NAG Library Function Document

nag_fft_multid_full (c06pjc)

1 Purpose
nag_fft_multid_full (c06pjc) computes the multidimensional discrete Fourier transform of a multivariate sequence of complex data values.

2 Specification
#include <nag.h>
#include <nagc06.h>
void nag_fft_multid_full (Nag_TransformDirection direct, Integer ndim,
const Integer nd[], Integer n, Complex x[], NagError *fail)

3 Description
nag_fft_multid_full (c06pjc) computes the multidimensional discrete Fourier transform of a multivariate sequence of complex data values $z_{j_1j_2...j_m}$, where $j_1 = 0, 1, ..., n_1 - 1$, $j_2 = 0, 1, ..., n_2 - 1$, and so on. Thus the individual dimensions are $n_1, n_2, ..., n_m$, and the total number of data values is $n = n_1 \times n_2 \times \cdots \times n_m$.

The discrete Fourier transform is here defined (e.g., for $m = 2$) by:

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

where $k_1 = 0, 1, ..., n_1 - 1$ and $k_2 = 0, 1, ..., n_2 - 1$. The plus or minus sign in the argument of the exponential terms in the above definition determine the direction of the transform: a minus sign defines the **forward** direction and a plus sign defines the **backward** direction.

The extension to higher dimensions is obvious. (Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.)

A call of nag_fft_multid_full (c06pjc) with `direct` = Nag_ForwardTransform followed by a call with `direct` = Nag_BackwardTransform will restore the original data.

The data values must be supplied in a one-dimensional array using column-major storage ordering of multidimensional data (i.e., with the first subscript $j_1$ varying most rapidly).

This function 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).

4 References

5 Arguments

t1:  **direct** – Nag_TransformDirection

    *Input*

    **On entry:** if the forward transform as defined in Section 3 is to be computed, then `direct` must be set equal to Nag_ForwardTransform.
If the backward transform is to be computed then **direct** must be set equal to Nag_BackwardTransform.

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

2: **ndim** – Integer  
**Input**  
*On entry*: \(m\), the number of dimensions (or variables) in the multivariate data.

*Constraint*: \(ndim \geq 1\).

3: **nd[ndim]** – const Integer  
**Input**  
*On entry*: the elements of **nd** must contain the dimensions of the **ndim** variables; that is, \(nd[i - 1]\) must contain the dimension of the \(i\)th variable.

*Constraint*: \(nd[i - 1] \geq 1\), for \(i = 1, 2, \ldots, ndim\).

4: **n** – Integer  
**Input**  
*On entry*: \(n\), the total number of data values.

*Constraint*: \(n\) must equal the product of the first **ndim** elements of the array **nd**.

5: **x[n]** – Complex  
**Input/Output**  
*On entry*: the complex data values. Data values are stored in **x** using column-major ordering for storing multidimensional arrays; that is, \(z_{j_1 j_2 \ldots j_m}\) is stored in \(x[j_1 + n_1 j_2 + n_1 n_2 j_3 + \ldots]\).

*On exit*: the corresponding elements of the computed transform.

6: **fail** – NagError *  
**Input/Output**  
The NAG error argument (see Section 3.6 in the Essential Introduction).

6  **Error Indicators and Warnings**

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \(<\text{value}\>\) had an illegal value.

**NE_INT**

On entry, \(ndim = \langle\text{value}\rangle\).

*Constraint*: \(ndim \geq 1\).

**NE_INT_2**

\(n\) must equal the product of the dimensions held in array **nd**: \(n = \langle\text{value}\rangle\), product of **nd** elements is \(<\text{value}\>\).

*On entry*: \(nd[I - 1] = \langle\text{value}\rangle\) and \(I = \langle\text{value}\rangle\).

*Constraint*: \(nd[I - 1] \geq 1\).

**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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 in the Essential Introduction 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

nag_fft_multid_full (c06pjc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_fft_multid_full (c06pjc) 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 function. 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 the individual dimensions $nd[i - 1]$. nag_fft_multid_full (c06pjc) is faster if the only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2.

