

nag_fft_2d_complex (c06fuc)

1. Purpose

nag_fft_2d_complex (c06fuc) computes the two-dimensional discrete Fourier transform of a bivariate sequence of complex data values.

2. Specification

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

```
void nag_fft_2d_complex(Integer m, Integer n, double x[], double y[],
    double trigm[], double trign[], NagError *fail)
```

3. Description

This function computes the two-dimensional discrete Fourier transform of a bivariate sequence of complex data values $z_{j_1 j_2}$, where $j_1 = 0, 1, \dots, m-1$, $j_2 = 0, 1, \dots, n-1$.

The discrete Fourier transform is here defined by

$$\hat{z}_{k_1 k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} z_{j_1 j_2} \exp \left(-2\pi i \left(\frac{j_1 k_1}{m} + \frac{j_2 k_2}{n} \right) \right)$$

for $k_1 = 0, 1, \dots, m-1$; $k_2 = 0, 1, \dots, n-1$.

(Note the scale factor of $1/\sqrt{mn}$ in this definition.)

The first call of **nag_fft_2d_complex** must be preceded by calls to **nag_fft_init_trig (c06gzc)** to initialise the **trigm** and **trign** arrays with trigonometric coefficients according to the value of **m** and **n** respectively.

To compute the inverse discrete Fourier transform, defined with $\exp(+2\pi i(\dots))$ in the above formula instead of $\exp(-2\pi i(\dots))$, this function should be preceded and followed by calls of **nag_conjugate_complex (c06gcc)** to form the complex conjugates of the data values and the transform.

This function calls **nag_fft_multiple_complex (c06frc)** to perform multiple one-dimensional discrete Fourier transforms by the fast Fourier transform algorithm in Brigham (1974).

4. Parameters

m

Input: the number of rows, m , of the bivariate data sequence.
Constraint: **m** ≥ 1 .

n

Input: the number of columns, n , of the bivariate data sequence.
Constraint: **n** ≥ 1 .

x[m*n]

y[m*n]

Input: the real and imaginary parts of the complex data values must be stored in arrays **x** and **y** respectively. Each row of the data must be stored consecutively; hence if the real parts of z_{j_1, j_2} are denoted by x_{j_1, j_2} , for $j_1 = 0, 1, \dots, m-1$, $j_2 = 0, 1, \dots, n-1$, then the mn elements of **x** must contain the values

$$x_{0,0}, x_{0,1}, \dots, x_{0,n-1}, x_{1,0}, x_{1,1}, \dots, x_{1,n-1}, \dots, x_{m-1,0}, x_{m-1,1}, \dots, x_{m-1,n-1}.$$

The imaginary parts must be ordered similarly in **y**.

Output: the real and imaginary parts respectively of the corresponding elements of the computed transform.

trigm[2*m]
trign[2*n]

Input: **trigm** and **trign** must contain trigonometric coefficients as returned by calls of `nag_fft_init_trig` (c06gzc). `nag_fft_2d_complex` performs a simple check to ensure that both arrays have been initialised and that they are compatible with **m** and **n**. If $m = n$ the same array may be supplied for **trigm** and **trign**.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LT

On entry, **m** must not be less than 1: **m** = *<value>*.

On entry, **n** must not be less than 1: **n** = *<value>*.

NE_C06_NOT_TRIG

Value of **n** and **trign** array are incompatible or **trign** array not initialised.

Value of **m** and **trigm** array are incompatible or **trigm** array not initialised.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

The time taken by the function is approximately proportional to $mn \log(mn)$, but also depends on the factorization of the individual dimensions m and n . The function is somewhat faster than average if their only prime factors are 2, 3 or 5; and fastest of all if they are powers of 2; it is particularly slow if m or n is a large prime, or has large prime factors.

6.1. 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).

6.2. 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.

7. See Also

`nag_fft_multiple_complex` (c06frc)
`nag_conjugate_complex` (c06gcc)
`nag_fft_init_trig` (c06gzc)

8. Example

This program reads in a bivariate sequence of complex data values and prints the two-dimensional Fourier transform. It then performs an inverse transform and prints the sequence so obtained, which may be compared to the original data values.

