NAG Toolbox for MATLAB

c06ec

1 Purpose

c06ec calculates the discrete Fourier transform of a sequence of n complex data values. (No extra workspace required.)

2 Syntax

```
[x, y, ifail] = c06ec(x, y, 'n', n)
```

3 Description

Given a sequence of n complex data values z_j, for $j = 0, 1, \ldots, n-1$, c06ec calculates their discrete Fourier transform defined by

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

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

To compute the inverse discrete Fourier transform defined by

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

this function should be preceded and followed by calls of c06gc to form the complex conjugates of the z_j and the $\hat{z}_k$.

c06ec uses the fast Fourier transform (FFT) algorithm (see Brigham 1974). There are some restrictions on the value of n (see Section 5).

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **x(n) – double array**

If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06ec is called, then $\mathbf{x}(j)$ must contain x_j, the real part of z_j, for $j = 0, 1, \ldots, n-1$.

2: **y(n) – double array**

If **y** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06ec is called, then $\mathbf{y}(j)$ must contain y_j, the imaginary part of z_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 data values. The largest prime factor of **n** must not exceed 19, and the total number of prime factors of **n**, counting repetitions, must not exceed 20.

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

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **x(n) – double array**

The real parts a_k of the components of the discrete Fourier transform. If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06ec is called, then for $0 \leq k \leq n - 1$, a_k is contained in $\mathbf{x}(k)$.

2: **y(n) – double array**

The imaginary parts b_k of the components of the discrete Fourier transform. If **y** is declared with bounds $(0 : \mathbf{n} - 1)$ in the (sub)program from which c06ec is called, then for $0 \leq k \leq n - 1$, b_k is contained in $\mathbf{y}(k)$.

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 $= 1$

At least one of the prime factors of **n** is greater than 19.

ifail $= 2$

n has more than 20 prime factors.

ifail $= 3$

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

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 is approximately proportional to $n \times \log n$, but also depends on the factorization of n. c06ec 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.

On the other hand, c06ec is particularly slow if n has several unpaired prime factors, i.e., if the 'square-free' part of n has several factors. For such values of n, c06fc (which requires an additional n double elements of workspace) is considerably faster.

9 Example

```
x = [0.34907;
     0.5489000000000001;
     0.74776;
     0.94459;
     1.1385;
     1.3285;
     1.5137];
y = [-0.37168;
     -0.35669;
     -0.31175;
     -0.23702;
     -0.13274;
     0.00074;
     0.16298];
[xOut, yOut, ifail] = c06ec(x, y)
```

```
xOut =
    2.4836
   -0.5518
   -0.3671
   -0.2877
   -0.2251
   -0.1483
    0.0198
yOut =
   -0.4710
    0.4968
    0.0976
   -0.0586
   -0.1748
   -0.3084
   -0.5650
ifail =
           0
```