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

   ```c
   #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 \( z_{j_1j_2}, \) where \( j_1 = 0, 1, \ldots, m - 1, j_2 = 0, 1, \ldots, n - 1. \)

The discrete Fourier transform is here defined by

\[
\hat{z}_{k_1k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} z_{j_1j_2} \exp \left( -2\pi i \left( \frac{j_1k_1}{m} + \frac{j_2k_2}{n} \right) \right)
\]

for \( k_1 = 0, 1, \ldots, m - 1; k_2 = 0, 1, \ldots, n - 1. \)

(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 \( \text{trigm} \) and \( \text{trign} \) arrays with trigonometric coefficients according to the value of \( m \) and \( n \) respectively.

To compute the inverse discrete Fourier transform, defined with \( \exp(+2\pi i(\ldots)) \) in the above formula instead of \( \exp(-2\pi i(\ldots)) \), 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: \( m \geq 1 \).

   **n**
   - Input: the number of columns, \( n \), of the bivariate data sequence.
   - Constraint: \( n \geq 1 \).

   **x[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 \( z_{j_1j_2} \) are denoted by \( x_{j_1j_2} \), for \( j_1 = 0, 1, \ldots, m - 1, j_2 = 0, 1, \ldots, n - 1 \), then the \( mn \) elements of \( x \) must contain the values

   \[ x_{0,0}, x_{0,1}, \ldots, x_{0,n-1}, x_{1,0}, x_{1,1}, \ldots, x_{1,n-1}, \ldots, x_{m-1,0}, x_{m-1,1}, \ldots, x_{m-1,n-1}. \]

   The imaginary parts must be ordered similarly in \( y \).

   **y[m+n]**
   - Output: the real and imaginary parts respectively of the corresponding elements of the computed transform.
nag_fft_2d_complex

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

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
```c
#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("%*[^
");
    while (scanf("%ld%ld", &m, &n)!=EOF)
        if (m*n>=1 && m*n<=MNMAX)
        {
            Vprintf("m = %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("Original data values\n");
            for (j = 0; j<m; ++j)
            {
                Vprintf("Real\n");
                for (i = 0; i<n; ++i)
                    Vprintf("%10.4f%s", x[j*n + i],
                        (i%6==5 && i!=n-1 ? \n " : "));
                Vprintf("Imag\n");
                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("Components of discrete Fourier transforms\n");
            for (j = 0; j<m; ++j)
            {
                Vprintf("Real\n");
                for (i = 0; i<n; ++i)
                    Vprintf("%10.4f%s", x[j*n + i],
                        (i%6==5 && i!=n-1 ? \n " : "));
                Vprintf("Imag\n");
                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");
            for (j = 0; j<m; ++j)
            {
                Vprintf("Real\n");
                for (i = 0; i<n; ++i)
                    Vprintf("%10.4f%s", x[j*n + i],
                        (i%6==5 && i!=n-1 ? \n " : "));
                Vprintf("Imag\n");
                for (i = 0; i<n; ++i)
                    Vprintf("%10.4f%s", y[j*n + i],
                        (i%6==5 && i!=n-1 ? \n " : "));
            }
8.2. Program Data

c06fuc Example Program Data

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Original data values</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
</tbody>
</table>

Components of discrete Fourier transforms

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Components of discrete Fourier transforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
</tbody>
</table>

Original data as restored by inverse transform

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Original data as restored by inverse transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imag</td>
</tr>
</tbody>
</table>