8.1. Program Text

```
/* nag_ftt_2d_complex(c06fuc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 */

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

```

#define MMAX 6
#define NMAX 6
#define MNMAX MMAX*NMAX

main()
{
    Integer i, j, m, n;
    double trigm[2*MMAX], trign[2*NMAX];
    double x[MNMAX], y[MNMAX];

    Vprintf("c06fuc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\\n]");
    while (scanf("%ld%ld", &m, &n)!=EOF)
        if (m*n>=1 && m*n<=MNMAX)
            {
                Vprintf("\\n\\nm = %2ld  n = %2ld\\n", m, n);
                /* Read in complex data and print out. */
                for (j = 0; j<m; ++j)
                    {
                        for (i = 0; i<n; ++i)
                            Vscanf("%lf", &x[j*n + i]);
                        for (i = 0; i<n; ++i)
                            Vscanf("%lf", &y[j*n + i]);
                    }
                Vprintf("\\nOriginal data values\\n\\n");
                for (j = 0; j<m; ++j)
                    {
                        Vprintf("Real");
                        for (i = 0; i<n; ++i)
                            Vprintf("%10.4f%s", x[j*n + i],
                                (i%6==5 && i!=n-1 ? "\\n      " : ""));
                        Vprintf("\\nImag");
                        for (i = 0; i<n; ++i)
                            Vprintf("%10.4f%s", y[j*n + i],
                                (i%6==5 && i!=n-1 ? "\\n      " : ""));
                        Vprintf("\\n\\n");
                    }
                /* Initialize trig arrays */
                c06gzc(m, trigm, NAGERR_DEFAULT);
                c06gzc(n, trign, NAGERR_DEFAULT);
                /* Compute transform */
                c06fuc(m, n, x, y, trigm, trign, NAGERR_DEFAULT);
                Vprintf("\\nComponents of discrete Fourier transforms\\n\\n");
                for (j = 0; j<m; ++j)
                    {
                        Vprintf("Real");
                        for (i = 0; i<n; ++i)
                            Vprintf("%10.4f%s", x[j*n + i],
                                (i%6==5 && i!=n-1 ? "\\n      " : ""));
                        Vprintf("\\nImag");
                        for (i = 0; i<n; ++i)
                            Vprintf("%10.4f%s", y[j*n + i],
                                (i%6==5 && i!=n-1 ? "\\n      " : ""));
                        Vprintf("\\n\\n");
                    }
                /* Compute inverse transform */
                c06gcc(m*n, y, NAGERR_DEFAULT);
                c06fuc(m, n, x, y, trigm, trign, NAGERR_DEFAULT);
                c06gcc(m*n, y, NAGERR_DEFAULT);
                Vprintf("\\nOriginal data as restored by inverse transform\\n\\n");
                for (j = 0; j<m; ++j)
                    {
                        Vprintf("Real");
                        for (i = 0; i<n; ++i)
                            Vprintf("%10.4f%s", x[j*n + i],
                                (i%6==5 && i!=n-1 ? "\\n      " : ""));
                        Vprintf("\\nImag");
                        for (i = 0; i<n; ++i)

```

```

        Vprintf("%10.4f%s", y[j*n + i],
                (i%6==5 && i!=n-1 ? "\n      " : ""));
        Vprintf("\n\n");
    }
}
else
    Vfprintf(stderr, "\nInvalid value of m or n.\n");
exit(EXIT_SUCCESS);
}

```

8.2. Program Data

c06fuc Example Program Data

```

3  5
    1.000    0.999    0.987    0.936    0.802
    0.000   -0.040   -0.159   -0.352   -0.597
    0.994    0.989    0.963    0.891    0.731
   -0.111   -0.151   -0.268   -0.454   -0.682
    0.903    0.885    0.823    0.694    0.467
   -0.430   -0.466   -0.568   -0.720   -0.884

```

8.3. Program Results

c06fuc Example Program Results

m = 3 n = 5

Original data values

Real	1.0000	0.9990	0.9870	0.9360	0.8020
Imag	0.0000	-0.0400	-0.1590	-0.3520	-0.5970
Real	0.9940	0.9890	0.9630	0.8910	0.7310
Imag	-0.1110	-0.1510	-0.2680	-0.4540	-0.6820
Real	0.9030	0.8850	0.8230	0.6940	0.4670
Imag	-0.4300	-0.4660	-0.5680	-0.7200	-0.8840

Components of discrete Fourier transforms

Real	3.3731	0.4814	0.2507	0.0543	-0.4194
Imag	-1.5187	-0.0907	0.1776	0.3188	0.4145
Real	0.4565	0.0549	0.0093	-0.0217	-0.0759
Imag	0.1368	0.0317	0.0389	0.0356	0.0045
Real	-0.1705	-0.0375	-0.0423	-0.0377	-0.0022
Imag	0.4927	0.0584	0.0082	-0.0255	-0.0829

Original data as restored by inverse transform

Real	1.0000	0.9990	0.9870	0.9360	0.8020
Imag	0.0000	-0.0400	-0.1590	-0.3520	-0.5970
Real	0.9940	0.9890	0.9630	0.8910	0.7310
Imag	-0.1110	-0.1510	-0.2680	-0.4540	-0.6820
Real	0.9030	0.8850	0.8230	0.6940	0.4670
Imag	-0.4300	-0.4660	-0.5680	-0.7200	-0.8840