1 Purpose

nag_sum_fft_real_3d (c06pyc) computes the three-dimensional discrete Fourier transform of a trivariate sequence of real data values.

2 Specification

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

void nag_sum_fft_real_3d (Integer n1, Integer n2, Integer n3, const double x[], Complex y[], NagError *fail)
```

3 Description

nag_sum_fft_real_3d (c06pyc) 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

\[
\hat{z}_{k_1,k_2,k_3} = \frac{1}{\sqrt{n_1n_2n_3}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} x_{j_1,j_2,j_3} \times \exp\left(-2\pi i \left( \frac{j_1k_1}{n_1} + \frac{j_2k_2}{n_2} + \frac{j_3k_3}{n_3} \right) \right),
\]

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 \( \frac{1}{\sqrt{n_1n_2n_3}} \) 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 \( \hat{z}_{(n_1-k_1),(n_2-k_2),(n_3-k_3)} \)), only slightly more than half of the Fourier coefficients need to be stored in the output.

A call of nag_sum_fft_real_3d (c06pyc) followed by a call of nag_sum_fft_hermitian_3d (c06pzc) will restore the original data.

This function performs multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm in Brigham (1974) and Temperton (1983).

4 References


5 Arguments

1: \( \textbf{n1} \) – Integer \hspace{1cm} Input

\textit{On entry:} \( n_1 \), the first dimension of the transform.

\textit{Constraint:} \( \textbf{n1} \geq 1 \).

2: \( \textbf{n2} \) – Integer \hspace{1cm} Input

\textit{On entry:} \( n_2 \), the second dimension of the transform.

\textit{Constraint:} \( \textbf{n2} \geq 1 \).
3: \( n3 \) – Integer

\textit{Input}

\textit{On entry:} \( n3 \), the third dimension of the transform.

\textit{Constraint:} \( n3 \geq 1 \).

4: \( x[n1 \times n2 \times n3] \) – const double

\textit{Input}

\textit{On entry:} the real input dataset \( x \), where \( x_{j_1,j_2,j_3} \) is stored in \( x[j_3 \times n1n2 + j_2 \times n1 + j1] \), for \( j_1 = 0,1,\ldots,n_1 - 1 \), \( j_2 = 0,1,\ldots,n_2 - 1 \) and \( j_3 = 0,1,\ldots,n_3 - 1 \).

5: \( y[dim] \) – Complex

\textit{Output}

\textit{Note:} the dimension, \( dim \), of the array \( y \) must be at least \( (n1/2 + 1) \times n2 \times n3 \).

\textit{On exit:} the complex output dataset \( \tilde{z} \), where \( \tilde{z}_{k_1,k_2,k_3} \) is stored in \( y[k_3 \times (n1/2 + 1)n2 + k_2 \times (n1/2 + 1) + k1] \), for \( k_1 = 0,1,\ldots,n1/2 \), \( k_2 = 0,1,\ldots,n2 - 1 \) and \( k_3 = 0,1,\ldots,n3 - 1 \). Note the first dimension is cut roughly by half to remove the redundant information due to conjugate symmetry.

6: \( \text{fail} \) – NagError *

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \hspace{1em} \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE_INT}

On entry, \( n1 = \langle \text{value} \rangle \).

\textbf{Constraint:} \( n1 \geq 1 \).

On entry, \( n2 = \langle \text{value} \rangle \).

\textbf{Constraint:} \( n2 \geq 1 \).

On entry, \( n3 = \langle \text{value} \rangle \).

\textbf{Constraint:} \( n3 \geq 1 \).

\textbf{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.

\textbf{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 \hspace{1em} \textbf{Accuracy}

Some indication of accuracy can be obtained by performing a forward transform using \texttt{nag_sum_fft_real_3d (c06pyc)} and a backward transform using \texttt{nag_sum_fft_hermitian_3d (c06pzc)}, and comparing the results with the original sequence (in exact arithmetic they would be identical).
8 Parallelism and Performance

`nag_sum_fft_real_3d (c06pyc)` is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_sum_fft_real_3d (c06pyc)` 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 by `nag_sum_fft_real_3d (c06pyc)` 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 $n_3$. `nag_sum_fft_real_3d (c06pyc)` 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 `nag_sum_fft_real_3d (c06pyc)`. 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 `nag_sum_fft_real_3d (c06pyc)`. Inverse transforms are then calculated by calling `nag_sum_fft_hermitian_3d (c06pzc)` showing that the original sequences are restored.

10.1 Program Text

