NAG Toolbox

nag_sum_fft_real_sine_simple (c06ra)

1 Purpose

nag_sum_fft_real_sine_simple (c06ra) computes the discrete Fourier sine transforms of m sequences of real data values.

2 Syntax

```
[x, ifail] = nag_sum_fft_real_sine_simple(m, n, x)
[x, ifail] = c06ra(m, n, x)
```

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_real_sine_simple (c06ra) simultaneously calculates the Fourier sine transforms of all the sequences defined by

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

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

Since the Fourier sine transform defined above is its own inverse, two consecutive calls of this function 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 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 and 5.

4 References

Brigham E O (1974) *The Fast Fourier Transform* Prentice–Hall

Swarztrauber P N (1977) The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle *SIAM Rev.* **19(3)** 490–501

Swarztrauber P N (1982) Vectorizing the FFT's *Parallel Computation* (ed G Rodrique) 51–83 Academic Press

Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

5 Parameters

5.1 Compulsory Input Parameters

1: **m** – INTEGER

m, the number of sequences to be transformed.

Constraint: **m** ≥ 1.

2: **n** – INTEGER

One more than the number of real values in each sequence, i.e., the number of values in each sequence is $n-1$.

Constraint: $\mathbf{n} \geq 1$.

3: **x**($\mathbf{m} \times (\mathbf{n}+\mathbf{2})$) – REAL (KIND=nag_wp) array

the data must be stored in **x** as if in a two-dimensional array of dimension $(1:\mathbf{m}, 1:\mathbf{n}+2)$; each of the m sequences is stored in a **row** of the array. In other words, if the $n-1$ data values of the pth sequence to be transformed are denoted by x_j^p, 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

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

The nth to $(n+2)$th elements of each row $x_n^p, \ldots, x_{n+2}^p$, for $p=1,2,\ldots,m$, are required as workspace. These $3m$ elements may contain arbitrary values as they are set to zero by the function.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **x**($\mathbf{m} \times (\mathbf{n}+\mathbf{2})$) – REAL (KIND=nag_wp) array

the m Fourier sine transforms stored as if in a two-dimensional array of dimension $(1:\mathbf{m}, 1:\mathbf{n}+2)$. Each of the m transforms is stored in a **row** of the array, overwriting the corresponding original sequence. If the $(n-1)$ components of the pth Fourier sine transform are denoted by $\hat{x}_k^p$, for $k=1,2,\ldots,n-1$ and $p=1,2,\ldots,m$, then the $m(n+2)$ elements of the array **x** contain the values

$$\hat{x}_1^1, \hat{x}_1^2, \ldots, \hat{x}_1^m, \hat{x}_2^1, \hat{x}_2^2, \ldots, \hat{x}_2^m, \ldots, \hat{x}_{n-1}^1, \hat{x}_{n-1}^2, \ldots, \hat{x}_{n-1}^m, 0, 0, \ldots, 0 \ (3m \text{ times}).$$

2: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{m} < 1$.

ifail $= 2$

On entry, $\mathbf{n} < 1$.

ifail $= 3$

An unexpected error has occurred in an internal call. Check all function calls and array dimensions. Seek expert help.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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 Further Comments

The time taken by nag_sum_fft_real_sine_simple (c06ra) is approximately proportional to $nm\log(n)$, but also depends on the factors of n. nag_sum_fft_real_sine_simple (c06ra) 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.

9 Example

This example reads in sequences of real data values and prints their Fourier sine transforms (as computed by nag_sum_fft_real_sine_simple (c06ra)). It then calls nag_sum_fft_real_sine_simple (c06ra) again and prints the results which may be compared with the original sequence.

9.1 Program Text

```
function c06ra_example

fprintf('c06ra example results\n\n');

% Discrete sine transform of 3 sequences of length 5
m = nag_int(3);
n = nag_int(6);
x = zeros(m,n+2);
x(1:m,1:n-1) = [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];

[xt, ifail] = c06ra(m, n, x);

disp('X under discrete sine transform:');
y = reshape(xt(1:m*(n-1)),m,n-1);
disp(y);

% Reconstruct using same transform
[xr, ifail] = c06ra(m, n, xt);

disp('X reconstructed under second sine transform:');
y = reshape(x(1:m*(n-1)),m,n-1);
disp(y);
```

9.2 Program Results

```
c06ra example results

X under discrete sine transform:
    1.0014    0.0062    0.0834    0.5286    0.2514
    1.2477   -0.6599    0.2570    0.0859    0.2658
    0.8521    0.0719   -0.0561   -0.4890    0.2056

X reconstructed under second sine transform:
    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
```