NAG Toolbox for MATLAB

c06fr

1 Purpose

c06fr computes the discrete Fourier transforms of m sequences, each containing n complex data values. This function is designed to be particularly efficient on vector processors.

2 Syntax

```
[x, y, trig, ifail] = c06fr(m, n, x, y, init, trig)
```

3 Description

Given m sequences of n complex data values z_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, c06fr simultaneously calculates the Fourier transforms of all the sequences defined by

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

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

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \times \exp\left(+i\frac{2\pi jk}{n}\right).$$

To compute this form, this function should be preceded and followed by a call of c06gc to form the complex conjugates of the z_j^p and the $\hat{z}_k^p$.

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 1983b. Special code is provided for the factors 2, 3, 4, 5 and 6. This function is designed to be particularly efficient on vector processors, and it becomes especially fast as m, the number of transforms to be computed in parallel, increases.

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: **m – int32 scalar**

m, the number of sequences to be transformed.

Constraint: $\mathbf{m} \geq 1$.

2: **n – int32 scalar**

n, the number of complex values in each sequence.

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

3: **x**(**m** × **n**) – **double array**

4: **y**(**m** × **n**) – **double array**

The real and imaginary parts of the complex data must be stored in **x** and **y** respectively 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 real parts of the pth sequence to be transformed are denoted by x_j^p, for $j = 0, 1, \ldots, n - 1$, then the mn elements of the array **x** must contain the values

$$x_0^1, x_0^2, \ldots, x_0^m, x_1^1, x_1^2, \ldots, x_1^m, \ldots, x_{n-1}^1, x_{n-1}^2, \ldots, x_{n-1}^m.$$

5: **init** – **string**

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

If **init** = 'S' (**S**ubsequent call), then the function assumes that trigonometric coefficients for the specified value of n are supplied in the array **trig**, having been calculated in a previous call to one of c06fp, c06fq or c06fr.

If **init** = 'R' (**R**estart), the function assumes that trigonometric coefficients for the specified value of n are supplied in the array **trig**, but does not check that c06fp, c06fq or c06fr have previously been called. This option allows the **trig** array 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 value of n is consistent with the value used to generate the array **trig**.

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

6: **trig**(**2** × **n**) – **double array**

If **init** = 'S' or 'R', **trig** must contain the required coefficients calculated in a previous call of the function. Otherwise **trig** need not be set.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

1: **x**(**m** × **n**) – **double array**

2: **y**(**m** × **n**) – **double array**

x and **y** are overwritten by the real and imaginary parts of the complex transforms.

3: **trig**(**2** × **n**) – **double array**

Contains the required coefficients (computed by the function if **init** = 'I').

4: **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, **m** < 1.

ifail = 2

On entry, **n** < 1.

ifail = 3

On entry, **init** ≠ 'I', 'S' or 'R'.

ifail = 4

Not used at this Mark.

ifail = 5

On entry, **init** = 'S' or 'R', but the array **trig** and the current value of **n** are inconsistent.

ifail = 6

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 by c06fr is approximately proportional to $nm \log n$, but also depends on the factors of n. c06fr 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

```
m = int32(3);
n = int32(6);
x = [0.3854;
     0.9172;
     0.1156;
     0.6772;
     0.0644;
     0.06850000000000001;
     0.1138;
     0.6037;
     0.206;
     0.6751;
     0.643;
     0.863;
     0.6362;
     0.0428;
     0.6967;
     0.1424;
     0.4815;
     0.2792];
y = [0.5417;
     0.9089;
     0.6214;
     0.2983;
     0.3118;
     0.8681;
     0.1181;
     0.3465;
     0.706;
```

```
     0.7255;
     0.6198;
     0.8652;
     0.8638;
     0.2668;
     0.919;
     0.8723;
     0.1614;
     0.3355];
init = 'Initial';
trig = zeros(12,1);
[xOut, yOut, trigOut, ifail] = c06fr(m, n, x, y, init, trig)
```

```
xOut =
    1.0737
    1.1237
    0.9100
   -0.5706
    0.1728
   -0.3054
    0.1733
    0.4185
    0.4079
   -0.1467
    0.1530
   -0.0785
    0.0518
    0.3686
   -0.1193
    0.3625
    0.0101
   -0.5314
yOut =
    1.3961
    1.0677
    1.7617
   -0.0409
    0.0386
    0.0624
   -0.2958
    0.7481
   -0.0695
   -0.1521
    0.1752
    0.0725
    0.4517
    0.0565
    0.1285
   -0.0321
    0.1403
   -0.4335
trigOut =
     1
     1
     1
     1
     1
     6
     0
     0
     0
     0
     0
     6
ifail =
           0
```