NAG Library Function Document
nag_sum_fft_sine (c06rec)

1 Purpose
nag_sum_fft_sine (c06rec) computes the discrete Fourier sine transforms of \( m \) sequences of real data values. The elements of each sequence and its transform are stored contiguously.

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

3 Description
Given \( m \) sequences of \( n-1 \) real data values \( x_j^p \), for \( j = 1, 2, \ldots, n-1 \) and \( p = 1, 2, \ldots, m \), nag_sum_fft_sine (c06rec) simultaneously calculates the Fourier sine transforms of all the sequences defined by

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

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

This transform is also known as type-I DST.

Since the Fourier sine transform defined above is its own inverse, two consecutive calls of nag_sum_fft_sine (c06rec) will restore the original data.

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 (see Swarztrauber (1977)).

The function uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockholm 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 and 5.

4 References

5 Arguments
1: \( m \) – Integer

\textit{Input}

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

Constraint: \( m \geq 1 \).
2:  \textbf{n} \quad \text{Integer} \quad \text{Input}

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

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

3:  \mathbf{x}[(n - 1) \times m] \quad \text{double} \quad \text{Input/Output}

\textit{On entry:} the \( p \)th sequence to be transformed, denoted by \( x^p_j \), for \( j = 1, 2, \ldots, n - 1 \) and \( p = 1, 2, \ldots, m \), must be stored in \( x[(p - 1) \times (n - 1) + j - 1] \).

\textit{On exit:} the \( m \) Fourier sine transforms, overwriting the corresponding original sequences. The \((n - 1)\) components of the \( p \)th Fourier sine transform, denoted by \( x^p_k \), for \( k = 1, 2, \ldots, n - 1 \) and \( p = 1, 2, \ldots, m \), are stored in \( x[(p - 1) \times (n - 1) + k - 1] \).

4:  \textbf{fail} \quad \text{NagError} * \quad \text{Input/Output}

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

6 \quad \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, \( m = \langle \text{value} \rangle \).

Constraint: \( m \geq 1 \).

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

Constraint: \( n \geq 1 \).

\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 \quad \textbf{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 \quad \textbf{Parallelism and Performance}

\texttt{c06rec} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
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_sine (c06rec) is approximately proportional to $nm\log(n)$, but also depends on the factors of $n$. nag_sum_fft_sine (c06rec) 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. Workspace is internally allocated by this function. The total amount of memory allocated is $O(n)$ double values.

10 Example

This example reads in sequences of real data values and prints their Fourier sine transforms (as computed by nag_sum_fft_sine (c06rec)). It then calls nag_sum_fft_sine (c06rec) again and prints the results which may be compared with the original sequence.

10.1 Program Text

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

int main(void)
{
    /* Scalars */
    Integer exit_status = 0, j, m, n, n1;
    /* Arrays */
    double *x = 0;
    char title[60];
    /* Nag Types */
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_sum_fft_sine (c06rec) Example Program Results\n");
    fflush(stdout);

    /* Read dimensions of array from data file. */
    #ifdef _WIN32
    scanf_s("%*\[\n\] %" NAG_IFMT %" NAG_IFMT %"[\n]\", &m, &n1);
    #else
    scanf("%*\[\n\] %" NAG_IFMT %" NAG_IFMT %"[\n]\", &m, &n1);
    #endif
    n=n1+1;

    if (!(x = NAG_ALLOC((m * n1), double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read array values from data file and print out. */
    for (j = 0; j < m*n1; j++)
    #ifdef _WIN32
    scanf_s("%lf", &x[j]);
    #else
    scanf("%lf", &x[j]);
    #endif
```
```c
#include <stdio.h>
#include <nag.h>

int main(void)
{
    int m = 5, n = 3, n1 = 3;
    double x[n1] = {1.0, 2.0, 3.0, 4.0, 5.0};

    // Original data values
    printf("%9.4f\n", x[0]);
    printf("%9.4f\n", x[1]);
    printf("%9.4f\n", x[2]);
    printf("%9.4f\n", x[3]);
    printf("%9.4f\n", x[4]);

    // Discrete sine transforms
    nag_sum_fft_sine(m, n, x, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_sum_fft_sine (c06rec).\n%s\n", fail.message);
        exit_status = 2;
        goto END;
    }

    // Original data as restored by inverse transform
    printf("%9.4f\n", x[0]);
    printf("%9.4f\n", x[1]);
    printf("%9.4f\n", x[2]);
    printf("%9.4f\n", x[3]);
    printf("%9.4f\n", x[4]);

    return 0;
}
```

The code above demonstrates the usage of the `nag_sum_fft_sine` function from the NAG Library to perform discrete sine transforms on an array of data. It also includes code to print the original data values and the restored data values, handling errors with appropriate messages.
printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
}

END:
    NAG_FREE(x);
    return exit_status;
}

10.2 Program Data

nag_sum_fft_sine (c06rec) Example Program Data

3  5 : m, n-1

0.6772  0.1138  0.6751  0.6362  0.1424
0.2983  0.1182  0.7255  0.8638  0.8723
0.0644  0.6037  0.6430  0.0428  0.4815 : x

10.3 Program Results

nag_sum_fft_sine (c06rec) Example Program Results

Original data values

0.6772  0.1138  0.6751  0.6362  0.1424
0.2983  0.1182  0.7255  0.8638  0.8723
0.0644  0.6037  0.6430  0.0428  0.4815

Discrete Fourier sine transforms

1.0014  0.0062  0.0834  0.5286  0.2514
1.2478 -0.6598  0.2570  0.0858  0.2658
0.8521  0.0719 -0.0561 -0.4890  0.2056

Original data as restored by inverse transform

0.6772  0.1138  0.6751  0.6362  0.1424
0.2983  0.1182  0.7255  0.8638  0.8723
0.0644  0.6037  0.6430  0.0428  0.4815