

NAG Library Function Document

nag_sum_fft_complex_1d_multi (c06psc)

1 Purpose

nag_sum_fft_complex_1d_multi (c06psc) computes the discrete Fourier transforms of m sequences each containing n complex data values.

2 Specification

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

void nag_sum_fft_complex_1d_multi (Nag_TransformDirection direct, Integer n,
    Integer m, Complex x[], NagError *fail)
```

3 Description

Given m sequences of n complex data values z_j^p , for $j = 0, 1, \dots, n-1$ and $p = 1, 2, \dots, m$, nag_sum_fft_complex_1d_multi (c06psc) simultaneously calculates the (**forward** or **backward**) discrete Fourier transforms of all the sequences defined by

$$\hat{z}_k^p = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j^p \times \exp\left(\pm i \frac{2\pi jk}{n}\right), \quad k = 0, 1, \dots, n-1 \text{ and } p = 1, 2, \dots, m.$$

(Note the scale factor $\frac{1}{\sqrt{n}}$ in this definition.) The minus sign is taken in the argument of the exponential within the summation when the forward transform is required, and the plus sign is taken when the backward transform is required.

A call of nag_sum_fft_complex_1d_multi (c06psc) with **direct** = Nag_ForwardTransform followed by a call with **direct** = Nag_BackwardTransform will restore the original data.

The function uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, which is described in Temperton (1983). Special code is provided for the factors 2, 3 and 5.

4 References

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

Temperton C (1983) Self-sorting mixed-radix fast Fourier transforms *J. Comput. Phys.* **52** 1–23

5 Arguments

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

On entry: if the forward transform as defined in Section 3 is to be computed, then **direct** must be set equal to Nag_ForwardTransform.

If the backward transform is to be computed then **direct** must be set equal to Nag_BackwardTransform.

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

2: **n** – Integer *Input*

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

Constraint: $n \geq 1$.

- 3: **m** – Integer *Input*
On entry: m , the number of sequences to be transformed.
Constraint: $m \geq 1$.
- 4: **x[n × m]** – Complex *Input/Output*
On entry: the complex data values z_j^p stored in $x[(p-1) \times n + j]$, for $j = 0, 1, \dots, n-1$ and $p = 1, 2, \dots, m$.
On exit: is overwritten by the complex transforms.
- 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.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.
" $\langle value \rangle$ " is an invalid value of **direct**.

NE_INT

On entry, **m** = $\langle value \rangle$.
Constraint: $m \geq 1$.
On entry, **n** = $\langle value \rangle$.
Constraint: $n \geq 1$.

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.

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_complex_1d_multi` (c06psc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_sum_fft_complex_1d_multi` (c06psc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by `nag_sum_fft_complex_1d_multi` (c06psc) is approximately proportional to $n \log(n)$, but also depends on the factors of n . `nag_sum_fft_complex_1d_multi` (c06psc) 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. This function internally allocates a workspace of $nm + n + 15$ Complex values.

10 Example

This example reads in sequences of complex data values and prints their discrete Fourier transforms (as computed by `nag_sum_fft_complex_1d_multi` (c06psc) with `direct = Nag_ForwardTransform`). Inverse transforms are then calculated using `nag_sum_fft_complex_1d_multi` (c06psc) with `direct = Nag_BackwardTransform` and printed out, showing that the original sequences are restored.

10.1 Program Text

```

/* nag_sum_fft_complex_1d_multi (c06psc) Example Program.
 *
 * Copyright 2013 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, i, m, n;
    /* Arrays */
    Complex *x = 0;
    char title[60];
    /* Nag Types */
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_sum_fft_complex_1d_multi (c06psc) Example Program Results\n\n");

    /* Read dimensions of array and array values from data file. */
    scanf("%m[^\n] %m NAG_IFMT %m NAG_IFMT %m[^]\n]", &m, &n);
    if (!(x = NAG_ALLOC((m * n), Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 0; i < m*n; i++)
        scanf(" ( %lf , %lf ) ", &x[i].re, &x[i].im);

