NAG Library Function Document

nag_sum_fft_real_2d (c06pvc)

1 Purpose
nag_sum_fft_real_2d (c06pvc) computes the two-dimensional discrete Fourier transform of a bivariate sequence of real data values.

2 Specification
#include <nag.h>
#include <nagc06.h>

void nag_sum_fft_real_2d (Integer m, Integer n, const double x[],
                    Complex y[], NagError *fail)

3 Description
nag_sum_fft_real_2d (c06pvc) computes the two-dimensional discrete Fourier transform of a bivariate sequence of real data values $x_{j_1,j_2}$, for $j_1 = 0, 1, \ldots, m - 1$ and $j_2 = 0, 1, \ldots, n - 1$.

The discrete Fourier transform is here defined by

$$
\hat{z}_{k_1,k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} x_{j_1,j_2} \exp \left( -2\pi i \left( \frac{j_1 k_1}{m} + \frac{j_2 k_2}{n} \right) \right),
$$

where $k_1 = 0, 1, \ldots, m - 1$ and $k_2 = 0, 1, \ldots, n - 1$. (Note the scale factor of $\frac{1}{\sqrt{mn}}$ in this definition.)

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

A call of nag_sum_fft_real_2d (c06pvc) followed by a call of nag_sum_fft_hermitian_2d (c06pwc) 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: $m$ – Integer
   
   On entry: $m$, the first dimension of the transform.
   
   Constraint: $m \geq 1$.

2: $n$ – Integer
   
   On entry: $n$, the second dimension of the transform.
   
   Constraint: $n \geq 1$.
3: \(x[m \times n] \) – const double

On entry: the real input dataset \(x\), where \(x_{j_1,j_2}\) is stored in \(x[j_2 \times m + j_1]\), for \(j_1 = 0, 1, \ldots, m - 1\) and \(j_2 = 0, 1, \ldots, n - 1\).

4: \(y[(m/2 + 1) \times n] \) – Complex

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

5: \(\text{fail} \) – NagError *

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 \(\langle\text{value}\rangle\) had an illegal value.

**NE_INT**
On entry, \(m = \langle\text{value}\rangle\).
Constraint: \(m \geq 1\).

On entry, \(n = \langle\text{value}\rangle\).
Constraint: \(n \geq 1\).

**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 forward transform using \(\text{nag_sum_fft_real_2d (c06pvc)}\) and a backward transform using \(\text{nag_sum_fft_hermitian_2d (c06pwc)}\), and comparing the results with the original sequence (in exact arithmetic they would be identical).

8 Parallelism and Performance

\(\text{nag_sum_fft_real_2d (c06pvc)}\) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\(\text{nag_sum_fft_real_2d (c06pvc)}\) 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_2d (c06pvc) is approximately proportional to $mn \log (mn)$, but also depends on the factors of $m$ and $n$. nag_sum_fft_real_2d (c06pvc) is fastest if the only prime factors of $m$ and $n$ are 2, 3 and 5, and is particularly slow if $m$ or $n$ is a large prime, or has large prime factors.

Workspace is internally allocated by nag_sum_fft_real_2d (c06pvc). The total size of these arrays is approximately proportional to $mn$.

10 Example

This example reads in a bivariate sequence of real data values and prints their discrete Fourier transforms as computed by nag_sum_fft_real_2d (c06pvc). Inverse transforms are then calculated by calling nag_sum_fft_hermitian_2d (c06pwc) showing that the original sequences are restored.

10.1 Program Text

/* nag_sum_fft_real_2d (c06pvc) 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, i, m, n;
    /* Arrays */
    Complex *y = 0;
    double *x = 0;
    char title[60];
    /* Nag Types */
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_sum_fft_real_2d (c06pvc) Example Program Results\n");
    fflush(stdout);

    /* Read dimensions of array from data file. */
    #ifdef _WIN32
    scanf_s("%*[\n]%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &m, &n);
    #else
    scanf("%*[\n]%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &m, &n);
    #endif

    if (! (x = NAG_ALLOC(m*n, double)) ||
        !(y = NAG_ALLOC((m/2+1)*n, Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read array values from data file and print out. */
    for (i = 0; i < m*n; i++)
```c
#ifdef _WIN32
  scanf_s("%lf", &x[i]);
#else
  scanf("%lf", &x[i]);
#endif

#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n Original data values\n");
#else
  sprintf(title, "\n Original data values\n");
#endif

nag_gen_real_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, x, m, "%6.3f", title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0, NULL, &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 two-dimensional real-to-complex discrete Fourier transform using * nag_sum_fft_real_2d (c06pvc) and print out. */

nag_sum_fft_real_2d(m, n, x, y, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_sum_fft_real_2d (c06pvc).\n%s\n", fail.message);
  exit_status = 2;
  goto END;
}

#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n Components of discrete Fourier transform\n");
#else
  sprintf(title, "\n Components of discrete Fourier transform\n");
#endif

nag_gen_complx_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m/2+1, y, m/2+1, Nag_BracketForm, "%6.3f", title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0, NULL, &fail);

if (fail.code != NE_NOERROR) {
  printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
  exit_status = 3;
  goto END;
}

/* Compute two-dimensional complex-to-real discrete Fourier transform using * nag_sum_fft_hermitian_2d (c06pwc) and print out. */

nag_sum_fft_hermitian_2d(m, n, y, x, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_sum_fft_hermitian_2d (c06pwc).\n%s\n", fail.message);
  exit_status = 4;
  goto END;
}

#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n Original sequence as restored by inverse transform\n");
#else
  sprintf(title, "\n Original sequence as restored by inverse transform\n");
#endif

```
10.2 Program Data

nag_sum_fft_real_2d (c06pvc) Example Program Data

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>0.010</td>
<td>1.284</td>
</tr>
<tr>
<td>0.346</td>
<td>1.960</td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_sum_fft_real_2d (c06pvc) Example Program Results

Original data values

| 0.010 | 1.284 | 1.754 | 0.089 | 1.004 |
| 0.346 | 1.960 | 0.855 | 0.161 | 1.844 |

Components of discrete Fourier transform

(2.943, 0.000) (-0.024, -0.558) (-1.167, 0.636)
(-0.324, 0.000) (-0.466, -0.230) (0.362, 0.262)

Original sequence as restored by inverse transform

| 0.010 | 1.284 | 1.754 | 0.089 | 1.004 |
| 0.346 | 1.960 | 0.855 | 0.161 | 1.844 |