NAG Toolbox for MATLAB

c06fx

1 Purpose

c06fx computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values. This function is designed to be particularly efficient on vector processors.

2 Syntax

```
[x, y, trign1, trign2, trign3, ifail] = c06fx(n1, n2, n3, x, y, init,
trign1, trign2, trign3)
```

3 Description

c06fx computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values $z_{j_1 j_2 j_3}$, where $j_1 = 0, 1, \ldots, n_1 - 1$, $j_2 = 0, 1, \ldots, n_2 - 1$, $j_3 = 0, 1, \ldots, n_3 - 1$.

The discrete Fourier transform is here defined by

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

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

(Note the scale factor of $\frac{1}{\sqrt{n_1 n_2 n_3}}$ in this definition.)

To compute the inverse discrete Fourier transform, defined with $\exp(+2\pi i(\ldots))$ in the above formula instead of $\exp(-2\pi i(\ldots))$, this function should be preceded and followed by calls of c06gc to form the complex conjugates of the data values and the transform.

This function calls c06fr to perform multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm (see Brigham 1974). It is designed to be particularly efficient on vector processors.

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: **n1 – int32 scalar**

n_1, the first dimension of the transform.

Constraint: **n1** ≥ 1.

2: **n2 – int32 scalar**

n_2, the second dimension of the transform.

Constraint: **n2** ≥ 1.

3: **n3 – int32 scalar**

n_3, the third dimension of the transform.

Constraint: **n3** ≥ 1.

4: **x**(**n1** $\times$ **n2** $\times$ **n3**) **– double array**

5: **y**(**n1** $\times$ **n2** $\times$ **n3**) **– double array**

The real and imaginary parts of the complex data values must be stored in arrays **x** and **y** respectively. If **x** and **y** are regarded as three-dimensional arrays of dimension $(0 : \mathbf{n1} - 1, 0 : \mathbf{n2} - 1, 0 : \mathbf{n3} - 1)$, then $\mathbf{x}(j_1,j_2,j_3)$ and $\mathbf{y}(j_1,j_2,j_3)$ must contain the real and imaginary parts of $z_{j_1j_2j_3}$.

6: **init – string**

If the trigonometric coefficients required to compute the transforms are to be calculated by the function and stored in the arrays **trign1**, **trign2** and **trign3**, then **init** must be set equal to 'I', (**I**nitial call).

If **init** = 'S' (**S**ubsequent call), the function assumes that trigonometric coefficients for the specified values of n_1, n_2 and n_3 are supplied in the arrays **trign1**, **trign2** and **trign3**, having been calculated in a previous call to the function.

If **init** = 'R' (**R**estart), the function assumes that trigonometric coefficients for the specified values of n_1, n_2 and n_3 are supplied in the arrays **trign1**, **trign2** and **trign3**, but does not check that the function has previously been called. This option allows the **trign1**, **trign2** and **trign3** arrays to be stored in an external file, read in and re-used without the need for a call with **init** equal to 'I'. The function carries out a simple test to check that the current values of n_1, n_2 and n_3 are compatible with the arrays **trign1**, **trign2** and **trign3**.

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

7: **trign1**(**2** $\times$ **n1**) **– double array**

8: **trign2**(**2** $\times$ **n2**) **– double array**

9: **trign3**(**2** $\times$ **n3**) **– double array**

If **init** = 'S' or 'R', **trign1**, **trign2** and **trign3** must contain the required coefficients calculated in a previous call of the function. Otherwise **trign1**, **trign2** and **trign3** need not be set. If **n1** = **n2**, the same array may be supplied for **trign1** and **trign2**. Similar considerations apply if **n2** = **n3** or **n1** = **n3**.

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**) **– double array**

2: **y**(**n1** $\times$ **n2** $\times$ **n3**) **– double array**

The real and imaginary parts respectively of the corresponding elements of the computed transform.

3: **trign1**(**2** $\times$ **n1**) **– double array**

4: **trign2**(**2** $\times$ **n2**) **– double array**

5: **trign3**(**2** $\times$ **n3**) **– double array**

trign1, **trign2** and **trign3** contain the required coefficients (computed by the function if **init** = 'I').

6: **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, **init** $\neq$ 'I', 'S' or 'R'.

ifail = 5

Not used at this Mark.

ifail = 6

On entry, **init** = 'S' or 'R', but at least one of the arrays **trign1**, **trign2** and **trign3** is inconsistent with the current value of **n1**, **n2** or **n3**.

ifail = 7

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. c06fx 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

```
n1 = int32(2);
n2 = int32(3);
n3 = int32(4);
x = [1;
     0.5;
     0.994;
     0.494;
     0.903;
     0.403;
     0.999;
```

```
     0.499;
     0.989;
     0.489;
     0.885;
     0.385;
     0.987;
     0.487;
     0.963;
     0.463;
     0.823;
     0.323;
     0.9360000000000001;
     0.436;
     0.891;
     0.391;
     0.694;
     0.194];
y = [0;
     0.5;
     -0.111;
     0.111;
     -0.43;
     0.43;
     -0.04;
     0.04;
     -0.151;
     0.151;
     -0.466;
     0.466;
     -0.159;
     0.159;
     -0.268;
     0.268;
     -0.5679999999999999;
     0.5679999999999999;
     -0.352;
     0.352;
     -0.454;
     0.454;
     -0.72;
     0.72];
init = 'Initial';
trign1 = zeros(4,1);
trign2 = zeros(6,1);
trign3 = zeros(8,1);
[xOut, yOut, trign1Out, trign2Out, trign3Out, ifail] = ...
    c06fx(n1, n2, n3, x, y, init, trign1, trign2, trign3)
```

```
xOut =
    3.2921
    1.2247
    0.1433
    0.4243
    0.1433
   -0.4243
    0.0506
    0.3548
    0.0155
    0.0204
   -0.0502
    0.0070
    0.1127
   -0.0000
   -0.0245
    0.0134
   -0.0245
   -0.0134
    0.0506
   -0.3548
   -0.0502
```

```
   -0.0070
    0.0155
   -0.0204
yOut =
    0.1021
   -1.6203
   -0.0860
    0.3197
    0.2902
    0.3197
   -0.0416
    0.0833
    0.1527
   -0.1147
    0.1180
   -0.0800
    0.1021
    0.1621
    0.1268
   -0.0914
    0.0773
   -0.0914
    0.2458
    0.0833
    0.0861
   -0.0800
    0.0515
   -0.1147
trign1Out =
     1
     2
     0
     2
trign2Out =
     1
     1
     3
     0
     0
     3
trign3Out =
     1
     1
     1
     4
     0
     0
     0
     4
ifail =
           0
```