NAG Toolbox for MATLAB

c06rb

1 Purpose

c06rb computes the discrete Fourier cosine transforms of m sequences of real data values.

2 Syntax

```
[x, ifail] = c06rb(m, n, x)
```

3 Description

Given m sequences of $n+1$ real data values x_j^p, for $j = 0, 1, \ldots, n$ and $p = 1, 2, \ldots, m$, c06rb simultaneously calculates the Fourier cosine transforms of all the sequences defined by

$$\hat{x}_k^p = \sqrt{\frac{2}{n}}\left(\frac{1}{2}x_0^p + \sum_{j=1}^{n-1} x_j^p \times \cos\left(jk\frac{\pi}{n}\right) + \frac{1}{2}-1^k x_n^p\right), \qquad k = 0, 1, \ldots, n; \qquad p = 1, 2, \ldots, m.$$

(Note the scale factor $\sqrt{\frac{2}{n}}$ in this definition.)

Since the Fourier cosine transform is its own inverse, two consecutive calls of this function 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 both left and right boundaries (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: **m – int32 scalar**

m, the number of sequences to be transformed.

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

2: **n – int32 scalar**

One less than the number of real values in each sequence, i.e., the number of values in each sequence is $n+1$.

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

3: **x(m × (n + 3)) – double array**

The data must be stored in **x** as if in a two-dimensional array of dimension $(1 : \mathbf{m}, 0 : \mathbf{n} + 2)$; each of the m sequences is stored in a **row** of the array. In other words, if the $(n + 1)$ data values of the pth sequence to be transformed are denoted by x_j^p, for $j = 0, 1, \ldots, n$ and $p = 1, 2, \ldots, m$, then the first $m(n + 1)$ elements of the array **x** must contain the values

$$x_0^1, x_0^2, \ldots, x_0^m, x_1^1, x_1^2, \ldots, x_1^m, \ldots, x_n^1, x_n^2, \ldots, x_n^m.$$

The $(n + 2)$th and $(n + 3)$th elements of each row x_{n+2}^p, x_{n+3}^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(m × (n + 3)) – double array**

The m Fourier cosine transforms stored as if in a two-dimensional array of dimension $(1 : \mathbf{m}, 0 : \mathbf{n} + 2)$. Each of the m transforms is stored in a **row** of the array, overwriting the corresponding original data. If the $(n + 1)$ components of the pth Fourier cosine transform are denoted by $\hat{x}_k^p$, for $k = 0, 1, \ldots, n$ and $p = 1, 2, \ldots, m$, then the $m(n + 3)$ elements of the array **x** contain the values

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

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

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 c06rb is approximately proportional to $nm \log n$, but also depends on the factors of n. c06rb 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;
     0.9562;
     0.4936;
     0.2057;
     0;
     0;
     0;
     0;
     0;
     0];
[xOut, ifail] = c06rb(m, n, x)
```

```
xOut =
    1.6833
    1.9605
    1.3838
   -0.0482
   -0.4884
    0.1588
    0.0176
   -0.0655
   -0.0761
    0.1368
    0.4444
   -0.1184
    0.3240
    0.0964
    0.3512
   -0.5830
    0.0856
    0.5759
   -0.0427
   -0.2289
    0.0110
         0
         0
         0
   -0.0482
   -0.4884
```

```
    0.1588
ifail =
           0
```