NAG Toolbox

nag_sum_fft_qtrsine (c06rg)

1 Purpose

nag_sum_fft_qtrsine (c06rg) computes the discrete quarter-wave Fourier sine transforms of m sequences of real data values. The elements of each sequence and its transform are stored contiguously.

2 Syntax

```
[x, ifail] = nag_sum_fft_qtrsine(idir, x, 'm', m, 'n', n)
[x, ifail] = c06rg(idir, x, 'm', m, 'n', n)
```

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_qtrsine (c06rg) simultaneously calculates the quarter-wave Fourier sine transforms of all the sequences defined by

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

or its inverse

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

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_qtrsine (c06rg) with $\mathbf{idir} = 1$ followed by a call with $\mathbf{idir} = -1$ will restore the original data.

The two transforms are also known as type-III DST and type-II DST, respectively.

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: **idir** – INTEGER

Indicates the transform, as defined in Section 3, to be computed.

idir $= 1$
  Forward transform.

idir $= -1$
  Inverse transform.

Constraint: **idir** $= 1$ or -1.

2: $\mathbf{x}(\mathbf{n},\mathbf{m})$ – REAL (KIND=nag_wp) array

The data values of the pth sequence to be transformed, denoted by x_j^p, for $j = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$, must be stored in $\mathbf{x}(j,p)$.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the second dimension of the array **x**.

m, the number of sequences to be transformed.

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

2: **n** – INTEGER

Default: the first dimension of the array **x**.

n, the number of real values in each sequence.

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

5.3 Output Parameters

1: $\mathbf{x}(\mathbf{n},\mathbf{m})$ – REAL (KIND=nag_wp) array

The n components of the pth quarter-wave sine transform, denoted by $\hat{x}_k^p$, for $k = 1, 2, \ldots, n$ and $p = 1, 2, \ldots, m$, are stored in $\mathbf{x}(k,p)$, overwriting the corresponding original values.

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 $= 1$

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

ifail $= 2$

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

ifail $= 3$

  Constraint: **idir** $= -1$ or 1.

ifail $= 4$

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.

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_qtrsine (c06rg) is approximately proportional to $nm\log(n)$, but also depends on the factors of n. nag_sum_fft_qtrsine (c06rg) 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 of order $O(n)$ is internally allocated by this function.

9 Example

This example reads in sequences of real data values and prints their quarter-wave sine transforms as computed by nag_sum_fft_qtrsine (c06rg) with **idir** $= 1$. It then calls the function again with **idir** $= -1$ and prints the results which may be compared with the original data.

9.1 Program Text

```
     function c06rg_example

fprintf('c06rg example results\n\n');

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

idir = nag_int(1);
[x, ifail] = c06rg(idir,x);
disp('X under discrete quarter-wave sine transform:');
disp(x);

% Reconstruct using same transform
idir = -idir;
[x, ifail] = c06rg(idir,x);
disp('X reconstructed by inverse quarter-wave sine transform:');
disp(x);
```

9.2 Program Results

```
     c06rg example results

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

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