NAG Toolbox

nag_sum_fft_hermitian_2d (c06pw)

1 Purpose

nag_sum_fft_hermitian_2d (c06pw) computes the two-dimensional inverse discrete Fourier transform of a bivariate Hermitian sequence of complex data values.

2 Syntax

```
[x, ifail] = nag_sum_fft_hermitian_2d(m, n, y)
[x, ifail] = c06pw(m, n, y)
```

3 Description

nag_sum_fft_hermitian_2d (c06pw) computes the two-dimensional inverse discrete Fourier transform of a bivariate Hermitian sequence of complex data values z_{j1j2}, for $j1 = 0, 1, \ldots, m-1$ and $j2 = 0, 1, \ldots, n-1$.

The discrete Fourier transform is here defined by

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

where $k_1 = 0, 1, \ldots, m-1$ and $k_2 = 0, 1, \ldots, n-1$. (Note the scale factor of $\frac{1}{\sqrt{mn}}$ in this definition.)

Because the input data satisfies conjugate symmetry (i.e., $z_{k_1 k_2}$ is the complex conjugate of $z_{(m-k_1)k_2}$, the transformed values $\hat{x}_{k_1 k_2}$ are real.

A call of nag_sum_fft_real_2d (c06pv) followed by a call of nag_sum_fft_hermitian_2d (c06pw) will restore the original data.

This function calls nag_sum_fft_realherm_1d_multi_col (c06pq) and nag_sum_fft_complex_1d_multi_row (c06pr) to perform multiple one-dimensional discrete Fourier transforms by the fast Fourier transform (FFT) algorithm in Brigham (1974) and Temperton (1983).

4 References

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

Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350

5 Parameters

5.1 Compulsory Input Parameters

1: **m** – INTEGER

m, the first dimension of the transform.

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

2: **n** – INTEGER

n, the second dimension of the transform.

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

3: $\mathbf{y}((\mathbf{m}/\mathbf{2}+\mathbf{1})\times\mathbf{n})$ – COMPLEX (KIND=nag_wp) array

The Hermitian sequence of complex input dataset z, where z_{j1j2} is stored in $\mathbf{y}(j2\times(m/2+1)+j_1)$, for $j1=0,1,\ldots,m/2$ and $j2=0,1,\ldots,n-1$. That is, if **y** is regarded as a two-dimensional array of dimension $(0:\mathbf{m}/2,0:\mathbf{n}-1)$, then $\mathbf{y}(j_1,j_2)$ must contain $z_{j_1j_2}$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

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

The real output dataset $\hat{x}$, where $\hat{x}_{k1k2}$ is stored in $\mathbf{x}(k2\times m+k1)$, for $k1=0,1,\ldots,m-1$ and $k2=0,1,\ldots,n-1$. That is, if **x** is regarded as a two-dimensional array of dimension $(0:\mathbf{m}-1,0:\mathbf{n}-1)$, then $\mathbf{x}(k_1,k_2)$ contains $\hat{x}_{k_1k_2}$.

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 $=3$

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 forward transform using nag_sum_fft_real_2d (c06pv) and a backward transform using nag_sum_fft_hermitian_2d (c06pw), 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_hermitian_2d (c06pw) is approximately proportional to $mn\log(mn)$, but also depends on the factors of m and n. nag_sum_fft_hermitian_2d (c06pw) is fastest if the only prime factors of m and n are 2, 3 and 5, and is particularly slow if m or n is a large prime, or has large prime factors.

Workspace is internally allocated by nag_sum_fft_hermitian_2d (c06pw). The total size of these arrays is approximately proportional to mn.

9 Example

See Section 10 in nag_sum_fft_real_2d (c06pv).

9.1 Program Text

```
     function c06pw_example

fprintf('c06pw example results\n\n');

m = nag_int(5);
n = nag_int(2);
x = [0.010  0.346;
     1.284  1.960;
     1.754  0.855;
     0.089  0.161;
     1.004  1.844];

% Compute Transform
[y, ifail] = c06pv(m, n, x);
fprintf('\nComponents of discrete Fourier transform\n');
% Display as 2-d array
disp(reshape(y, m/2, n));

% Compute Inverse Transform
[x, ifail] = c06pw(m, n, y);
fprintf('Original sequence as restored by inverse transform\n');
% Display as 2-d array
disp(reshape(x, m, n));
```

9.2 Program Results

```
     c06pw example results

Components of discrete Fourier transform
   2.9431 + 0.0000i  -0.3241 + 0.0000i
  -0.0235 - 0.5576i  -0.4660 - 0.2298i
  -1.1666 + 0.6359i   0.3624 + 0.2615i

Original sequence as restored by inverse transform
    0.0100    0.3460
    1.2840    1.9600
    1.7540    0.8550
    0.0890    0.1610
    1.0040    1.8440
```