C06FPF – NAG Fortran Library Routine Document

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

C06FPF computes the discrete Fourier transforms of m sequences, each containing n real data values. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
SUBROUTINE C06FPF(M, N, X, INIT, TRIG, WORK, IFAIL)
INTEGER          M, N, IFAIL
real             X(M*N), TRIG(2*N), WORK(M*N)
CHARACTER*1      INIT
```

3 Description

Given m sequences of n real data values x_j^p, for $j = 0,1,\ldots,n-1$; $p = 1,2,\ldots,m$, this routine simultaneously calculates the Fourier transforms of all the sequences defined by:

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

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

The transformed values $\hat{z}_k^p$ are complex, but for each value of p the $\hat{z}_k^p$ form a Hermitian sequence (i.e., $\hat{z}_{n-k}^p$ is the complex conjugate of $\hat{z}_k^p$), so they are completely determined by mn real numbers (see also the Chapter Introduction).

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

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

To compute this form, this routine should be followed by a call to C06GQF to form the complex conjugates of the $\hat{z}_k^p$.

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [2]. Special coding 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

[1] Brigham E O (1973) *The Fast Fourier Transform* Prentice–Hall

[2] Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

5 Parameters

1: M — INTEGER *Input*

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

Constraint: $M \geq 1$.

2: N — INTEGER *Input*

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

Constraint: $N \geq 1$.

3: X(M∗N) — ***real*** 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 $j = 0, 1, \ldots, n-1$, 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: the m discrete Fourier 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 in Hermitian form, overwriting the corresponding original sequence. If the n components of the discrete Fourier transform $\hat{z}_k^p$ are written as $a_k^p + ib_k^p$, then for $0 \le k \le n/2$, a_k^p is contained in X(p,k), and for $1 \le k \le (n-1)/2$, b_k^p is contained in X$(p, n-k)$. (See also Section 2.1.2 of the Chapter Introduction.)

4: INIT — CHARACTER*1 *Input*

On entry: if the trigonometric coefficients required to compute the transforms are to be calculated by the routine and stored in the array TRIG, then INIT must be set equal to 'I' (**I**nitial call).

If INIT contains 'S' (**S**ubsequent call), then the routine assumes that trigonometric coefficients for the specified value of n are supplied in the array TRIG, having been calculated in a previous call to one of C06FPF, C06FQF or C06FRF.

If INIT contains 'R' (**R**estart) then the routine assumes that trigonometric coefficients for the particular value of n are supplied in the array TRIG, but does not check that C06FPF, C06FQF or C06FRF have previously been called. This option allows the TRIG array to be stored in an external file, read in and re-used without the need for a call with INIT equal to 'I'. The routine carries out a simple test to check that the current value of n is consistent with the array TRIG.

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

5: TRIG(2∗N) — ***real*** array *Input/Output*

On entry: if INIT = 'S' or 'R', TRIG must contain the required coefficients calculated in a previous call of the routine. Otherwise TRIG need not be set.

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

6: WORK(M∗N) — ***real*** array *Workspace*

7: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (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 detected by the routine:

IFAIL = 1

On entry, M < 1.

IFAIL = 2

N < 1.

IFAIL = 3

INIT is not one of 'I', 'S' or 'R'.

IFAIL = 4

INIT = 'S', but none of C06FPF, C06FQF or C06FRF have previously been called.

IFAIL = 5

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 the routine is approximately proportional to $nm \times \log n$, but also depends on the factors of n. The routine 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 program reads in sequences of real data values and prints their discrete Fourier transforms (as computed by C06FPF). The Fourier transforms are expanded into full complex form using C06GSF and printed. Inverse transforms are then calculated by calling C06GQF followed by C06FQF showing that the original sequences are restored.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     C06FPF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          MMAX, NMAX
      PARAMETER        (MMAX=5,NMAX=20)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, M, N
*     .. Local Arrays ..
      real             TRIG(2*NMAX), U(NMAX*MMAX), V(NMAX*MMAX),
     +                 WORK(2*MMAX*NMAX), X(NMAX*MMAX)
*     .. External Subroutines ..
      EXTERNAL         C06FPF, C06FQF, C06GQF, C06GSF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'C06FPF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
   20 READ (NIN,*,END=140) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
         DO 40 J = 1, M
            READ (NIN,*) (X(I*M+J),I=0,N-1)
   40    CONTINUE
```

```
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original data values'
         WRITE (NOUT,*)
         DO 60 J = 1, M
            WRITE (NOUT,99999) '      ', (X(I*M+J),I=0,N-1)
   60    CONTINUE
         IFAIL = 0
*
         CALL C06FPF(M,N,X,'Initial',TRIG,WORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     'Discrete Fourier transforms in Hermitian format'
         WRITE (NOUT,*)
         DO 80 J = 1, M
            WRITE (NOUT,99999) '      ', (X(I*M+J),I=0,N-1)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Fourier transforms in full complex form'
*
         CALL C06GSF(M,N,X,U,V,IFAIL)
*
         DO 100 J = 1, M
            WRITE (NOUT,*)
            WRITE (NOUT,99999) 'Real ', (U(I*M+J),I=0,N-1)
            WRITE (NOUT,99999) 'Imag ', (V(I*M+J),I=0,N-1)
  100    CONTINUE
*
         CALL C06GQF(M,N,X,IFAIL)
         CALL C06FQF(M,N,X,'Subsequent',TRIG,WORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original data as restored by inverse transform'
         WRITE (NOUT,*)
         DO 120 J = 1, M
            WRITE (NOUT,99999) '      ', (X(I*M+J),I=0,N-1)
  120    CONTINUE
         GO TO 20
      ELSE
         WRITE (NOUT,*) 'Invalid value of M or N'
      END IF
  140 STOP
*
99999 FORMAT (1X,A,6F10.4)
      END
```

9.2 Program Data

```
C06FPF Example Program Data
     3     6
     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
```

9.3 Program Results

```
C06FPF 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 Fourier transforms in Hermitian format

         1.0737   -0.1041    0.1126   -0.1467   -0.3738   -0.0044
         1.3961   -0.0365    0.0780   -0.1521   -0.0607    0.4666
         1.1237    0.0914    0.3936    0.1530    0.3458   -0.0508

Fourier transforms in full complex form

Real     1.0737   -0.1041    0.1126   -0.1467    0.1126   -0.1041
Imag     0.0000   -0.0044   -0.3738    0.0000    0.3738    0.0044

Real     1.3961   -0.0365    0.0780   -0.1521    0.0780   -0.0365
Imag     0.0000    0.4666   -0.0607    0.0000    0.0607   -0.4666

Real     1.1237    0.0914    0.3936    0.1530    0.3936    0.0914
Imag     0.0000   -0.0508    0.3458    0.0000   -0.3458    0.0508

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
```