NAG Toolbox for MATLAB

c06fq

1 Purpose

c06fq computes the discrete Fourier transforms of m Hermitian sequences, each containing n complex data values. This function is designed to be particularly efficient on vector processors.

2 Syntax

```
[x, trig, ifail] = c06fq(m, n, x, init, trig)
```

3 Description

Given m Hermitian sequences of n complex data values z_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, c06fq simultaneously calculates the Fourier transforms of all the sequences defined by

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

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

The transformed values are purely real (see also the C06 Chapter Introduction).

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

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

To compute this form, this function should be preceded by a call to c06gq to form the complex conjugates of the $\hat{z}_j^p$.

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 1983a. Special coding 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 1983a Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

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 data values in each sequence.

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

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

The 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 the array in Hermitian form. If the n data values z_j^p are written as $x_j^p + iy_j^p$, then for $0 \le j \le n/2$, x_j^p is contained in $\mathbf{x}(p,j)$, and for $1 \le j \le (n-1)/2$, y_j^p is contained in $\mathbf{x}(p, n-j)$. (See also Section missing entity c06background12 in the C06 Chapter Introduction.)

4: **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'.

5: **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**

The components of the m discrete Fourier transforms, stored as if in a two-dimensional array of dimension $(1:\mathbf{m}, 0:\mathbf{n}-1)$. Each of the m transforms is stored as a **row** of the array, overwriting the corresponding original sequence. If the n components of the discrete Fourier transform are denoted by $\hat{x}_k^p$, for $k = 0, 1, \ldots, n-1$, then the mn elements of the array **x** contain the values

$$\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^m, \hat{x}_1^1, \hat{x}_1^2, \ldots, \hat{x}_1^m, \ldots, \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \ldots, \hat{x}_{n-1}^m.$$

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

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

3: **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** $\neq$ '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 c06fq is approximately proportional to $nm \log n$, but also depends on the factors of n. c06fq 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;
     0.5417;
     0.9172;
     0.6772;
     0.2983;
     0.0644;
     0.1138;
     0.1181;
     0.6037;
     0.6751;
     0.7255;
     0.643;
     0.6362;
     0.8638;
     0.0428;
     0.1424;
     0.8723;
     0.4815];
init = 'Initial';
```

```
trig = [0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0];
[xOut, trigOut, ifail] = c06fq(m, n, x, init, trig)
```

```
xOut =
    1.0788
    0.8573
    1.1825
    0.6623
    1.2261
    0.2625
   -0.2391
    0.3533
    0.6744
   -0.5783
   -0.2222
    0.5523
    0.4592
    0.3413
    0.0540
   -0.4388
   -1.2291
   -0.4790
trigOut =
     1
     1
     1
     1
     1
     6
     0
     0
     0
     0
     0
     6
ifail =
           0
```