# NAG Toolbox for MATLAB

# c06px

## 1 Purpose

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

## 2 Syntax

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

## 3 Description

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

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

(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 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 1983b Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **direct – string**

If the **F**orward transform as defined in Section 3 is to be computed, then **direct** must be set equal to 'F'.

If the **B**ackward transform is to be computed then **direct** must be set equal to 'B'.

*Constraint*: **direct** = 'F' or 'B'.

2: **n1 – int32 scalar**

$n_1$, the first dimension of the transform.

*Constraint*: **n1** $\geq 1$.

3: **n2 – int32 scalar**

$n_2$, the second dimension of the transform.

*Constraint*: **n2** $\geq 1$.

4: **n3 – int32 scalar**

$n_3$, the third dimension of the transform.

*Constraint*: **n3** $\geq 1$.

5: **x**(**n1** $\times$ **n2** $\times$ **n3**) **– complex array**

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

### 5.2 Optional Input Parameters

None.

### 5.3 Input Parameters Omitted from the MATLAB Interface

work

### 5.4 Output Parameters

1: **x**(**n1** $\times$ **n2** $\times$ **n3**) **– complex array**

The corresponding elements of the computed transform.

2: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

On entry, **n1** $< 1$.

**ifail** $= 2$

On entry, **n2** $< 1$.

**ifail** $= 3$

On entry, **n3** $< 1$.

**ifail** $= 4$

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

**ifail** $= 5$

On entry, **n1** has more than 30 prime factors.

**ifail** $= 6$

On entry, **n2** has more than 30 prime factors.

**ifail** = 7

On entry, **n3** has more than 30 prime factors.

**ifail** = 8

An unexpected error has occurred in an internal call. Check all (sub)program calls and array dimensions. Seek expert help.

## 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 $n_3$. c06px is somewhat faster than average if their only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2.

## 9 Example

```
direct = 'F';
n1 = int32(2);
n2 = int32(3);
n3 = int32(4);
x = [complex(1, +0);
     complex(0.5, +0.5);
     complex(0.994, -0.111);
     complex(0.494, +0.111);
     complex(0.903, -0.43);
     complex(0.403, +0.43);
     complex(0.999, -0.04);
     complex(0.499, +0.04);
     complex(0.989, -0.151);
     complex(0.489, +0.151);
     complex(0.885, -0.466);
     complex(0.385, +0.466);
     complex(0.987, -0.159);
     complex(0.487, +0.159);
     complex(0.963, -0.268);
     complex(0.463, +0.268);
     complex(0.823, -0.5679999999999999);
     complex(0.323, +0.5679999999999999);
     complex(0.9360000000000001, -0.352);
     complex(0.436, +0.352);
     complex(0.891, -0.454);
     complex(0.391, +0.454);
     complex(0.694, -0.72);
     complex(0.194, +0.72)];
[xOut, ifail] = c06px(direct, n1, n2, n3, x)
```

```
xOut =
   3.2921 + 0.1021i
   1.2247 - 1.6203i
   0.1433 - 0.0860i
   0.4243 + 0.3197i
   0.1433 + 0.2902i
  -0.4243 + 0.3197i
   0.0506 - 0.0416i
   0.3548 + 0.0833i
   0.0155 + 0.1527i
   0.0204 - 0.1147i
  -0.0502 + 0.1180i
```

```
   0.0070 - 0.0800i
   0.1127 + 0.1021i
        0 + 0.1621i
  -0.0245 + 0.1268i
   0.0134 - 0.0914i
  -0.0245 + 0.0773i
  -0.0134 - 0.0914i
   0.0506 + 0.2458i
  -0.3548 + 0.0833i
  -0.0502 + 0.0861i
  -0.0070 - 0.0800i
   0.0155 + 0.0515i
  -0.0204 - 0.1147i
ifail =
           0
```