NAG Library Routine Document

C06RBF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

C06RBF computes the discrete Fourier cosine transforms of m sequences of real data values.

2 Specification

```
SUBROUTINE C06RBF (M, N, X, WORK, IFAIL)

INTEGER             M, N, IFAIL
REAL (KIND=nag_wp) X(M*(N+3)), WORK(*)
```

3 Description

Given m sequences of $n+1$ real data values x_j^p, for $j=0,1,\ldots,n$ and $p=1,2,\ldots,m$, C06RBF simultaneously calculates the Fourier cosine transforms of all the sequences defined by

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

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

Since the Fourier cosine transform is its own inverse, two consecutive calls of C06RBF will restore the original data.

The transform calculated by this routine 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 routine uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, described in Temperton (1983), 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 (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

5 Parameters

1: M – INTEGER *Input*

On entry: m, the number of sequences to be transformed.

Constraint: $M \geq 1$.

2: N – INTEGER *Input*

On entry: one less than the number of real values in each sequence, i.e., the number of values in each sequence is $n+1$.

Constraint: $\mathrm{N} \geq 1$.

3: $\mathrm{X}(\mathrm{M} \times (\mathrm{N}+3))$ – REAL (KIND=nag_wp) array *Input/Output*

On entry: the data must be stored in X as if in a two-dimensional array of dimension $(1:\mathrm{M}, 0:\mathrm{N}+2)$; 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 routine.

On exit: the m Fourier cosine transforms stored as if in a two-dimensional array of dimension $(1:\mathrm{M}, 0:\mathrm{N}+2)$. 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

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

4: $\mathrm{WORK}(*)$ – REAL (KIND=nag_wp) array *Workspace*

Note: the dimension of the array WORK must be at least $\mathrm{M} \times \mathrm{N} + 2 \times \mathrm{N} + 15$.

The workspace requirements as documented for C06RBF may be an overestimate in some implementations.

On exit: $\mathrm{WORK}(1)$ contains the minimum workspace required for the current values of M and N with this implementation.

5: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, $\mathrm{M} < 1$.

$\mathrm{IFAIL} = 2$

On entry, $\mathrm{N} < 1$.

IFAIL = 3

An unexpected error has occurred in an internal call. Check all subroutine 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 C06RBF is approximately proportional to $nm \log n$, but also depends on the factors of n. C06RBF 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

This example reads in sequences of real data values and prints their Fourier cosine transforms (as computed by C06RBF). It then calls the routine again and prints the results which may be compared with the original sequence.

9.1 Program Text

```
    Program c06rbfe

!     C06RBF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: c06rbf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ieof, ifail, j, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06RBF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) m, n
        If (ieof<0) Exit loop

        Allocate (x(m*(n+3)),work(m*n+2*n+15))
        Do j = 1, m
          Read (nin,*)(x(i*m+j),i=0,n)
        End Do
        Write (nout,*)
        Write (nout,*) 'Original data values'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
!       -- Compute transform
        Call c06rbf(m,n,x,work,ifail)

        Write (nout,*)
```

```
        Write (nout,*) 'Discrete Fourier cosine transforms'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do

!       -- Compute inverse transform
        Call c06rbf(m,n,x,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Original data as restored by inverse transform'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x(i*m+j),i=0,n)
        End Do
        Deallocate (x,work)
      End Do loop

99999 Format (6X,7F10.4)
    End Program c06rbfe
```

9.2 Program Data

```
C06RBF Example Program Data
3  6                                                          : m, n
 0.3854  0.6772  0.1138  0.6751  0.6362  0.1424  0.9562
 0.5417  0.2983  0.1181  0.7255  0.8638  0.8723  0.4936
 0.9172  0.0644  0.6037  0.6430  0.0428  0.4815  0.2057   : x
```

9.3 Program Results

```
 C06RBF Example Program Results

 Original data values

          0.3854    0.6772    0.1138    0.6751    0.6362    0.1424    0.9562
          0.5417    0.2983    0.1181    0.7255    0.8638    0.8723    0.4936
          0.9172    0.0644    0.6037    0.6430    0.0428    0.4815    0.2057

 Discrete Fourier cosine transforms

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

 Original data as restored by inverse transform

          0.3854    0.6772    0.1138    0.6751    0.6362    0.1424    0.9562
          0.5417    0.2983    0.1181    0.7255    0.8638    0.8723    0.4936
          0.9172    0.0644    0.6037    0.6430    0.0428    0.4815    0.2057
```