NAG 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 (KIND=nag_wp) 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 $j = 0, 1, \ldots, n-1$ and $p = 1, 2, \ldots, m$, C06FQF simultaneously calculates the Fourier transforms of all the sequences defined by

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

(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 forming the complex conjugates of the z_k^p; that is $x(k) = -x(k)$, for $k = (n/2 + 1) \times m + 1, \ldots, m \times n$.

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 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

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

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 ≥ 1.

2: N – INTEGER *Input*

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

Constraint: N ≥ 1.

3: X(M $\times$ N) – 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 $(1 : \mathrm{M}, 0 : \mathrm{N}-1)$; 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 in 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: 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 or C06FQF. 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'.

5: TRIG(2 $\times$ 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').

6: WORK(M $\times$ N) – REAL (KIND=nag_wp) array *Workspace*

7: 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, INIT $\neq$ '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.

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

C06FQF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06FQF 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 C06FQF is approximately proportional to $nm\log(n)$, but also depends on the factors of n. C06FQF 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 which are assumed to be Hermitian sequences of complex data stored in Hermitian form. The sequences are expanded into full complex form and printed. The discrete Fourier transforms are then computed (using C06FQF) and printed out. Inverse transforms are then calculated by conjugating and calling C06FPF showing that the original sequences are restored.

10.1 Program Text

```
    Program c06fqfe

!     C06FQF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c06fpf, c06fqf, 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  :: trig(:), u(:), v(:), work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06FQF 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 (trig(2*n),u(n),v(n),work(2*m*n),x(m*n))
        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,*)
        Write (nout,99999)('     ',(x(i*m+j),i=0,n-1),j=1,m)
        Write (nout,*)
        Write (nout,*) 'Original data written in full complex form'

        Do j = 1, m
          u(1:n) = x(j:m*n:m)
          v(1:n) = 0.0_nag_wp
          v(2:(n+1)/2) = u(n:n-(n-1)/2+1:-1)
          u(n:n-(n-1)/2+1:-1) = u(2:(n+1)/2)
          v(n-(n-1)/2+1:n) = -v((n+1)/2:2:-1)
          Write (nout,*)
          Write (nout,99999) 'Real ', u(1:n)
          Write (nout,99999) 'Imag ', v(1:n)
```

```
        End Do

        ifail = 0
        Call c06fqf(m,n,x,'Initial',trig,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Discrete Fourier transforms (real values)'
        Write (nout,*)
        Write (nout,99999)('     ',(x(i*m+j),i=0,n-1),j=1,m)

        Call c06fpf(m,n,x,'Subsequent',trig,work,ifail)
        x((n/2+1)*m+1:m*n) = -x((n/2+1)*m+1:m*n)

        Write (nout,*)
        Write (nout,*) 'Original data as restored by inverse transform'
        Write (nout,*)
        Write (nout,99999)('     ',(x(i*m+j),i=0,n-1),j=1,m)
        Deallocate (trig,u,v,work,x)
      End Do loop

99999 Format (1X,A,6F10.4)
    End Program c06fqfe
```

10.2 Program Data

```
C06FQF 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

```
 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
```