 0.1510i  0.8850 - 0.4660i
0.4990 + 0.0400i  0.4890 + 0.1510i  0.3850 + 0.4660i

(:,:,1,3) =
0.9870 - 0.1590i  0.9630 - 0.2680i  0.8230 - 0.5680i
0.4870 + 0.1590i  0.4630 + 0.2680i  0.3230 + 0.5680i

(:,:,1,4) =
0.9360 - 0.3520i  0.8910 - 0.4540i  0.6940 - 0.7200i
0.4360 + 0.3520i  0.3910 + 0.4540i  0.1940 + 0.7200i

```

---