NAG C Library Function Document

nag_fft_3d (c06pxc)

1 Purpose

nag_fft_3d (c06pxc) computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values (using complex data type).

2 Specification

```
void nag_fft_3d (Nag_TransformDirection direct, Integer n1, Integer n2, Integer n3,
     Complex x[], NagError *fail)
```

3 Description

nag_fft_3d (c06pxc) computes the three-dimensional discrete Fourier transform of a trivariate sequence of complex data values $z_{j_1 j_2 j_3}$, where $j_1 = 0, 1, \ldots, n_1 - 1$, $j_2 = 0, 1, \ldots, n_2 - 1$ and $j_3 = 0, 1, \ldots, n_3 - 1$.

The discrete Fourier transform is here defined by

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

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 minus sign is taken in the argument of the exponential within the summation when the forward transform is required, and the plus sign is taken when the backward transform is required. A call of the function with **direct** = **Nag_ForwardTransform** followed by a call with **direct** = **Nag_BackwardTransform** will restore the original data.

This function performs multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm (Brigham (1974)).

4 References

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

Temperton C (1983b) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

5 Parameters

1: **direct** – Nag_TransformDirection *Input*

On entry: if the **F**orward transform as defined in Section 3 is to be computed, then **direct** must be set equal to **Nag_ForwardTransform**. If the **B**ackward transform is to be computed then **direct** must be set equal to **Nag_BackwardTransform**.

Constraint: **direct** = **Nag_ForwardTransform** or **Nag_BackwardTransform**.

2: **n1** – Integer *Input*

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

Constraint: **n1** ≥ 1.

3: **n2** – Integer *Input*

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

Constraint: **n2** ≥ 1.

4: **n3** – Integer *Input*

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

Constraint: **n3** ≥ 1.

5: **x**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **x** must be at least **n1** $\times$ **n2** $\times$ **n3**.

On entry: the complex data values. Data values are stored in **x** using column-major ordering for storing multi-dimensional arrays; that is, $z_{j_1 j_2 j_3}$ is stored in **x**[$j_1 + n_1 j_2 + n_1 n_2 j_3$].

On exit: the corresponding elements of the computed transform.

6: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n1** = ⟨*value*⟩.
Constraint: **n1** ≥ 1.

On entry, **n2** = ⟨*value*⟩.
Constraint: **n2** ≥ 1.

On entry, **n3** = ⟨*value*⟩.
Constraint: **n3** ≥ 1.

n3 must have less than 31 prime factors: **n3** = ⟨*value*⟩.

n2 must have less than 31 prime factors: **n2** = ⟨*value*⟩.

n1 must have less than 31 prime factors: **n1** = ⟨*value*⟩.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

NE_INTERNAL_ERROR

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

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 $n_1 n_2 n_3 \times \log(n_1 n_2 n_3)$, but also depends on the factorization of the individual dimensions n_1, n_2 and n_3. The function is somewhat faster than average if their only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2.

9 Example

This program reads in a trivariate sequence of complex data values and prints the three-dimensional Fourier transform. It then performs an inverse transform and prints the sequence so obtained, which may be compared to the original data values.

9.1 Program Text

```
/* nag_ftt_3d(c06pxc) Example Program
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagc06.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer  i, j, k, n1, n2, n3;
  Integer  exit_status=0;
  NagError fail;
  /* Arrays */
  Complex *x=0;
#define X(I,J,K) x[(K-1)*n2*n1 + (J-1)*n1 + I - 1]

  INIT_FAIL(fail);
  Vprintf("c06pxc Example Program Results\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  Vscanf("%ld%ld%ld", &n1, &n2, &n3);
  Vscanf("%*[^\n]");
  if (n1*n2*n3>=1)
    {
      /* Allocate memory */
      if ( !(x = NAG_ALLOC(n1 * n2 * n3, Complex)) )
        {
          Vprintf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      /* Read in complex data and print out. */
      for (i = 1; i <= n1; ++i)
        {
          for (j = 1; j <= n2; ++j)
            {
              for (k = 1; k <= n3; ++k)
                {
                  Vscanf(" ( %lf, %lf ) ", &X(i,j,k).re, &X(i,j,k).im);
                }
            }
        }
      Vscanf("%*[^\n]");
      Vprintf("\nOriginal data values\n\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, x, n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 1\n", Nag_NoLabels, 0,
             Nag_NoLabels, 0, 90, 0, 0, &fail);
      Vprintf("\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, &X(2,1,1), n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 2\n", Nag_NoLabels, 0, Nag_NoLabels,
             0, 90, 0, 0, &fail);
```

```
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from x04dbc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      /* Compute transform */
      c06pxc(Nag_ForwardTransform, n1, n2, n3, x, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from c06pxc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      Vprintf("\nComponents of discrete Fourier transforms\n\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, x, n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 1\n", Nag_NoLabels, 0, Nag_NoLabels,
             0, 90, 0, 0, &fail);
      Vprintf("\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, &X(2,1,1), n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 2\n", Nag_NoLabels, 0, Nag_NoLabels,
             0, 90, 0, 0, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from x04dbc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      /* Compute inverse transform */
      c06pxc(Nag_BackwardTransform, n1, n2, n3, x, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from c06pxc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }
      Vprintf("\nOriginal data as restored by inverse transform\n\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, x, n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 1\n", Nag_NoLabels, 0, Nag_NoLabels,
             0, 90, 0, 0, &fail);
      Vprintf("\n");
      x04dbc(Nag_ColMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n2,
             n3, &X(2,1,1), n1*n2, Nag_BracketForm, "%6.3f",
             "X(i,j,k) for i = 2\n", Nag_NoLabels, 0, Nag_NoLabels,
             0, 90, 0, 0, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from x04dbc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }
    }
  else
    Vfprintf(stderr,"\nInvalid value of n1, n2 or n3.\n");

 END:
  if (x) NAG_FREE(x);

  return exit_status;
}
```

