NAG Library Routine Document

C06FRF

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

C06FRF computes the discrete Fourier transforms of m sequences, each containing n complex data values. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
SUBROUTINE C06FRF (M, N, X, Y, INIT, TRIG, WORK, IFAIL)

INTEGER             M, N, IFAIL
REAL (KIND=nag_wp)  X(M*N), Y(M*N), TRIG(2*N), WORK(2*M*N)
CHARACTER(1)        INIT
```

3 Description

Given m sequences of n complex data values z_j^p, for $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, C06FRF simultaneously calculates the Fourier transforms of all the sequences defined by

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \times \exp\left(-i\frac{2\pi jk}{n}\right), \qquad k = 0, 1, \ldots, n-1 \text{ and } p = 1, 2, \ldots, m.$$

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

The discrete Fourier transform is sometimes defined using a positive sign in the exponential term

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \times \exp\left(+i\frac{2\pi jk}{n}\right).$$

To compute this form, this routine should be preceded and followed by a call of C06GCF to form the complex conjugates of the z_j^p and the $\hat{z}_k^p$.

The routine uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, which is described in Temperton (1983). Special code is provided for the factors 2, 3, 4, 5 and 6. This routine is designed to be particularly efficient on vector processors, and it becomes especially fast as m, the number of transforms to be computed in parallel, increases.

4 References

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

Temperton C (1983) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

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: n, the number of complex values in each sequence.

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

3: X(M × N) – REAL (KIND=nag_wp) array *Input/Output*

4: Y(M × N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the real and imaginary parts of the complex data must be stored in X and Y respectively as if in a two-dimensional array of dimension $(1:\mathrm{M}, 0:\mathrm{N}-1)$; each of the m sequences is stored in a **row** of each array. In other words, if the real parts of the pth sequence to be transformed are denoted by x_j^p, for $j = 0, 1, \ldots, n-1$, then the mn elements of the array X must contain the values

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

On exit: X and Y are overwritten by the real and imaginary parts of the complex transforms.

5: INIT – CHARACTER(1) *Input*

On entry: indicates whether trigonometric coefficients are to be calculated.

INIT = 'I'
: Calculate the required trigonometric coefficients for the given value of n, and store in the array TRIG.

INIT = 'S' or 'R'
: The required trigonometric coefficients are assumed to have been calculated and stored in the array TRIG in a prior call to one of C06FPF, C06FQF or C06FRF. The routine performs a simple check that the current value of n is consistent with the values stored in TRIG.

Constraint: INIT = 'I', 'S' or 'R'.

6: TRIG(2 × N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if INIT = 'S' or 'R', TRIG must contain the required trigonometric coefficients that have been previously calculated. Otherwise TRIG need not be set.

On exit: contains the required coefficients (computed by the routine if INIT = 'I').

7: WORK(2 × M × N) – REAL (KIND=nag_wp) array *Workspace*

8: 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: 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, $\mathrm{M} < 1$.

IFAIL = 2

On entry, N < 1.

IFAIL = 3

On entry, INIT ≠ 'I', 'S' or 'R'.

IFAIL = 4

Not used at this Mark.

IFAIL = 5

On entry, INIT = 'S' or 'R', but the array TRIG and the current value of N are inconsistent.

IFAIL = 6

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 C06FRF is approximately proportional to $nm \log n$, but also depends on the factors of n. C06FRF 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 complex data values and prints their discrete Fourier transforms (as computed by C06FRF). Inverse transforms are then calculated using C06FRF and C06GCF and printed out, showing that the original sequences are restored.

9.1 Program Text

```
    Program c06frfe

!     C06FRF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: c06frf, c06gcf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
      Character (7), Parameter         :: init_first = 'Initial'
      Character (10), Parameter        :: init_repeat = 'Subsequent'
!     .. Local Scalars ..
      Integer                          :: i, ieof, ifail, lwork, m, mn, n, n2
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: trig(:), work(:), x(:,:), y(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'C06FRF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) m, n
        mn = m*n
        n2 = 2*n
        lwork = 2*mn
        If (ieof<0) Exit loop

        Allocate (trig(n2),x(m,n),y(m,n),work(lwork))
        Do i = 1, m
          Read (nin,*) x(i,1:n)
          Read (nin,*) y(i,1:n)
        End Do
        Write (nout,*)
        Write (nout,*) 'Original data values'
        Do i = 1, m
          Write (nout,*)
          Write (nout,99999) 'Real ', x(i,1:n)
          Write (nout,99999) 'Imag ', y(i,1:n)
        End Do

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call c06frf(m,n,x,y,init_first,trig,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Discrete Fourier transforms'
        Do i = 1, m
          Write (nout,*)
          Write (nout,99999) 'Real ', x(i,1:n)
          Write (nout,99999) 'Imag ', y(i,1:n)
        End Do

        Call c06gcf(y,mn,ifail)
        ifail = 0
        Call c06frf(m,n,x,y,init_repeat,trig,work,ifail)
        Call c06gcf(y,mn,ifail)

        Write (nout,*)
        Write (nout,*) 'Original data as restored by inverse transform'
        Do i = 1, m
          Write (nout,*)
          Write (nout,99999) 'Real ', x(i,1:n)
          Write (nout,99999) 'Imag ', y(i,1:n)
        End Do
        Deallocate (trig,x,y,work)
      End Do loop

99999 Format (1X,A,6F10.4)
    End Program c06frfe
```

9.2 Program Data

```
C06FRF 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
     0.9089    0.3118    0.3465    0.6198    0.2668    0.1614
     0.1156    0.0685    0.2060    0.8630    0.6967    0.2792
     0.6214    0.8681    0.7060    0.8652    0.9190    0.3355   : x, y
```

9.3 Program Results

```
 C06FRF Example Program Results

 Original data values

 Real      0.3854    0.6772    0.1138    0.6751    0.6362    0.1424
 Imag      0.5417    0.2983    0.1181    0.7255    0.8638    0.8723

 Real      0.9172    0.0644    0.6037    0.6430    0.0428    0.4815
 Imag      0.9089    0.3118    0.3465    0.6198    0.2668    0.1614
 Real     0.1156    0.0685    0.2060    0.8630    0.6967    0.2792
 Imag     0.6214    0.8681    0.7060    0.8652    0.9190    0.3355

 Discrete Fourier transforms

 Real     1.0737   -0.5706    0.1733   -0.1467    0.0518    0.3625
 Imag     1.3961   -0.0409   -0.2958   -0.1521    0.4517   -0.0321

 Real     1.1237    0.1728    0.4185    0.1530    0.3686    0.0101
 Imag     1.0677    0.0386    0.7481    0.1752    0.0565    0.1403

 Real     0.9100   -0.3054    0.4079   -0.0785   -0.1193   -0.5314
 Imag     1.7617    0.0624   -0.0695    0.0725    0.1285   -0.4335

 Original data as restored by inverse transform

 Real     0.3854    0.6772    0.1138    0.6751    0.6362    0.1424
 Imag     0.5417    0.2983    0.1181    0.7255    0.8638    0.8723

 Real     0.9172    0.0644    0.6037    0.6430    0.0428    0.4815
 Imag     0.9089    0.3118    0.3465    0.6198    0.2668    0.1614

 Real     0.1156    0.0685    0.2060    0.8630    0.6967    0.2792
 Imag     0.6214    0.8681    0.7060    0.8652    0.9190    0.3355
```