NAG Library Routine Document

C06PAF

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

C06PAF calculates the discrete Fourier transform of a sequence of n real data values or of a Hermitian sequence of n complex data values stored in compact form in a real array.

2 Specification

```
SUBROUTINE C06PAF (DIRECT, X, N, WORK, IFAIL)
INTEGER            N, IFAIL
REAL (KIND=nag_wp) X(N+2), WORK(*)
CHARACTER(1)       DIRECT
```

3 Description

Given a sequence of n real data values x_j, for $j = 0, 1, \ldots, n-1$, C06PAF calculates their discrete Fourier transform (in the **forward** direction) defined by

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

The transformed values $\hat{z}_k$ are complex, but they form a Hermitian sequence (i.e., $\hat{z}_{n-k}$ is the complex conjugate of $\hat{z}_k$), so they are completely determined by n real numbers (since $\hat{z}_0$ is real, as is $\hat{z}_{n/2}$ for n even).

Alternatively, given a Hermitian sequence of n complex data values z_j, this routine calculates their inverse (**backward**) discrete Fourier transform defined by

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

The transformed values $\hat{x}_k$ are real.

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in the above definitions.)

A call of C06PAF with DIRECT = 'F' followed by a call with DIRECT = 'B' will restore the original data.

C06PAF 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).

The same functionality is available using the forward and backward transform routine pair: C06PVF and C06PWF on setting N = 1. This pair use a different storage solution; real data is stored in a real array, while Hermitian data (the first unconjugated half) is stored in a complex array.

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 Arguments

1: DIRECT – CHARACTER(1) *Input*

On entry: if the forward transform as defined in Section 3 is to be computed, then DIRECT must be set equal to `F'.

If the backward transform is to be computed then DIRECT must be set equal to `B'.

Constraint: DIRECT = 'F' or 'B'.

2: X(N + 2) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if X is declared with bounds $(0 : \mathrm{N} + 1)$ in the subroutine from which C06PAF is called, then:

if DIRECT = 'F', X(j) must contain x_j, for $j = 0, 1, \ldots, n - 1$;

if DIRECT = 'B', X($2 \times k$) and X($2 \times k + 1$) must contain the real and imaginary parts respectively of z_k, for $k = 0, 1, \ldots, n/2$. (Note that for the sequence z_k to be Hermitian, the imaginary part of z_0, and of $z_{n/2}$ for n even, must be zero.)

On exit:

if DIRECT = 'F' and X is declared with bounds $(0 : \mathrm{N} + 1)$, X($2 \times k$) and X($2 \times k + 1$) will contain the real and imaginary parts respectively of $\hat{z}_k$, for $k = 0, 1, \ldots, n/2$;

if DIRECT = 'B' and X is declared with bounds $(0 : \mathrm{N} + 1)$, X(j) will contain $\hat{x}_j$, for $j = 0, 1, \ldots, n - 1$.

3: N – INTEGER *Input*

On entry: n, the number of data values.

Constraint: N ≥ 1.

4: WORK($*$) – REAL (KIND=nag_wp) array *Workspace*

Note: the dimension of the array WORK must be at least $3 \times \mathrm{N} + 100$.

The workspace requirements as documented for C06PAF may be an overestimate in some implementations.

On exit: WORK(1) contains the minimum workspace required for the current value of N with this implementation.

5: 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, N $= \langle value \rangle$.
Constraint: N ≥ 1.

IFAIL $= 2$

$\langle value \rangle$ is an invalid value of DIRECT.

IFAIL $= 3$

An internal error has occurred in this routine. Check the routine call and any array sizes. If the call is correct then please contact NAG for assistance.

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

C06PAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06PAF 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 is approximately proportional to $n \times \log(n)$, but also depends on the factorization of n. C06PAF 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.

10 Example

This example reads in a sequence of real data values and prints their discrete Fourier transform (as computed by C06PAF with DIRECT = 'F'), after expanding it from complex Hermitian form into a full complex sequence. It then performs an inverse transform using C06PAF with DIRECT = 'B', and prints the sequence so obtained alongside the original data values.

10.1 Program Text

```
    Program c06pafe

!     C06PAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c06paf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: ieof, ifail, j, n, nj
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: work(:), x(:), xx(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06PAF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) n
        If (ieof<0) Then
          Exit loop
        End If

        Allocate (work(3*n+100),x(0:n+1),xx(0:n-1))
        Read (nin,*) x(0:n-1)
        xx(0:n-1) = x(0:n-1)

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call c06paf('F',x,n,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Components of discrete Fourier transform'
        Write (nout,*)
        Write (nout,*) '          Real      Imag'
        Write (nout,*)
        Write (nout,99999)(j,x(2*j),x(2*j+1),j=0,n/2)
        Do j = n/2 + 1, n - 1
          nj = n - j
          Write (nout,99999) j, x(2*nj), -x(2*nj+1)
        End Do

        Call c06paf('B',x,n,work,ifail)

        Write (nout,*)
        Write (nout,*) 'Original sequence as restored by inverse transform'
        Write (nout,*)
        Write (nout,*) '        Original  Restored'
        Write (nout,*)
        Write (nout,99999)(j,xx(j),x(j),j=0,n-1)
        Deallocate (work,x,xx)
      End Do loop

99999 Format (1X,I5,2F10.5)
    End Program c06pafe
```

10.2 Program Data

```
C06PAF Example Program Data
    7      : n
  0.34907
  0.54890
  0.74776
  0.94459
  1.13850
  1.32850
  1.51370  : x
```

10.3 Program Results

```
 C06PAF Example Program Results

 Components of discrete Fourier transform

           Real      Imag

     0   2.48361   0.00000
     1  -0.26599   0.53090
     2  -0.25768   0.20298
     3  -0.25636   0.05806
     4  -0.25636  -0.05806
     5  -0.25768  -0.20298
     6  -0.26599  -0.53090

 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
```