9.2 Program Data

```
c06pxc Example Program Data
2 3 4  : values of n1, n2, n3
    ( 1.000, 0.000) ( 0.999,-0.040) ( 0.987,-0.159) ( 0.936,-0.352)
    ( 0.994,-0.111) ( 0.989,-0.151) ( 0.963,-0.268) ( 0.891,-0.454)
    ( 0.903,-0.430) ( 0.885,-0.466) ( 0.823,-0.568) ( 0.694,-0.720)
    ( 0.500, 0.500) ( 0.499, 0.040) ( 0.487, 0.159) ( 0.436, 0.352)
    ( 0.494, 0.111) ( 0.489, 0.151) ( 0.463, 0.268) ( 0.391, 0.454)
    ( 0.403, 0.430) ( 0.385, 0.466) ( 0.323, 0.568) ( 0.194, 0.720)
```

9.3 Program Results

```
c06pxc Example Program Results

Original data values

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

   ( 1.000, 0.000)  ( 0.999,-0.040)  ( 0.987,-0.159)  ( 0.936,-0.352)
   ( 0.500, 0.500)  ( 0.499, 0.040)  ( 0.487, 0.159)  ( 0.436, 0.352)
   ( 0.994,-0.111)  ( 0.989,-0.151)  ( 0.963,-0.268)  ( 0.891,-0.454)

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

   ( 0.500, 0.500)  ( 0.499, 0.040)  ( 0.487, 0.159)  ( 0.436, 0.352)
   ( 0.994,-0.111)  ( 0.989,-0.151)  ( 0.963,-0.268)  ( 0.891,-0.454)
   ( 0.494, 0.111)  ( 0.489, 0.151)  ( 0.463, 0.268)  ( 0.391, 0.454)

Components of discrete Fourier transforms

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

   ( 3.292, 0.102)  ( 0.051,-0.042)  ( 0.113, 0.102)  ( 0.051, 0.246)
   ( 1.225,-1.620)  ( 0.355, 0.083)  ( 0.000, 0.162)  (-0.355, 0.083)
   ( 0.143,-0.086)  ( 0.016, 0.153)  (-0.024, 0.127)  (-0.050, 0.086)

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

   ( 1.225,-1.620)  ( 0.355, 0.083)  ( 0.000, 0.162)  (-0.355, 0.083)
   ( 0.143,-0.086)  ( 0.016, 0.153)  (-0.024, 0.127)  (-0.050, 0.086)
   ( 0.424, 0.320)  ( 0.020,-0.115)  ( 0.013,-0.091)  (-0.007,-0.080)

Original data as restored by inverse transform

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

   ( 1.000,-0.000)  ( 0.999,-0.040)  ( 0.987,-0.159)  ( 0.936,-0.352)
   ( 0.500, 0.500)  ( 0.499, 0.040)  ( 0.487, 0.159)  ( 0.436, 0.352)
   ( 0.994,-0.111)  ( 0.989,-0.151)  ( 0.963,-0.268)  ( 0.891,-0.454)

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

   ( 0.500, 0.500)  ( 0.499, 0.040)  ( 0.487, 0.159)  ( 0.436, 0.352)
   ( 0.994,-0.111)  ( 0.989,-0.151)  ( 0.963,-0.268)  ( 0.891,-0.454)
   ( 0.494, 0.111)  ( 0.489, 0.151)  ( 0.463, 0.268)  ( 0.391, 0.454)
```