NAG Library Function Document

nag_fft_multiple_sine (c06hac)

1 Purpose

nag_fft_multiple_sine (c06hac) computes the discrete Fourier sine transforms of \( m \) sequences of real data values.

2 Specification

```c
#include <nag.h>
#include <nagc06.h>
void nag_fft_multiple_sine (Integer m, Integer n, double x[],
const double trig[], NagError *fail)
```

3 Description

Given \( m \) sequences of \( n-1 \) real data values \( x_p^j \), for \( j = 1, 2, \ldots, n-1 \) and \( p = 1, 2, \ldots, m \), this function simultaneously calculates the Fourier sine transforms of all the sequences defined by

\[
\widehat{x}_k^p = \sqrt{\frac{2}{n}} \sum_{j=1}^{n-1} x_j^p \sin\left(\frac{jk\pi}{n}\right), \quad k = 1, 2, \ldots, n-1 \text{ and } p = 1, 2, \ldots, m.
\]

(Note the scale factor \( \sqrt{\frac{2}{n}} \) in this definition.)

The Fourier sine transform defined above is its own inverse, and two consecutive calls of this function with the same data will restore the original data (but see Section 9).

The transform calculated by this function can be used to solve Poisson’s equation when the solution is specified at both left and right boundaries (Swarztrauber (1977)).

The function uses a variant of the fast Fourier transform (FFT) algorithm (Brigham (1974)) known as the Stockham self-sorting algorithm, described in Temperton (1983), together with pre- and post-processing stages described in Swarztrauber (1982). Special coding is provided for the factors 2, 3, 4, 5 and 6.

4 References


5 Arguments

1: \( m – \text{ Integer} \)

\( \text{Input} \)

\( \text{On entry:} \) the number of sequences to be transformed, \( m \).

\( \text{Constraint:} \ m \geq 1. \)
2: \( n \) – Integer

*Input*

*On entry:* one more than the number of real values in each sequence, i.e., the number of values in each sequence is \( n - 1 \).

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

3: \( x[m \times (n - 1)] \) – double

*Input/Output*

*On entry:* the \( m \) data sequences stored in \( x \) consecutively. If the \( n - 1 \) data values of the \( p \)th sequence to be transformed are denoted by \( x_p^j \), for \( j = 1, 2, \ldots, n - 1 \) and \( p = 1, 2, \ldots, m \), then the first \( m(n - 1) \) elements of the array \( x \) must contain the values

\[
x_1^1, x_2^1, \ldots, x_{n-1}^1, x_1^2, x_2^2, \ldots, x_{n-1}^2, \ldots, x_1^m, x_2^m, \ldots, x_{n-1}^m.
\]

*On exit:* the \( m \) Fourier sine transforms stored consecutively, overwriting the corresponding original sequence.

4: \( \text{trig}[2 \times n] \) – const double

*Input*

*On entry:* trigonometric coefficients as returned by a call of \text{nag_fft_init_trig} (c06gzc). \text{nag_fft_multiple_sine} (c06hac) makes a simple check to ensure that \( \text{trig} \) has been initialized and that the initialization is compatible with the value of \( n \).

5: \( \text{fail} \) – \text{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.

**NE_C06_NOT_TRIG**

Value of \( n \) and \( \text{trig} \) array are incompatible or \( \text{trig} \) array not initialized.

**NE_INT_ARG_LT**

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

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

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

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

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

Not applicable.

9 Further Comments

The time taken is approximately proportional to \( nm\log(n) \), but also depends on the factors of \( n \). The function is fastest if the only prime factors of \( n \) are 2, 3 and 5, and is particularly slow if \( n \) is a large prime, or has large prime factors.
10 Example

This program reads in sequences of real data values and prints their Fourier sine transforms (as computed by nag_fft_multiple_sine (c06hac)). It then calls nag_fft_multiple_sine (c06hac) again and prints the results which may be compared with the original sequence.

