NAG Toolbox for MATLAB

c06pk

1 Purpose

c06pk calculates the circular convolution or correlation of two complex vectors of period n.

2 Syntax

```
[x, y, ifail] = c06pk(job, x, y, 'n', n)
```

3 Description

c06pk computes:

if **job** $= 1$, the discrete **convolution** of x and y, defined by

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j} = \sum_{j=0}^{n-1} x_{k-j} y_j;$$

if **job** $= 2$, the discrete **correlation** of x and y defined by

$$w_k = \sum_{j=0}^{n-1} \bar{x}_j y_{k+j}.$$

Here x and y are complex vectors, assumed to be periodic, with period n, i.e., $x_j = x_{j\pm n} = x_{j\pm 2n} = \ldots$; z and w are then also periodic with period n.

Note: this usage of the terms 'convolution' and 'correlation' is taken from Brigham 1974. The term 'convolution' is sometimes used to denote both these computations.

If $\hat{x}$, $\hat{y}$, $\hat{z}$ and $\hat{w}$ are the discrete Fourier transforms of these sequences, and $\tilde{x}$ is the inverse discrete Fourier transform of the sequence x_j, i.e.,

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

and

$$\tilde{x}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \times \exp\left(i\frac{2\pi jk}{n}\right),$$

then $\hat{z}_k = \sqrt{n}.\hat{x}_k\hat{y}_k$ and $\hat{w}_k = \sqrt{n}.\bar{\hat{x}}_k\hat{y}_k$ (the bar denoting complex conjugate).

This function calls the same auxiliary functions as c06pc to compute discrete Fourier transforms.

4 References

Brigham E O 1974 *The Fast Fourier Transform* Prentice–Hall

5 Parameters

5.1 Compulsory Input Parameters

1: **job – int32 scalar**

The computation to be performed:

job $= 1$

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j} \text{ (convolution);}$$

job $= 2$

$$w_k = \sum_{j=0}^{n-1} \bar{x}_j y_{k+j} \text{ (correlation).}$$

Constraint: **job** $= 1$ or 2.

2: **x(n) – complex array**

The elements of one period of the vector x. If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06pk is called, then $\mathbf{x}(j)$ must contain x_j, for $j = 0, 1, \ldots, n-1$.

3: **y(n) – complex array**

The elements of one period of the vector y. If **y** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06pk is called, then $\mathbf{y}(j)$ must contain y_j, for $j = 0, 1, \ldots, n-1$.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

n, the number of values in one period of the vectors **x** and **y**. The total number of prime factors of **n**, counting repetitions, must not exceed 30.

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

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

1: **x(n) – complex array**

The corresponding elements of the discrete convolution or correlation.

2: **y(n) – complex array**

The discrete Fourier transform of the convolution or correlation returned in the array **x**.

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, $\mathbf{n} < 1$.

ifail $= 2$

On entry, **job** $\neq 1$ or 2.

ifail = 3

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

ifail = 4

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

7 Accuracy

The results should be accurate to within a small multiple of the ***machine precision***.

8 Further Comments

The time taken is approximately proportional to $n \times \log n$, but also depends on the factorization of n. c06pk is faster if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

9 Example

```
job = int32(1);
x = [complex(1, -0.5);
     complex(1, -0.5);
     complex(1, -0.5);
     complex(1, -0.5);
     complex(1, -0.5);
     complex(0, -0.5);
     complex(0, -0.5);
     complex(0, -0.5);
     complex(0, -0.5)];
y = [complex(0.5, -0.25);
     complex(0.5, -0.25);
     complex(0.5, -0.25);
     complex(0.5, -0.25);
     complex(0, -0.25);
     complex(0, -0.25);
     complex(0, -0.25);
     complex(0, -0.25);
     complex(0, -0.25)];
[xOut, yOut, ifail] = c06pk(job, x, y)
```

```
xOut =
  -0.6250 - 2.2500i
  -0.1250 - 2.2500i
   0.3750 - 2.2500i
   0.8750 - 2.2500i
   0.8750 - 2.2500i
   0.3750 - 2.2500i
  -0.1250 - 2.2500i
  -0.6250 - 2.2500i
  -1.1250 - 2.2500i
yOut =
  -0.0417 - 6.7500i
  -1.0585 - 0.8882i
  -0.0082 - 0.0465i
   0.0833 - 0.1443i
   0.0667 - 0.0243i
   0.0667 + 0.0243i
   0.0833 + 0.1443i
  -0.0082 + 0.0465i
  -1.0585 + 0.8882i
ifail =
           0
```