NAG Toolbox for MATLAB

c06pq

1 Purpose

c06pq computes the discrete Fourier transforms of m sequences, each containing n real data values or a Hermitian complex sequence stored column-wise in a complex storage format.

2 Syntax

```
[x, ifail] = c06pq(direct, n, m, x)
```

3 Description

Given m sequences of n real data values x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, c06pq simultaneously calculates the Fourier transforms of all the sequences defined by

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

The transformed values $\hat{z}_k^p$ are complex, but for each value of p the $\hat{z}_k^p$ form a Hermitian sequence (i.e., $\hat{z}_{n-k}^p$ is the complex conjugate of $\hat{z}_k^p$), so they are completely determined by mn real numbers (since $\hat{z}_0^p$ is real, as is $\hat{z}_{n/2}^p$ for n even).

Alternatively, given m Hermitian sequences of n complex data values z_j^p, this function simultaneously calculates their inverse (**backward**) discrete Fourier transforms defined by

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

The transformed values $\hat{x}_k^p$ are real.

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

A call of c06pq with **direct** $=$ 'F' followed by a call with **direct** $=$ 'B' will restore the original data.

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 and 5.

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: **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: **n – int32 scalar**

n, the number of real or complex values in each sequence.

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

3: **m – int32 scalar**

m, the number of sequences to be transformed.

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

4: $\mathbf{x}((\mathbf{n}+2) \times \mathbf{m})$ **– double array**

The data must be stored in **x** as if in a two-dimensional array of dimension $(0 : \mathbf{n}+1, 1 : \mathbf{m})$; each of the m sequences is stored in a **column** of the array. In other words, if the data values of the pth sequence to be transformed are denoted by x_j^p, for $j = 0, 1, \ldots, n-1$, then:

if **direct** = 'F', $\mathbf{x}((p-1) \times (\mathbf{n}+2) + j)$ must contain x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$;
if **direct** = 'B', $\mathbf{x}((p-1) \times (\mathbf{n}+2) + 2 \times k)$ and $\mathbf{x}((p-1) \times (\mathbf{n}+2) + 2 \times k + 1)$ must contain the real and imaginary parts respectively of $\hat{z}_k^p$, for $k = 0, 1, \ldots, n/2$ and $p = 1, 2, \ldots, m$. (Note that for the sequence $\hat{z}_k^p$ to be Hermitian, the imaginary part of $\hat{z}_0^p$, and of $\hat{z}_{n/2}^p$ for n even, must be zero.)

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

1: $\mathbf{x}((\mathbf{n}+2) \times \mathbf{m})$ **– double array**

if **direct** = 'F' and **x** is declared with bounds $(0 : \mathbf{n}+1, 1 : \mathbf{m})$ then $\mathbf{x}(2 \times k, p)$ and $\mathbf{x}(2 \times k + 1, p)$ will contain the real and imaginary parts respectively of $\hat{z}_k^p$, for $k = 0, 1, \ldots, n/2$ and $p = 1, 2, \ldots, m$;
if **direct** = 'B' and **x** is declared with bounds $(0 : \mathbf{n}+1, 1 : \mathbf{m})$ then $\mathbf{x}(j, p)$ will contain x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$.

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, $\mathbf{m} < 1$.

ifail = 2

On entry, $\mathbf{n} < 1$.

ifail = 3

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

ifail = 4

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 c06pq is approximately proportional to $nm \log n$, but also depends on the factors of n. c06pq 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

```
direct = 'F';
n = int32(6);
m = int32(3);
x = [0.3854;
     0.6772;
     0.1138;
     0.6751;
     0.6362;
     0.1424;
     0;
     0;
     0.5417;
     0.2983;
     0.1181;
     0.7255;
     0.8638;
     0.8723;
     0;
     0;
     0.9172;
     0.0644;
     0.6037;
     0.643;
     0.0428;
     0.4815;
     0;
     0];
[xOut, ifail] = c06pq(direct, n, m, x)
```

```
xOut =
    1.0737
         0
   -0.1041
   -0.0044
    0.1126
   -0.3738
   -0.1467
         0
    1.3961
         0
   -0.0365
    0.4666
    0.0780
   -0.0607
   -0.1521
         0
```

```
    1.1237
         0
    0.0914
   -0.0508
    0.3936
    0.3458
    0.1530
         0
ifail =
           0
```