NAG Library Routine Document

C06RDF

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

C06RDF computes the discrete quarter-wave Fourier cosine transforms of m sequences of real data values.

2 Specification

```
SUBROUTINE C06RDF (DIRECT, M, N, X, WORK, IFAIL)

INTEGER              M, N, IFAIL
REAL (KIND=nag_wp) X(M*(N+2)), WORK(*)
CHARACTER(1)         DIRECT
```

3 Description

Given m sequences of n real data values x_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, C06RDF simultaneously calculates the quarter-wave Fourier cosine transforms of all the sequences defined by

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

or its inverse

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

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

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

A call of C06RDF with DIRECT = 'F' followed by a call with DIRECT = 'B' 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 the left boundary, and the solution is specified at the right boundary (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 Arguments

1: DIRECT – CHARACTER(1) *Input*

On entry: if the forward transform as defined in Section 3 is to be computed, then DIRECT must be set equal to `F'.

If the backward transform is to be computed then DIRECT must be set equal to `B'.

Constraint: DIRECT = 'F' or 'B'.

2: M – INTEGER *Input*

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

Constraint: $M \geq 1$.

3: N – INTEGER *Input*

On entry: n, the number of real values in each sequence.

Constraint: $N \geq 1$.

4: $X(M \times (N+2))$ – 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 $(1:M, 0:N+1)$; 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 routine.

On exit: the m quarter-wave cosine transforms stored as if in a two-dimensional array of dimension $(1:M, 0:N+1)$. 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

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

5: WORK($*$) – REAL (KIND=nag_wp) array *Workspace*

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

The workspace requirements as documented for C06RDF may be an overestimate in some implementations.

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

6: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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: 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 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:

IFAIL $= 1$

On entry, $M < 1$.

IFAIL $= 2$

On entry, $N < 1$.

IFAIL $= 3$

On entry, DIRECT $\neq$ 'F' or 'B'.

IFAIL $= 4$

An unexpected error has occurred in an internal call. Check all subroutine calls and array dimensions. Seek expert help.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 Parallelism and Performance

C06RDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06RDF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by C06RDF is approximately proportional to $nm\log(n)$, but also depends on the factors of n. C06RDF 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.

10 Example

This example reads in sequences of real data values and prints their quarter-wave cosine transforms as computed by C06RDF with DIRECT = 'F'. It then calls the routine again with DIRECT = 'B' and prints the results which may be compared with the original data.

10.1 Program Text

```
    Program c06rdfe

!     C06RDF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c06rdf, 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,*) 'C06RDF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) m, n
        If (ieof<0) Then
          Exit loop
        End If

        Allocate (x(m*(n+2)),work(m*n+2*n+2*m+15))
        Do j = 1, m
          Read (nin,*)(x(i*m+j),i=0,n-1)
        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-1)
        End Do

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
!       -- Compute transform
        Call c06rdf('Forward',m,n,x,work,ifail)

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

!       -- Compute inverse transform
        Call c06rdf('Backward',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-1)
        End Do
        Deallocate (x,work)
      End Do loop

99999 Format (6X,7F10.4)
    End Program c06rdfe
```

10.2 Program Data

```
C06RDF Example Program Data
 3  6                                            : m, n
 0.3854  0.6772  0.1138  0.6751  0.6362  0.1424
 0.5417  0.2983  0.1181  0.7255  0.8638  0.8723
 0.9172  0.0644  0.6037  0.6430  0.0428  0.4815  : x
```

10.3 Program Results

```
 C06RDF Example Program Results

 Original data values

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

 Discrete quarter-wave Fourier cosine transforms

          0.7257   -0.2216    0.1011    0.2355   -0.1406   -0.2282
          0.7479   -0.6172    0.4112    0.0791    0.1331   -0.0906
          0.6713   -0.1363   -0.0064   -0.0285    0.4758    0.1475

 Original data as restored by inverse transform

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