NAG Toolbox for MATLAB

c06rd

1 Purpose

c06rd computes the discrete quarter-wave Fourier cosine transforms of m sequences of real data values.

2 Syntax

```
[x, ifail] = c06rd(direct, m, n, x)
```

3 Description

Given m sequences of n real data values x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, c06rd simultaneously calculates the quarter-wave Fourier cosine transforms of all the sequences defined by

$$\hat{x}_k^p = \frac{1}{\sqrt{n}}\left(\tfrac{1}{2}x_0^p + \sum_{j=1}^{n-1} x_j^p \times \cos\left(j(2k-1)\frac{\pi}{2n}\right)\right), \qquad \text{if } \textbf{direct} = \text{'F'},$$

or its inverse

$$x_k^p = \frac{2}{\sqrt{n}}\sum_{j=0}^{n-1} \hat{x}_j^p \times \cos\left((2j-1)k\frac{\pi}{2n}\right), \qquad \text{if } \textbf{direct} = \text{'B'},$$

for $k = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$.

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

A call of c06rd with **direct** = 'F' followed by a call with **direct** = 'B' will restore the original data.

The transform calculated by this function can be used to solve Poisson's equation when the derivative of the solution is specified at the left boundary, and the solution is specified at the right boundary (see Swarztrauber 1977).

The function uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham 1974) known as the Stockham self-sorting algorithm, described in Temperton 1983a, together with pre- and post-processing stages described in Swarztrauber 1982. Special coding is provided for the factors 2, 3, 4 and 5.

4 References

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

Swarztrauber P N 1977 The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle *SIAM Rev.* **19 (3)** 490–501

Swarztrauber P N 1982 Vectorizing the FFT's *Parallel Computation* (ed G Rodrique) 51–83 Academic Press

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: **m – int32 scalar**

m, the number of sequences to be transformed.

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

3: **n – int32 scalar**

n, the number of real values in each sequence.

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

4: **x**$(\mathbf{m} \times (\mathbf{n}+\mathbf{2}))$ **– 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 other words, if the data values of the pth sequence to be transformed are denoted by x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, then the first mn elements of the array **x** must contain the values

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

The $(n+1)$th and $(n+2)$th elements of each row x_n^p, x_{n+1}^p, for $p = 1, 2, \ldots, m$, are required as workspace. These $2m$ elements may contain arbitrary values as they are set to zero by the function.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

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

The m quarter-wave cosine 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 in a **row** of the array, overwriting the corresponding original sequence. If the n components of the pth quarter-wave cosine transform are denoted by $\hat{x}_k^p$, for $k = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, then the $m(n+2)$ elements of the array **x** contain the values

$$\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^m, \hat{x}_1^1, \hat{x}_1^2, \ldots, \hat{x}_1^m, \ldots, \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \ldots, \hat{x}_{n-1}^m, 0, 0, \ldots, 0 (2m \text{ times}).$$

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** ≠ '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 c06rd is approximately proportional to $nm \log n$, but also depends on the factors of n. c06rd 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 = 'Forward';
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;
     0;
     0;
     0;
     0;
     0;
     0];
[xOut, ifail] = c06rd(direct, m, n, x)
```

```
xOut =
    0.7257
    0.7479
    0.6713
   -0.2216
   -0.6172
   -0.1363
    0.1011
    0.4112
   -0.0064
    0.2355
    0.0791
   -0.0285
```

```
   -0.1406
    0.1331
    0.4758
   -0.2282
   -0.0906
    0.1475
   -0.4564
   -0.1812
    0.2950
         0
         0
         0
ifail =
           0
```