NAG Library Routine Document

C06PYF

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

C06PYF computes the three-dimensional discrete Fourier transform of a trivariate sequence of real data values.

2 Specification

```
SUBROUTINE C06PYF (N1, N2, N3, X, Y, IFAIL)
INTEGER                 N1, N2, N3, IFAIL
REAL (KIND=nag_wp)      X(N1*N2*N3)
COMPLEX (KIND=nag_wp) Y((N1/2+1)*N2*N3)
```

3 Description

C06PYF computes the three-dimensional discrete Fourier transform of a trivariate sequence of real data values $x_{j_1 j_2 j_3}$, for $j_1 = 0, 1, \ldots, n_1 - 1$, $j_2 = 0, 1, \ldots, n_2 - 1$ and $j_3 = 0, 1, \ldots, n_3 - 1$.

The discrete Fourier transform is here defined by

$$\hat{z}_{k_1 k_2 k_3} = \frac{1}{\sqrt{n_1 n_2 n_3}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} x_{j_1 j_2 j_3} \times \exp\left(-2\pi i \left(\frac{j_1 k_1}{n_1} + \frac{j_2 k_2}{n_2} + \frac{j_3 k_3}{n_3}\right)\right),$$

where $k_1 = 0, 1, \ldots, n_1 - 1$, $k_2 = 0, 1, \ldots, n_2 - 1$ and $k_3 = 0, 1, \ldots, n_3 - 1$. (Note the scale factor of $\frac{1}{\sqrt{n_1 n_2 n_3}}$ in this definition.)

The transformed values $\hat{z}_{k_1 k_2 k_3}$ are complex. Because of conjugate symmetry (i.e., $\hat{z}_{k_1 k_2 k_3}$ is the complex conjugate of $\hat{z}_{(n_1-k_1) k_2 k_3}$), only slightly more than half of the Fourier coefficients need to be stored in the output.

A call of C06PYF followed by a call of C06PZF will restore the original data.

This routine calls C06PQF and C06PRF to perform multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm in Brigham (1974) and Temperton (1983).

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: N1 – INTEGER *Input*

On entry: n_1, the first dimension of the transform.

Constraint: N1 ≥ 1.

2: N2 – INTEGER *Input*

On entry: n_2, the second dimension of the transform.

Constraint: N2 ≥ 1.

3: N3 – INTEGER *Input*

On entry: n_3, the third dimension of the transform.

Constraint: N3 ≥ 1.

4: X(N1 × N2 × N3) – REAL (KIND=nag_wp) array *Input*

On entry: the real input dataset x, where $x_{j_1 j_2 j_3}$ is stored in X($j_3 \times n_1 n_2 + j_2 \times n_1 + j_1 + 1$), for $j_1 = 0, 1, \ldots, n_1 - 1$, $j_2 = 0, 1, \ldots, n_2 - 1$ and $j_3 = 0, 1, \ldots, n_3 - 1$. That is, if X is regarded as a three-dimensional array of dimension $(0 : \text{N1} - 1, 0 : \text{N2} - 1, 0 : \text{N3} - 1)$, then X($j_1, j_2, j_3$) must contain $x_{j_1 j_2 j_3}$.

5: Y((N1/2 + 1) × N2 × N3) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the complex output dataset $\hat{z}$, where $\hat{z}_{k_1 k_2 k_3}$ is stored in Y($k_3 \times (n_1/2 + 1) n_2 + k_2 \times (n_1/2 + 1) + k_1 + 1$), for $k_1 = 0, 1, \ldots, n_1/2$, $k_2 = 0, 1, \ldots, n_2 - 1$ and $k_3 = 0, 1, \ldots, n_3 - 1$. That is, if Y is regarded as a three-dimensional array of dimension $(0 : \text{N1}/2, 0 : \text{N2} - 1, 0 : \text{N3} - 1)$, then Y($k_1, k_2, k_3$) contains $\hat{z}_{k_1 k_2 k_3}$. Note the first dimension is cut roughly by half to remove the redundant information due to conjugate symmetry.

6: 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, N1 = ⟨*value*⟩.
Constraint: N1 ≥ 1.

IFAIL = 2

On entry, N2 = ⟨*value*⟩.
Constraint: N2 ≥ 1.

IFAIL = 3

On entry, N3 = ⟨*value*⟩.
Constraint: N3 ≥ 1.

IFAIL = 4

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 forward transform using C06PYF and a backward transform using C06PZF, and comparing the results with the original sequence (in exact arithmetic they would be identical).

8 Parallelism and Performance

C06PYF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C06PYF 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 C06PYF is approximately proportional to $n_1 n_2 n_3 \log(n_1 n_2 n_3)$, but also depends on the factors of n_1, n_2 and n_3. C06PYF is fastest if the only prime factors of n_1, n_2 and n_3 are 2, 3 and 5, and is particularly slow if one of the dimensions is a large prime, or has large prime factors.

Workspace is internally allocated by C06PYF. The total size of these arrays is approximately proportional to $n_1 n_2 n_3$.

10 Example

This example reads in a trivariate sequence of real data values and prints their discrete Fourier transforms as computed by C06PYF. Inverse transforms are then calculated by calling C06PZF showing that the original sequences are restored.

