NAG Library Routine Document

C06EBF

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

C06EBF calculates the discrete Fourier transform of a Hermitian sequence of n complex data values. (No extra workspace required.)

2 Specification

```
SUBROUTINE C06EBF (X, N, IFAIL)

INTEGER             N, IFAIL
REAL (KIND=nag_wp)  X(N)
```

3 Description

Given a Hermitian sequence of n complex data values z_j (i.e., a sequence such that z_0 is real and z_{n-j} is the complex conjugate of z_j, for $j = 1, 2, \ldots, n-1$), C06EBF calculates their discrete Fourier transform defined by

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

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{x}_k$ are purely real (see also the C06 Chapter Introduction).

To compute the inverse discrete Fourier transform defined by

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

this routine should be preceded by a call of C06GBF to form the complex conjugates of the z_j.

C06EBF uses the fast Fourier transform (FFT) algorithm (see Brigham (1974)). There are some restrictions on the value of n (see Section 5).

4 References

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

5 Parameters

1: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the sequence to be transformed stored in Hermitian form. If the data values z_j are written as $x_j + iy_j$, and if X is declared with bounds $(0 : \mathrm{N} - 1)$ in the subroutine from which C06EBF is called, then for $0 \leq j \leq n/2$, x_j is contained in $\mathrm{X}(j)$, and for $1 \leq j \leq (n-1)/2$, y_j is contained in $\mathrm{X}(n-j)$. (See also Section 2.1.2 in the C06 Chapter Introduction and Section 9.)

On exit: the components of the discrete Fourier transform $\hat{x}_k$. If X is declared with bounds $(0 : \mathrm{N} - 1)$ in the subroutine from which C06EBF is called, then $\hat{x}_k$ is stored in $\mathrm{X}(k)$, for $k = 0, 1, \ldots, n-1$.

2: N – INTEGER *Input*

On entry: n, the number of data values. The largest prime factor of N must not exceed 19, and the total number of prime factors of N, counting repetitions, must not exceed 20.

Constraint: $N > 1$.

3: 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:

$IFAIL = 1$

At least one of the prime factors of N is greater than 19.

$IFAIL = 2$

N has more than 20 prime factors.

$IFAIL = 3$

On entry, $N \leq 1$.

$IFAIL = 4$

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 is approximately proportional to $n \times \log n$, but also depends on the factorization of n. C06EBF is faster if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

On the other hand, C06EBF is particularly slow if n has several unpaired prime factors, i.e., if the 'square-free' part of n has several factors. For such values of n, C06FBF (which requires an additional n elements of workspace) is considerably faster.

9 Example

This example reads in a sequence of real data values which is assumed to be a Hermitian sequence of complex data values stored in Hermitian form. The input sequence is expanded into a full complex sequence and printed alongside the original sequence. The discrete Fourier transform (as computed by C06EBF) is printed out. It then performs an inverse transform using C06EAF and C06GBF, and prints the sequence so obtained alongside the original data values.

9.1 Program Text

```
    Program c06ebfe

!     C06EBF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: c06eaf, c06ebf, c06gbf, c06gsf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: ieof, ifail, j, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: u(:), v(:), x(:), xx(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06EBF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) n
        If (ieof<0) Exit loop
        Allocate (u(0:n-1),v(0:n-1),x(0:n-1),xx(0:n-1))
        Read (nin,*) x(0:n-1)
        xx(0:n-1) = x(0:n-1)

!       Convert x to separated real and imaginary parts for printing.
        ifail = 0
        m = 1
        Call c06gsf(m,n,x,u,v,ifail)

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call c06ebf(x,n,ifail)

        Write (nout,*)
        Write (nout,*) 'Original sequence and corresponding complex sequence'
        Write (nout,*)
        Write (nout,*) '         Data           Real      Imag'
        Write (nout,*)
        Write (nout,99999)(j,xx(j),'     ',u(j),v(j),j=0,n-1)
        Write (nout,*)
        Write (nout,*) 'Components of discrete Fourier transform'
        Write (nout,*)
        Write (nout,99998)(j,x(j),j=0,n-1)

        Call c06eaf(x,n,ifail)
        Call c06gbf(x,n,ifail)

        Write (nout,*)
        Write (nout,*) 'Original sequence as restored by inverse transform'
        Write (nout,*)
        Write (nout,*) '        Original  Restored'
        Write (nout,*)
        Write (nout,99997)(j,xx(j),x(j),j=0,n-1)
        Deallocate (u,v,x,xx)
      End Do loop
```

```
99999 Format (1X,I5,F10.5,A,2F10.5)
99998 Format (1X,I5,F10.5)
99997 Format (1X,I5,2F10.5)
    End Program c06ebfe
```

9.2 Program Data

```
C06EBF Example Program Data
    7          : n
  0.34907
  0.54890
  0.74776
  0.94459
  1.13850
  1.32850
  1.51370      : x
```

9.3 Program Results

```
 C06EBF Example Program Results

 Original sequence and corresponding complex sequence

         Data           Real      Imag

     0   0.34907        0.34907   0.00000
     1   0.54890        0.54890   1.51370
     2   0.74776        0.74776   1.32850
     3   0.94459        0.94459   1.13850
     4   1.13850        0.94459  -1.13850
     5   1.32850        0.74776  -1.32850
     6   1.51370        0.54890  -1.51370

 Components of discrete Fourier transform

     0   1.82616
     1   1.86862
     2  -0.01750
     3   0.50200
     4  -0.59873
     5  -0.03144
     6  -2.62557

 Original sequence as restored by inverse transform

        Original  Restored

     0   0.34907   0.34907
     1   0.54890   0.54890
     2   0.74776   0.74776
     3   0.94459   0.94459
     4   1.13850   1.13850
     5   1.32850   1.32850
     6   1.51370   1.51370
```