NAG Fortran Library Routine Document

C06FQF

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

C06FQF computes the discrete Fourier transforms of m Hermitian sequences, each containing n complex data values. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
SUBROUTINE C06FQF(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 Hermitian sequences of n complex data values z_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:

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

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

The transformed values are purely real (see also the C06 Chapter Introduction).

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

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

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

The routine uses a variant of the fast Fourier transform (FFT) algorithm (Brigham (1974)) known as the Stockham self-sorting algorithm, which is described in Temperton (1983a). Special code is included 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 (1983a) 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 data values in each sequence, n.

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

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 $(1:\mathrm{M}, 0:\mathrm{N}-1)$; each of the m sequences is stored in a **row** of the array in Hermitian form. If the n data values z_j^p are written as $x_j^p + iy_j^p$, then for $0 \leq j \leq n/2$, x_j^p is contained in $\mathrm{X}(p, j)$, and for $1 \leq j \leq (n-1)/2$, y_j^p is contained in $\mathrm{X}(p, n-j)$. (See also Section 2.1.2 of the C06 Chapter Introduction.)

On exit: the components of the m discrete Fourier transforms, stored as if in a two-dimensional array of dimension $(1:\mathrm{M}, 0:\mathrm{N}-1)$. Each of the m transforms is stored as a **row** of the array, overwriting the corresponding original sequence. If the n components of the discrete Fourier transform are denoted by $\hat{x}_k^p$, for $k = 0, 1, \ldots, n-1$, then the mn elements of the array X contain the values

$$\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^m,\ \hat{x}_1^1, \hat{x}_1^2, \ldots,\ \hat{x}_1^m, \ldots,\ \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \ldots, \hat{x}_{n-1}^m.$$

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 compatible 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. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

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, for users 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.**

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, INIT is not one of '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 $= 6$

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 which are assumed to be Hermitian sequences of complex data stored in Hermitian form. The sequences are expanded into full complex form using C06GSF and printed. The discrete Fourier transforms are then computed (using C06FQF) and printed out. Inverse transforms are then calculated by calling C06FPF followed by C06GQF 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.

```
*     C06FQF 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(MMAX*NMAX), V(MMAX*NMAX),
     +                   WORK(2*NMAX*MMAX), X(MMAX*NMAX)
*     .. External Subroutines ..
      EXTERNAL           C06FPF, C06FQF, C06GQF, C06GSF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'C06FQF Example Program Results'
*     Skip heading in data Ûle
      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
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Original data written in full complex form'
         IFAIL = 0
*
         CALL C06GSF(M,N,X,U,V,IFAIL)
*
         DO 80 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)
   80    CONTINUE
*
         CALL C06FQF(M,N,X,'Initial',TRIG,WORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Discrete Fourier transforms (real values)'
         WRITE (NOUT,*)
         DO 100 J = 1, M
            WRITE (NOUT,99999) '     ', (X(I*M+J),I=0,N-1)
  100    CONTINUE
*
         CALL C06FPF(M,N,X,'Subsequent',TRIG,WORK,IFAIL)
         CALL C06GQF(M,N,X,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

```
C06FQF 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

```
 C06FQF 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

Original data written in full complex form

Real     0.3854    0.6772    0.1138    0.6751    0.1138    0.6772
Imag     0.0000    0.1424    0.6362    0.0000   -0.6362   -0.1424

Real     0.5417    0.2983    0.1181    0.7255    0.1181    0.2983
Imag     0.0000    0.8723    0.8638    0.0000   -0.8638   -0.8723

Real     0.9172    0.0644    0.6037    0.6430    0.6037    0.0644
Imag     0.0000    0.4815    0.0428    0.0000   -0.0428   -0.4815

Discrete Fourier transforms (real values)

         1.0788    0.6623   -0.2391   -0.5783    0.4592   -0.4388
         0.8573    1.2261    0.3533   -0.2222    0.3413   -1.2291
         1.1825    0.2625    0.6744    0.5523    0.0540   -0.4790

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
```