```c
/* `nag_sum_fft_real_3d (c06pyc)` Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 24, 2013. */

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

int main(void)
{
    /* Scalars */
    Integer exit_status = 0, k, n1, n2, n3;
    /* Arrays */
    Complex *y = 0;
    double *x = 0;
    char title[30];
    /* Nag Types */
    NagError fail;
    INIT_FAIL(fail);

    printf("nag_sum_fft_real_3d (c06pyc) Example Program Results\n");
    fflush(stdout);

    /* Read dimensions of array from data file. */
    #ifdef _WIN32
    scanf_s("%*[\n] %"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n1, &n2, &n3);
    #else
    scanf("%*[\n] %"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n1, &n2, &n3);
    #endif
```
if (!(x = NAG_ALLOC(n1*n2*n3, double)) ||
!(y = NAG_ALLOC((n1/2+1)*n2*n3, Complex)))
{
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}

/* Read array values from data file and print out. */
for (k = 0; k < n1*n2*n3; k++)
#ifdef _WIN32
  scanf_s("%lf", &x[k]);
#else
  scanf("%lf", &x[k]);
#endif

printf("Below we define X(i,j,k)=x[k*n1*n2+j*n1+i] where i and j are the row and column 
indices of the matrices printed.
Y is defined similarly (but having n1/2+1 rows
only due to conjugate symmetry).

Original data values\n");
fflush(stdout);
for (k = 0; k < n3; k++)
#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n X(i,j,k) for k = %" NAG_IFMT, k);
#else
  sprintf(title, "\n X(i,j,k) for k = %" NAG_IFMT, k);
#endif
  nag_gen_real_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
                           Nag_NonUnitDiag, n1, n2, &x[k*n1*n2], n1,
                           "%6.3f", title, Nag_NoLabels, 0,
                           Nag_NoLabels, 0, 80, 0, 0, &fail);

if (fail.code != NE_NOERROR)
{
  printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

Compute three-dimensional real-to-complex discrete Fourier transform using 
* nag_sum_fft_real_3d (c06pyc) and print out.
*/
nag_sum_fft_real_3d(n1, n2, n3, x, y, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_sum_fft_real_3d (c06pyc).\n%s\n", fail.message);
  exit_status = 2;
  goto END;
}

Components of discrete Fourier transform\n");
fflush(stdout);
for (k = 0; k < n3; k++)
#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n Y(i,j,k) for k = %" NAG_IFMT, k);
#else
  sprintf(title, "\n Y(i,j,k) for k = %" NAG_IFMT, k);
#endif
  /* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive) */
nag_gen_complx_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
                             Nag_NonUnitDiag, n1/2+1, n2,
                             &y[k*(n1/2+1)*n2], n1/2+1,
                             Nag_BracketForm, "%6.3f", title,
                             Nag_NoLabels, 0, Nag_NoLabels, 0, 90, 0, 0,
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complex_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 3;
    goto END;
}

/* Compute three-dimensional complex-to-real discrete Fourier transform using
 * nag_sum_fft_hermitian_3d (c06pzc) and print out. */
