1 Purpose

nag_fft_multid_single (c06pfc) computes the discrete Fourier transform of one variable in a multivariate sequence of complex data values.

2 Specification

```c
#include <nag.h>
#include <nagc06.h>

void nag_fft_multid_single (Nag_TransformDirection direct, Integer ndim,
                          Integer l, const Integer nd[], Integer n, Complex x[], NagError *fail)
```

3 Description

nag_fft_multid_single (c06pfc) computes the discrete Fourier transform of one variable (the \( l \)th say) in a multivariate sequence of complex data values \( z_{j_1, j_2, \ldots, j_m} \), where \( j_1 = 0, 1, \ldots, n_1 - 1, \quad j_2 = 0, 1, \ldots, n_2 - 1, \) and so on. Thus the individual dimensions are \( n_1, n_2, \ldots, n_m \), and the total number of data values is \( n = n_1 \times n_2 \times \cdots \times n_m \).

The function computes \( n/n_l \) one-dimensional transforms defined by

\[
\hat{z}_{j_1, k_2, \ldots, j_m} = \frac{1}{\sqrt{n_l}} \sum_{j_1=0}^{n_1-1} z_{j_1, j_2, \ldots, j_m} \exp\left( \pm \frac{2\pi ij_1 k_l}{n_l} \right),
\]

where \( k_l = 0, 1, \ldots, n_l - 1 \). 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.

(Note the scale factor of \( \frac{1}{\sqrt{n_l}} \) in this definition.)

A call of nag_fft_multid_single (c06pfc) with \texttt{direct} = Nag_FowardTransform followed by a call with \texttt{direct} = Nag_BackwardTransform will restore the original data.

The data values must be supplied in a one-dimensional complex 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

1: \texttt{direct} – Nag_TransformDirection  
   \textit{Input}

   \textit{On entry:} if the forward transform as defined in Section 3 is to be computed, then \texttt{direct} must be set equal to Nag_FowardTransform.
If the backward transform is to be computed then direct must be set equal to Nag_BackwardTransform.

Constraint: direct = Nag_ForwardTransform or Nag_BackwardTransform.

2: ndim – Integer  
   Input

On entry: m, the number of dimensions (or variables) in the multivariate data.

Constraint: ndim ≥ 1.

3: l – Integer  
   Input

On entry: l, the index of the variable (or dimension) on which the discrete Fourier transform is to be performed.

Constraint: 1 ≤ l ≤ ndim.

4: 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] ≥ 1, for i = 1, 2, ..., ndim.

5: 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.

6: 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_1j_2...j_m} is stored in x[j_1 + n_1j_2 + n_1n_2j_3 + ...].

On exit: the corresponding elements of the computed transform.

7: 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 <value> had an illegal value.

NE_INT  
On entry, l = <value>.  
Constraint: 1 ≥ l and l ≤ ndim.

On entry, ndim = <value>.  
Constraint: ndim ≥ 1.

NE_INT_2  
n must equal the product of the dimensions held in array nd: n = <value>, product of nd elements is <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.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 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_single (c06pfc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_fft_multid_single (c06pfc) 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 \( n \times \log n_t \), but also depends on the factorization of \( n_t \).
\( \text{nag_fft_multid_single (c06pfc) is faster if the only prime factors of } n_t \text{ are } 2, 3 \text{ or } 5; \text{ and fastest of all if } n_t \text{ is a power of } 2. \)

10 Example
This example reads in a bivariate sequence of complex data values and prints the discrete Fourier transform of the second variable. It then performs an inverse transform and prints the sequence so obtained, which may be compared with the original data values.

10.1 Program Text
/* nag_fft_multid_single (c06pfc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* * Mark 7, 2002.
*/
#include <nag.h>
#include <stdio.h>
#include <string.h>
#include <nag_stdlib.h>
#include <nagc06.h>
#include <nagx04.h>
int main(void)

{ /* Scalars */
  Integer i, l, 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_single (c06pfc) Example Program Results\n");
#ifdef _WIN32
  scanf_s("%*\n");
#else
  scanf("%*\n");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"", &ndim, &l, &n);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"", &ndim, &l, &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");
  fflush(stdout);
  /* 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",
    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_single (c06pfc).
* One-dimensional complex discrete Fourier transform of
* multi-dimensional data (using complex data type)
*/
nag_fft_multid_single(Nag_ForwardTransform, ndim, l, nd, n, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multid_single (c06pfc).\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],
    Nag_BracketForm, "%6.3f", "Discrete Fourier transform of variable 2",
    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_single (c06pfc), see above. */
nag_fft_multid_single(Nag_BackwardTransform, ndim, l, nd, n, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multid_single (c06pfc).\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],
    Nag_BracketForm, "%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_single (c06pfc) Example Program Data

2 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_single (c06pfc) Example Program Results

Original data
(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)

Discrete Fourier transform of variable 2
(2.113, -0.513) (0.288, 0.000) (0.126, 0.130) (-0.003, 0.190)
(2.043, -0.745) (0.286, 0.032) (0.139, 0.115) (0.018, 0.189)
(1.687, -1.372) (0.260, -0.125) (0.170, 0.063) (0.079, 0.173)
(-0.287, 0.194)
(-0.263, 0.225)
(-0.176, 0.299)

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)
(0.936, -0.430) (0.891, -0.466) (0.823, -0.568) (0.694, -0.720)
(0.802, -0.597)
(0.731, -0.682)
(0.467, -0.884)