10.1 Program Text

/* nag_fft_multiple_sine (c06hac) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 */

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

#define X(I, J) x[(I) *row_len + (J)]

int main(void)
{
    Integer exit_status = 0, i, j, m, n, row_len;
    NagError fail;
    double *trig = 0, *x = 0;
    INIT_FAIL(fail);
    printf("nag_fft_multiple_sine (c06hac) Example Program Results\n");
    if (m >= 1 && n >= 1)
    {
        if (!(trig = NAG_ALLOC(2*n, double)) ||
            !(x = NAG_ALLOC(m*(n-1), double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    row_len = n - 1;
    scanf(" %*[\n]"); /* Skip heading in data file */
    scanf(" %*[\n]"); /* Skip text in data file */
    c06 – Fourier Transforms

for (i = 0; i < m; ++i)
for (j = 0; j < row_len; ++j)
#ifdef _WIN32
    scanf_s("%lf", &X(i, j));
#else
    scanf("%lf", &X(i, j));
#endif
printf("Original data values\n\n");
for (i = 0; i < m; ++i)
{
    for (j = 0; j < row_len; ++j)
        printf(" %10.4f%s", X(i, j),
               (j%7 == 6 && j != row_len-1?"\n":"");
    printf("\n");
}
/* nag_fft_init_trig (c06gzc).
 * Initialization function for other c06 functions
*/
ag_fft_init_trig(n, trig, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_init_trig (c06gzc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Initialise trig array */
/* nag_fft_multiple_sine (c06hac).
 * Discrete sine transform
*/
ag_fft_multiple_sine(m, n, x, trig, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multiple_sine (c06hac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute transform */
printf("Discrete Fourier sine transforms\n\n");
for (i = 0; i < m; ++i)
{
    for (j = 0; j < row_len; ++j)
        printf(" %10.4f%s", X(i, j),
               (j%7 == 6 && j != row_len-1?"\n":"");
    printf("\n");
}
/* nag_fft_multiple_sine (c06hac), see above. */
ag_fft_multiple_sine(m, n, x, trig, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_fft_multiple_sine (c06hac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute inverse transform */
printf("Original data as restored by inverse transform\n\n");
for (i = 0; i < m; ++i)
{
    for (j = 0; j < row_len; ++j)
        printf(" %10.4f%s", X(i, j),
               (j%7 == 6 && j != row_len-1?"\n":"");
    printf("\n");
}
END:
NAG_FREE(trig);
NAG_FREE(x);
}
return exit_status;
}

10.2 Program Data

nag_fft_multiple_sine (c06hac) Example Program Data
3  6 : Number of sequences, m, (number of values in each sequence)+1, n
Real data sequences
0.6772  0.1138  0.6751  0.6362  0.1424
0.2983  0.1181  0.7255  0.8638  0.8723
0.0644  0.6037  0.6430  0.0428  0.4815

10.3 Program Results

nag_fft_multiple_sine (c06hac) Example Program Results

Original data values

<table>
<thead>
<tr>
<th></th>
<th>0.6772</th>
<th>0.1138</th>
<th>0.6751</th>
<th>0.6362</th>
<th>0.1424</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2983</td>
<td>0.1181</td>
<td>0.7255</td>
<td>0.8638</td>
<td>0.8723</td>
</tr>
<tr>
<td></td>
<td>0.0644</td>
<td>0.6037</td>
<td>0.6430</td>
<td>0.0428</td>
<td>0.4815</td>
</tr>
</tbody>
</table>

Discrete Fourier sine transforms

<table>
<thead>
<tr>
<th></th>
<th>1.0014</th>
<th>0.0062</th>
<th>0.0834</th>
<th>0.5286</th>
<th>0.2514</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.2477</td>
<td>-0.6599</td>
<td>0.2570</td>
<td>0.0858</td>
<td>0.2658</td>
</tr>
<tr>
<td></td>
<td>0.8521</td>
<td>0.0719</td>
<td>-0.0561</td>
<td>-0.4890</td>
<td>0.2056</td>
</tr>
</tbody>
</table>

Original data as restored by inverse transform

<table>
<thead>
<tr>
<th></th>
<th>0.6772</th>
<th>0.1138</th>
<th>0.6751</th>
<th>0.6362</th>
<th>0.1424</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2983</td>
<td>0.1181</td>
<td>0.7255</td>
<td>0.8638</td>
<td>0.8723</td>
</tr>
<tr>
<td></td>
<td>0.0644</td>
<td>0.6037</td>
<td>0.6430</td>
<td>0.0428</td>
<td>0.4815</td>
</tr>
</tbody>
</table>