nag_sum_fft_hermitian_3d(n1, n2, n3, y, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sum_fft_hermitian_3d (c06pzc).\n%s\n", fail.message);
    exit_status = 4;
    goto END;
}

printf("\n Original sequence as restored by inverse transform\n");
fflush(stdout);
for (k = 0; k < n3; k++)
{
#ifdef _WIN32
    sprintf_s(title, _countof(title), "\n X(i,j,k) for k = %" NAG_IFMT, k);
#else
    sprintf(title, "\n X(i,j,k) for k = %" NAG_IFMT, k);
#endif
    /* nag_gen_real_mat_print_comp (x04cbc).
     * Print out a real matrix (comprehensive) */
nag_gen_real_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
    Nag_NonUnitDiag, n1, n2, &x[k*n1*n2], n1,
    "%6.3f", title, Nag_NoLabels, 0,
    Nag_NoLabels, 0, 80, 0, 0, &fail);
}
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
}

END:
NAG_FREE(x);
NAG_FREE(y);
return exit_status;

10.2 Program Data

nag_sum_fft_real_3d (c06pyc) Example Program Data

<table>
<thead>
<tr>
<th>3</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.541</td>
<td>0.584</td>
<td>0.010</td>
</tr>
<tr>
<td>0.346</td>
<td>1.284</td>
<td>1.960</td>
</tr>
<tr>
<td>1.754</td>
<td>0.855</td>
<td>0.089</td>
</tr>
<tr>
<td>0.161</td>
<td>1.004</td>
<td>1.844</td>
</tr>
<tr>
<td>1.907</td>
<td>1.137</td>
<td>0.240</td>
</tr>
<tr>
<td>0.042</td>
<td>0.725</td>
<td>1.660</td>
</tr>
<tr>
<td>1.989</td>
<td>1.408</td>
<td>0.452</td>
</tr>
<tr>
<td>0.001</td>
<td>0.467</td>
<td>1.424</td>
</tr>
</tbody>
</table>
10.3 Program Results

nag_sum_fft_real_3d (c06pyc) Example Program Results

Below we define $X(i,j,k)=x[k*n1*n2+j*n1+i]$ where $i$ and $j$ are the row and column indices of the matrices printed. $Y$ is defined similarly (but having $n1/2+1$ rows only due to conjugate symmetry).

Original data values

$X(i,j,k)$ for $k = 0$
1.541 0.346 1.754  
0.584 1.284 0.855  
0.010 1.960 0.089

$X(i,j,k)$ for $k = 1$
0.161 1.907 0.042  
1.004 1.137 0.725  
1.844 0.240 1.660

$X(i,j,k)$ for $k = 2$
1.989 0.001 1.991  
1.408 0.467 1.647  
0.452 1.424 0.708

$X(i,j,k)$ for $k = 3$
0.037 1.915 0.151  
0.252 1.834 0.096  
1.154 0.987 0.872

Components of discrete Fourier transform

$Y(i,j,k)$ for $k = 0$
(5.755, 0.000) (-0.268,-0.420) (-0.268, 0.420) 
(0.081, 0.015) ( 0.038, 0.198) ( 0.067,-0.122)

$Y(i,j,k)$ for $k = 1$
(-0.277,-0.237) ( 0.109,-0.756) (-0.688, 0.210) 
( 0.060, 0.156) (-0.275, 0.295) ( 0.280, 0.012)

$Y(i,j,k)$ for $k = 2$
( 0.415, 0.000) ( 0.175, 0.871) ( 0.175,-0.871) 
( 0.645,-0.478) (1.585, 0.616) (-0.113,-1.555)

$Y(i,j,k)$ for $k = 3$
(-0.277, 0.237) (-0.688,-0.210) ( 0.109, 0.756) 
( 0.047,-0.077) ( 0.201, 0.061) (-0.128,-0.117)

Original sequence as restored by inverse transform

$X(i,j,k)$ for $k = 0$
1.541 0.346 1.754  
0.584 1.284 0.855  
0.010 1.960 0.089

$X(i,j,k)$ for $k = 1$
0.161 1.907 0.042  
1.004 1.137 0.725  
1.844 0.240 1.660

$X(i,j,k)$ for $k = 2$
1.989 0.001 1.991  
1.408 0.467 1.647  
0.452 1.424 0.708
\(X(i,j,k)\) for \(k = 3\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.037</td>
<td>1.915</td>
<td>0.151</td>
</tr>
<tr>
<td>0.252</td>
<td>1.834</td>
<td>0.096</td>
</tr>
<tr>
<td>1.154</td>
<td>0.987</td>
<td>0.872</td>
</tr>
</tbody>
</table>