10.1 Program Text

```
!   C06PYF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module c06pyfe_mod

!     C06PYF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
```

```
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: readx, writx, writy
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6
    Contains
      Subroutine readx(nin,x,n1,n2,n3)
!       Read 3-dimensional real data

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: n1, n2, n3, nin
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: x(n1,n2,n3)
!       .. Local Scalars ..
        Integer                        :: i, j, k
!       .. Executable Statements ..
        Do i = 1, n1
          Do j = 1, n2
            Read (nin,*)(x(i,j,k),k=1,n3)
          End Do
        End Do
        Return
      End Subroutine readx

      Subroutine writx(nout,x,n1,n2,n3)
!       Print 3-dimensional real data

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: n1, n2, n3, nout
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x(n1,n2,n3)
!       .. Local Scalars ..
        Integer                        :: i, j, k
!       .. Executable Statements ..
        Do i = 1, n1
          Write (nout,*)
          Write (nout,99998) 'x(i,j,k) for i =', i
          Do j = 1, n2
            Write (nout,*)
            Write (nout,99999) 'Real ', (x(i,j,k),k=1,n3)
          End Do
        End Do
        Return

99999   Format (1X,A,4F10.3)
99998   Format (1X,A,I6)
      End Subroutine writx

      Subroutine writy(nout,y,n1,n2,n3)
!       Print 3-dimensional complex data

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: n1, n2, n3, nout
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In) :: y(n1,n2,n3)
!       .. Local Scalars ..
        Integer                        :: i, j, k
!       .. Intrinsic Procedures ..
        Intrinsic                      :: aimag, real
!       .. Executable Statements ..
        Do i = 1, n1
          Write (nout,*)
          Write (nout,99998) 'y(i,j,k) for i =', i
          Do j = 1, n2
            Write (nout,*)
            Write (nout,99999) 'Real ', (real(y(i,j,k)),k=1,n3)
            Write (nout,99999) 'Imag ', (aimag(y(i,j,k)),k=1,n3)
          End Do
        End Do
```

```
        Return

99999   Format (1X,A,4F10.3)
99998   Format (1X,A,I6)
      End Subroutine writy
    End Module c06pyfe_mod

    Program c06pyfe

!     C06PYF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c06pyf, c06pzf, nag_wp
      Use c06pyfe_mod, Only: nin, nout, readx, writx, writy
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                          :: ieof, ifail, n1, n2, n3
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: y(:)
      Real (Kind=nag_wp), Allocatable  :: x(:)
!     .. Executable Statements ..
      Write (nout,*) 'C06PYF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
loop: Do
        Read (nin,*,Iostat=ieof) n1, n2, n3
        If (ieof<0) Then
          Exit loop
        End If

        Allocate (x(n1*n2*n3),y((n1/2+1)*n2*n3))

        Call readx(nin,x,n1,n2,n3)
        Write (nout,*)
        Write (nout,*) 'Original data values'
        Call writx(nout,x,n1,n2,n3)

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
!       -- Compute transform
        Call c06pyf(n1,n2,n3,x,y,ifail)

        Write (nout,*)
        Write (nout,*) 'Components of discrete Fourier transform'
        Call writy(nout,y,n1/2+1,n2,n3)

!       -- Compute inverse transform
        x = 0._nag_wp
        Call c06pzf(n1,n2,n3,y,x,ifail)

        Write (nout,*)
        Write (nout,*) 'Original sequence as restored by inverse transform'
        Call writx(nout,x,n1,n2,n3)
        Deallocate (x,y)

      End Do loop

    End Program c06pyfe
```

10.2 Program Data

```
C06PYF Example Program Data
  3  3  4                                  : n1, n2, n3
      1.541     0.161     1.989     0.037
      0.346     1.907     0.001     1.915
      1.754     0.042     1.991     0.151
      0.584     1.004     1.408     0.252
```

```
     1.284     1.137     0.467     1.834
     0.855     0.725     1.647     0.096
     0.010     1.844     0.452     1.154
     1.960     0.240     1.424     0.987
     0.089     1.660     0.708     0.872 : x
```

10.3 Program Results

```
 C06PYF Example Program Results

 Original data values

 x(i,j,k) for i =     1

 Real      1.541     0.161     1.989     0.037

 Real      0.346     1.907     0.001     1.915

 Real      1.754     0.042     1.991     0.151

 x(i,j,k) for i =     2

 Real      0.584     1.004     1.408     0.252

 Real      1.284     1.137     0.467     1.834

 Real      0.855     0.725     1.647     0.096

 x(i,j,k) for i =     3

 Real      0.010     1.844     0.452     1.154

 Real      1.960     0.240     1.424     0.987

 Real      0.089     1.660     0.708     0.872

 Components of discrete Fourier transform

 y(i,j,k) for i =     1

 Real      5.755    -0.277     0.415    -0.277
 Imag      0.000    -0.237     0.000     0.237

 Real     -0.268     0.109     0.175    -0.688
 Imag     -0.420    -0.756     0.871    -0.210

 Real     -0.268    -0.688     0.175     0.109
 Imag      0.420     0.210    -0.871     0.756

 y(i,j,k) for i =     2

 Real      0.081     0.060     0.645     0.047
 Imag      0.015     0.156    -0.478    -0.077

 Real      0.038    -0.275     1.585     0.201
 Imag      0.198     0.295     0.616     0.061

 Real      0.067     0.280    -0.113    -0.128
 Imag     -0.122     0.012    -1.555    -0.117

 Original sequence as restored by inverse transform

 x(i,j,k) for i =     1

 Real      1.541     0.161     1.989     0.037

 Real      0.346     1.907     0.001     1.915

 Real      1.754     0.042     1.991     0.151
```

```
x(i,j,k) for i =     2
Real      0.584     1.004     1.408     0.252
Real      1.284     1.137     0.467     1.834
Real      0.855     0.725     1.647     0.096
x(i,j,k) for i =     3
Real      0.010     1.844     0.452     1.154
Real      1.960     0.240     1.424     0.987
Real      0.089     1.660     0.708     0.872
```