NAG Toolbox

nag_sum_fft_real_qtrsine_simple (c06rc)

1 Purpose

nag_sum_fft_real_qtrsine_simple (c06rc) computes the discrete quarter-wave Fourier sine transforms of m sequences of real data values.

2 Syntax

```
[x, ifail] = nag_sum_fft_real_qtrsine_simple(direct, m, n, x)
[x, ifail] = c06rc(direct, m, n, x)
```

3 Description

Given m sequences of n real data values x_j^p, for $j = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$, nag_sum_fft_real_qtrsine_simple (c06rc) simultaneously calculates the quarter-wave Fourier sine transforms of all the sequences defined by

$$\hat{x}_k^p = \frac{1}{\sqrt{n}}\left(\sum_{j=1}^{n-1} x_j^p \times \sin\left(j(2k-1)\frac{\pi}{2n}\right) + \tfrac{1}{2}(-1)^{k-1} x_n^p\right), \qquad \text{if } \textbf{direct} = \text{'F'},$$

or its inverse

$$x_k^p = \frac{2}{\sqrt{n}}\sum_{j=1}^{n} \hat{x}_j^p \times \sin\left((2j-1)k\frac{\pi}{2n}\right), \qquad \text{if } \textbf{direct} = \text{'B'},$$

where $k = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$.

(Note the scale factor $\frac{1}{\sqrt{n}}$ in this definition.)

A call of nag_sum_fft_real_qtrsine_simple (c06rc) with **direct** = 'F' followed by a call with **direct** = 'B' 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 the left boundary, and the derivative of the solution is specified at the right boundary (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: **direct** – CHARACTER(1)

If the forward transform as defined in Section 3 is to be computed, then **direct** must be set equal to `F'.

If the backward transform is to be computed then **direct** must be set equal to `B'.

Constraint: **direct** = 'F' or 'B'.

2: **m** – INTEGER

m, the number of sequences to be transformed.

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

3: **n** – INTEGER

n, the number of real values in each sequence.

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

4: **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 data values of the pth sequence to be transformed are denoted by x_j^p, for $j = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$, then the first mn elements of the array **x** must contain the values

$$x_1^1, x_1^2, \ldots, x_1^m, x_2^1, x_2^2, \ldots, x_2^m, \ldots, x_n^1, x_n^2, \ldots, x_n^m.$$

The $(n+1)$th and $(n+2)$th elements of each row x_{n+1}^p, x_{n+2}^p, for $p = 1, 2, \ldots, m$, are required as workspace. These $2m$ 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 quarter-wave 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 components of the pth quarter-wave sine transform are denoted by $\hat{x}_k^p$, for $k = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$, then the $m(n+2)$ elements of the array **x** contain the values

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

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, **n** < 1.

ifail $= 3$

On entry, **direct** $\neq$ 'F' or 'B'.

ifail $= 4$

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_qtrsine_simple (c06rc) is approximately proportional to $nm\log(n)$, but also depends on the factors of n. nag_sum_fft_real_qtrsine_simple (c06rc) 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 quarter-wave sine transforms as computed by nag_sum_fft_real_qtrsine_simple (c06rc) with **direct** = 'F'. It then calls the function again with **direct** = 'B' and prints the results which may be compared with the original data.

9.1 Program Text

```
     function c06rc_example

fprintf('c06rc example results\n\n');

% Discrete quarter-wave sine transform of 3 sequences of length 6
direct = 'Forward';
m = nag_int(3);
n = nag_int(6);
x = zeros(m,(n+2));
x(1:m,1:n) = [ 0.3854  0.6772  0.1138  0.6751  0.6362  0.1424;
               0.5417  0.2983  0.1181  0.7255  0.8638  0.8723;
               0.9172  0.0644  0.6037  0.6430  0.0428  0.4815];

[xt, ifail] = c06rc(direct, m, n, x);
disp('X under discrete quarter-wave sine transform:');
disp(reshape(xt(1:m*n),m,n));

% Reconstruct using same transform
```

```
direct = 'Backward';
[xr, ifail] = c06rc(direct, m, n, xt);
disp('X reconstructed by inverse quarter-wave sine transform:');
y = reshape(xr(1:m*n),m,n);
disp(y);
```

9.2 Program Results

```
     c06rc example results

X under discrete quarter-wave sine transform:
    0.7304    0.2078    0.1150    0.2577   -0.2869   -0.0815
    0.9274   -0.1152    0.2532    0.2883   -0.0026   -0.0635
    0.6268    0.3547    0.0760    0.3078    0.4987   -0.0507

X reconstructed by inverse quarter-wave sine transform:
    0.3854    0.6772    0.1138    0.6751    0.6362    0.1424
    0.5417    0.2983    0.1181    0.7255    0.8638    0.8723
    0.9172    0.0644    0.6037    0.6430    0.0428    0.4815
```