10 Example

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

10.1 Program Text

/* nag_fft_multid_full (c06pjc) Example Program. *
 * Copyright 2014 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, n, ndim;
    Integer exit_status = 0;
    NagError fail;
    /* Arrays */
    Complex *x = 0;
    Integer *nd = 0;
    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */
INIT_FAIL(fail);

printf("nag_fft_multid_full (c06pjc) Example Program Results\n");
/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n"]);
#else
    scanf("%*[\n"]);
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"", &ndim, &n);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"", &ndim, &n);
#endif
if (n >= 1)
{
    /* Allocate memory */
    if (!(x = NAG_ALLOC(n, Complex)) ||
        !(nd = NAG_ALLOC(ndim, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 0; i < ndim; ++i)
    {
        #ifdef _WIN32
            scanf_s("%"NAG_IFMT"", &nd[i]);
        #else
            scanf("%"NAG_IFMT"", &nd[i]);
        #endif
    }
    /* Read in complex data and print out. */
    #ifdef _WIN32
        scanf_s("%*[\n"]);
    #else
        scanf("%*[\n"]);
    #endif
    for (i = 0; i < n; ++i)
    {
        #ifdef _WIN32
            scanf_s(" ( %lf, %lf ) ", &x[i].re, &x[i].im);
        #else
            scanf(" ( %lf, %lf ) ", &x[i].re, &x[i].im);
        #endif
    }
    #ifdef _WIN32
        scanf_s("%*[\n"]);
    #else
        scanf("%*[\n"]);
    #endif
    printf("\n");
    /* nag_gen_complx_mat_print_comp (x04dbc).
     * Print complex general matrix (comprehensive)
     */
    fflush(stdout);
    nag_gen_complx_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
        Nag_NonUnitDiag, nd[0], n/nd[0], x, nd[0],
        Nag_BracketForm, "%6.3f",
        "Original data values", Nag_NoLabels, 0,
        Nag_NoLabels, 0, 80, 0, 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
}
/* Compute transform */
/* nag_fft_multid_full (c06pjc). */
* Multi-dimensional complex discrete Fourier transform of
* multi-dimensional data (using complex data type)
*/

nag_fft_multid_full(Nag_ForwardTransform, ndim, nd, n, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multid_full (c06pjc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
/* nag_gen_complx_mat_print_comp (x04dbc), see above. */
fflush(stdout);
nag_gen_complx_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
    Nag_NonUnitDiag, nd[0], n/nd[0], x, nd[0],
    NagBracketForm, "%6.3f",
    "Components of discrete Fourier transform",
    Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute inverse transform */
/* nag_fft_multid_full (c06pjc), see above. */
nag_fft_multid_full(Nag_BackwardTransform, ndim, nd, n, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multid_full (c06pjc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
/* nag_gen_complx_mat_print_comp (x04dbc), see above. */
fflush(stdout);
nag_gen_complx_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
    Nag_NonUnitDiag, nd[0], n/nd[0], x, nd[0],
    NagBracketForm, "%6.3f",
    "Original data as restored by inverse "
    "transform", Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
else
{
    printf("\nInvalid value of n.\n");
}
END:
NAG_FREE(x);
NAG_FREE(nd);
return exit_status;
}
10.2 Program Data

nag_fft_multid_full (c06pjc) Example Program Data

2 15
3 5
(1.000,0.000) (0.994,-0.111) (0.903,-0.430)
(0.999,-0.040) (0.989,-0.151) (0.885,-0.466)
(0.987,-0.159) (0.963,-0.268) (0.823,-0.568)
(0.936,-0.352) (0.891,-0.454) (0.694,-0.720)
(0.802,-0.597) (0.731,-0.682) (0.467,-0.884)

10.3 Program Results

nag_fft_multid_full (c06pjc) Example Program Results

Original data values
( 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)
( 0.802,-0.597)
( 0.731,-0.682)
( 0.467,-0.884)

Components of discrete Fourier transform
( 3.373,-1.519) ( 0.481,-0.091) ( 0.251, 0.178) ( 0.054, 0.319)
( 0.457, 0.137) ( 0.055, 0.032) ( 0.009, 0.039) (-0.022, 0.036)
(-0.170, 0.493) (-0.037, 0.058) (-0.042, 0.008) (-0.038, 0.025)
(-0.419, 0.415)
(-0.076, 0.004)
(-0.002,-0.083)

Original data as restored by inverse transform
( 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)
( 0.802,-0.597)
( 0.731,-0.682)
( 0.467,-0.884)