NAG Library Routine Document

C06RAF

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

C06RAF computes the discrete Fourier sine transforms of m sequences of real data values.

2   Specification

```
SUBROUTINE C06RAF (M, N, X, WORK, IFAIL)
INTEGER          M, N, IFAIL
REAL (KIND=nag_wp) X(M*(N+2)), WORK(*)
```

3   Description

Given m sequences of $n-1$ real data values x_j^p, for $j = 1, 2, \ldots, n-1$ and $p = 1, 2, \ldots, m$, C06RAF simultaneously calculates the Fourier sine transforms of all the sequences defined by

$$\hat{x}_k^p = \sqrt{\tfrac{2}{n}} \sum_{j=1}^{n-1} x_j^p \times \sin\left(jk\frac{\pi}{n}\right), \quad k = 1, 2, \ldots, n-1 \text{ and } p = 1, 2, \ldots, m.$$

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

Since the Fourier sine transform defined above is its own inverse, two consecutive calls of this routine will restore the original data.

The transform calculated by this routine can be used to solve Poisson's equation when 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   Arguments

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 more than the number of real values in each sequence, i.e., the number of values in each sequence is $n - 1$.

Constraint: $N \geq 1$.

3:    $X(M \times (N + 2))$ – 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 $(1 : M, 1 : N + 2)$; 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 $j = 1, 2, \ldots, n - 1$ and $p = 1, 2, \ldots, m$, then the first $m(n - 1)$ elements of the array X must contain the values

$$x_1^1, x_1^2, \ldots, x_1^m, x_2^1, x_2^2, \ldots, x_2^m, \ldots, x_{n-1}^1, x_{n-1}^2, \ldots, x_{n-1}^m.$$

The nth to $(n + 2)$th elements of each row $x_n^p, \ldots, x_{n+2}^p$, for $p = 1, 2, \ldots, m$, are required as workspace. These $3m$ elements may contain arbitrary values as they are set to zero by the routine.

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

$$\hat{x}_1^1, \hat{x}_1^2, \ldots, \hat{x}_1^m, \hat{x}_2^1, \hat{x}_2^2, \ldots, \hat{x}_2^m, \ldots, \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \ldots, \hat{x}_{n-1}^m, 0, 0, \ldots, 0 \ (3m \text{ times}).$$

4:    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 C06RAF 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.

5:    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

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

C06RAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06RAF 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 C06RAF is approximately proportional to $nm\log(n)$, but also depends on the factors of n. C06RAF 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 Fourier sine transforms (as computed by C06RAF). It then calls C06RAF again and prints the results which may be compared with the original sequence.

10.1 Program Text

```
      Program c06rafe

!     C06RAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c06raf, 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,*) 'C06RAF 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-1)*m+j),i=1,n-1)
        End Do
        Write (nout,*)
        Write (nout,*) 'Original data values'
        Write (nout,*)
        Do j = 1, m
          Write (nout,99999)(x((i-1)*m+j),i=1,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 c06raf(m,n,x,work,ifail)

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

!       -- Compute inverse transform
        Call c06raf(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-1)*m+j),i=1,n-1)
        End Do
        Deallocate (x,work)
      End Do loop

99999 Format (6X,6F10.4)
      End Program c06rafe
```

10.2 Program Data

```
C06RAF Example Program Data
 3  6                                     : m, n
 0.6772  0.1138  0.6751  0.6362  0.1424
 0.2983  0.1182  0.7255  0.8638  0.8723
 0.0644  0.6037  0.6430  0.0428  0.4815  : x
```

10.3 Program Results

```
C06RAF Example Program Results

Original data values

        0.6772    0.1138    0.6751    0.6362    0.1424
        0.2983    0.1182    0.7255    0.8638    0.8723
        0.0644    0.6037    0.6430    0.0428    0.4815

Discrete Fourier sine transforms

        1.0014    0.0062    0.0834    0.5286    0.2514
        1.2478   -0.6598    0.2570    0.0858    0.2658
        0.8521    0.0719   -0.0561   -0.4890    0.2056

Original data as restored by inverse transform

        0.6772    0.1138    0.6751    0.6362    0.1424
        0.2983    0.1182    0.7255    0.8638    0.8723
        0.0644    0.6037    0.6430    0.0428    0.4815
```