NAG Library Function Document

nag_sum_fft_qtrsine (c06rgc)

1 Purpose

nag_sum_fft_qtrsine (c06rgc) 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 Specification

```
#include <nag.h>
#include <nagc06.h>

void nag_sum_fft_qtrsine (Nag_TransformDirection direct, Integer m,
     Integer n, double x[], NagError *fail)
```

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 (c06rgc) simultaneously calculates the quarter-wave Fourier sine transforms of all the sequences defined by

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

or its inverse

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

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 (c06rgc) with **direct** $=$ Nag_ForwardTransform followed by a call with **direct** $=$ Nag_BackwardTransform 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 Arguments

1: **direct** – Nag_TransformDirection *Input*

On entry: indicates the transform, as defined in Section 3, to be computed.

direct = Nag_ForwardTransform
Forward transform.

direct = Nag_BackwardTransform
Inverse transform.

Constraint: **direct** = Nag_ForwardTransform or Nag_BackwardTransform.

2: **m** – Integer *Input*

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

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

3: **n** – Integer *Input*

On entry: n, the number of real values in each sequence.

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

4: $\mathbf{x}[\mathbf{n} \times \mathbf{m}]$ – double *Input/Output*

On entry: the m data sequences to be transformed. 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 $\mathbf{x}[(p-1) \times \mathbf{n} + j - 1]$.

On exit: the m quarter-wave sine transforms, overwriting the corresponding original sequences. 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 $\mathbf{x}[(p-1) \times \mathbf{n} + k - 1]$.

5: **fail** – 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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 1$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 1$.

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.

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 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

nag_sum_fft_qtrsine (c06rgc) 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_qtrsine (c06rgc) is approximately proportional to $nm\log(n)$, but also depends on the factors of n. nag_sum_fft_qtrsine (c06rgc) 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 quarter-wave sine transforms as computed by nag_sum_fft_qtrsine (c06rgc) with **direct** $=$ Nag_ForwardTransform. It then calls the function again with **direct** $=$ Nag_BackwardTransform and prints the results which may be compared with the original data.

10.1 Program Text

```
/* nag_sum_fft_qtrsine (c06rgc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

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

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

  INIT_FAIL(fail);

  printf("nag_sum_fft_qtrsine (c06rgc) Example Program Results\n");
  fflush(stdout);

  /* Read dimensions of array from data file. */
#ifdef _WIN32
```

```
  scanf_s("%*[^\n] %" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &m, &n);
#else
  scanf("%*[^\n] %" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &m, &n);
#endif
  if (!(x = NAG_ALLOC((m * n), double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read array values from data file and print out. */
  for (j = 0; j < m*n; j++)
#ifdef _WIN32
    scanf_s("%lf", &x[j]);
#else
    scanf("%lf", &x[j]);
#endif
#ifdef _WIN32
  sprintf_s(title, _countof(title), "\n Original data values\n");
#else
  sprintf(title, "\n Original data values\n");
#endif
  nag_gen_real_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                              Nag_NonUnitDiag, m, n, x, n, "%9.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels,
                              0, 80, 0, NULL, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_sum_fft_qtrsine (c06rgc).
   * Discrete quarter-wave sine transforms
   */
  nag_sum_fft_qtrsine(Nag_ForwardTransform, m, n, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_sum_fft_qtrsine (c06rgc).\n%s\n",
             fail.message);
      exit_status = 2;
      goto END;
    }

#ifdef _WIN32
  sprintf_s(title, _countof(title),
            "\n Discrete quarter-wave Fourier sine transforms\n");
#else
  sprintf(title, "\n Discrete quarter-wave Fourier sine transforms\n");
#endif
  nag_gen_real_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                              Nag_NonUnitDiag, m, n, x, n, "%9.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels,
                              0, 80, 0, NULL, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
             fail.message);
      exit_status = 3;
      goto END;
    }

  /* Call backward transform to restore the original data. */
  nag_sum_fft_qtrsine(Nag_BackwardTransform, m, n, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_sum_fft_qtrsine (c06rgc).\n%s\n",
             fail.message);
```

```
      exit_status = 4;
      goto END;
    }

#ifdef _WIN32
  sprintf_s(title, _countof(title),
            "\n Original data as restored by inverse transform\n");
#else
  sprintf(title, "\n Original data as restored by inverse transform\n");
#endif
  nag_gen_real_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                              Nag_NonUnitDiag, m, n, x, n, "%9.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels,
                              0, 80, 0, NULL, &fail);
  if (fail.code != NE_NOERROR)
    {
      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_qtrsine (c06rgc) Example Program Data

 3       6                                       : m, 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  : x
```

10.3 Program Results

```
nag_sum_fft_qtrsine (c06rgc) Example Program Results

 Original data values

     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

 Discrete quarter-wave Fourier sine transforms

     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

 Original data as restored by inverse 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
```