    /* nag_gen_complx_mat_print_comp (x04dbc).
     * Print complex general matrix (comprehensive).
     */
    printf(title, "Original data values\n");
    nag_gen_complx_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                                  Nag_NonUnitDiag, m, n, x, n, Nag_AboveForm,
                                  "%7.4f", title, Nag_NoLabels, 0, Nag_NoLabels,
                                  0, 80, 0, NULL, &fail);

    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
              fail.message);
        exit_status = 1;
        goto END;
    }
}

```

```

    }

/* nag_sum_fft_complex_ld_multi (c06psc).
 * Multiple one-dimensional complex discrete Fourier transform (Forward).
 */
nag_sum_fft_complex_ld_multi(Nag_ForwardTransform, n, m, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sum_fft_complex_ld_multi (c06psc).\n%s\n",
           fail.message);
    exit_status = 2;
    goto END;
}
printf(title, "Components of discrete Fourier transform\n");
nag_gen_complx_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                              Nag_NonUnitDiag, m, n, x, n, Nag_AboveForm,
                              "%7.4f", title, Nag_NoLabels, 0, Nag_NoLabels,
                              0, 80, 0, NULL, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
           fail.message);
    exit_status = 3;
    goto END;
}

/* nag_sum_fft_complex_ld_multi (c06psc).
 * Multiple one-dimensional complex discrete Fourier transform (Backward).
 */
nag_sum_fft_complex_ld_multi(Nag_BackwardTransform, n, m, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sum_fft_complex_ld_multi (c06psc).\n%s\n",
           fail.message);
    exit_status = 4;
    goto END;
}
sprintf(title, "Original sequence as restored by inverse transform\n");
nag_gen_complx_mat_print_comp(Nag_RowMajor, Nag_GeneralMatrix,
                              Nag_NonUnitDiag, m, n, x, n, Nag_AboveForm,
                              "%7.4f", title, Nag_NoLabels, 0, Nag_NoLabels,
                              0, 80, 0, NULL, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
           fail.message);
    exit_status = 5;
}

END:
    NAG_FREE(x);
    return exit_status;
}

```

10.2 Program Data

nag_sum_fft_complex_ld_multi (c06psc) Example Program Data

```

3      6      : m, n
(0.3854,0.5417)
(0.6772,0.2983)
(0.1138,0.1181)
(0.6751,0.7255)
(0.6362,0.8638)
(0.1424,0.8723)
(0.9172,0.9089)
(0.0644,0.3118)
(0.6037,0.3465)
(0.6430,0.6198)
(0.0428,0.2668)
(0.4815,0.1614)

```

```

(0.1156,0.6214)
(0.0685,0.8681)
(0.2060,0.7060)
(0.8630,0.8652)
(0.6967,0.9190)
(0.2792,0.3355) : third sequence

```

10.3 Program Results

nag_sum_fft_complex_1d_multi (c06psc) 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
0.9089 0.3118 0.3465 0.6198 0.2668 0.1614

0.1156 0.0685 0.2060 0.8630 0.6967 0.2792
0.6214 0.8681 0.7060 0.8652 0.9190 0.3355

```

Components of discrete Fourier transform

```

1.0737 -0.5706 0.1733 -0.1467 0.0518 0.3625
1.3961 -0.0409 -0.2958 -0.1521 0.4517 -0.0321

1.1237 0.1728 0.4185 0.1530 0.3686 0.0101
1.0677 0.0386 0.7481 0.1752 0.0565 0.1403

0.9100 -0.3054 0.4079 -0.0785 -0.1193 -0.5314
1.7617 0.0624 -0.0695 0.0725 0.1285 -0.4335

```

Original sequence 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
0.9089 0.3118 0.3465 0.6198 0.2668 0.1614

0.1156 0.0685 0.2060 0.8630 0.6967 0.2792
0.6214 0.8681 0.7060 0.8652 0.9190 0.3355

```
