

NAG Toolbox

nag_sum_fft_hermitian_3d (c06pz)

1 Purpose

nag_sum_fft_hermitian_3d (c06pz) computes the three-dimensional inverse discrete Fourier transform of a trivariate Hermitian sequence of complex data values.

2 Syntax

```
[x, ifail] = nag_sum_fft_hermitian_3d(n1, n2, n3, y)
[x, ifail] = c06pz(n1, n2, n3, y)
```

3 Description

nag_sum_fft_hermitian_3d (c06pz) computes the three-dimensional inverse discrete Fourier transform of a trivariate Hermitian sequence of complex data values $z_{j_1 j_2 j_3}$, for $j_1 = 0, 1, \dots, n_1 - 1$, $j_2 = 0, 1, \dots, n_2 - 1$ and $j_3 = 0, 1, \dots, n_3 - 1$.

The discrete Fourier transform is here defined by

$$\hat{x}_{k_1 k_2 k_3} = \frac{1}{\sqrt{n_1 n_2 n_3}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} z_{j_1 j_2 j_3} \times \exp\left(2\pi i \left(\frac{j_1 k_1}{n_1} + \frac{j_2 k_2}{n_2} + \frac{j_3 k_3}{n_3}\right)\right),$$

where $k_1 = 0, 1, \dots, n_1 - 1$, $k_2 = 0, 1, \dots, n_2 - 1$ and $k_3 = 0, 1, \dots, n_3 - 1$. (Note the scale factor of $\frac{1}{\sqrt{n_1 n_2 n_3}}$ in this definition.)

Because the input data satisfies conjugate symmetry (i.e., $z_{k_1 k_2 k_3}$ is the complex conjugate of $z_{(n_1-k_1)k_2k_3}$), the transformed values $\hat{x}_{k_1 k_2 k_3}$ are real.

A call of nag_sum_fft_real_3d (c06py) followed by a call of nag_sum_fft_hermitian_3d (c06pz) 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: **n1** – INTEGER

n_1 , the first dimension of the transform.

Constraint: $\mathbf{n1} \geq 1$.

2: **n2** – INTEGER

n_2 , the second dimension of the transform.

Constraint: $\mathbf{n2} \geq 1$.

3: **n3** – INTEGER

n_3 , the third dimension of the transform.

Constraint: $\mathbf{n3} \geq 1$.

4: **y**((**n1/2 + 1**) × **n2** × **n3**) – COMPLEX (KIND=nag_wp) array

The Hermitian sequence of complex input dataset z , where $z_{j_1 j_2 j_3}$ is stored in $\mathbf{y}(j_3 \times (n_1/2 + 1)n_2 + j_2 \times (n_1/2 + 1) + j_1 + 1)$, for $j_1 = 0, 1, \dots, n_1/2$, $j_2 = 0, 1, \dots, n_2 - 1$ and $j_3 = 0, 1, \dots, n_3 - 1$. That is, if \mathbf{y} is regarded as a three-dimensional array of dimension $(0 : \mathbf{n1}/2, 0 : \mathbf{n2} - 1, 0 : \mathbf{n3} - 1)$, then $\mathbf{y}(j_1, j_2, j_3)$ must contain $z_{j_1 j_2 j_3}$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **x**(**n1** × **n2** × **n3**) – REAL (KIND=nag_wp) array

The real output dataset \hat{x} , where $\hat{x}_{k_1 k_2 k_3}$ is stored in $\mathbf{x}(k_3 \times n_1 n_2 + k_2 \times n_1 + k_1 + 1)$, for $k_1 = 0, 1, \dots, n_1 - 1$, $k_2 = 0, 1, \dots, n_2 - 1$ and $k_3 = 0, 1, \dots, n_3 - 1$. That is, if \mathbf{x} is regarded as a three-dimensional array of dimension $(0 : \mathbf{n1} - 1, 0 : \mathbf{n2} - 1, 0 : \mathbf{n3} - 1)$, then $\mathbf{x}(k_1, k_2, k_3)$ contains $\hat{x}_{k_1 k_2 k_3}$.

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: $\mathbf{n1} \geq 1$.

ifail = 2

Constraint: $\mathbf{n2} \geq 1$.

ifail = 3

Constraint: $\mathbf{n3} \geq 1$.

ifail = 4

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_3d (c06py) and a backward transform using nag_sum_fft_hermitian_3d (c06pz), 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_3d (c06pz) is approximately proportional to $n_1 n_2 n_3 \log(n_1 n_2 n_3)$, but also depends on the factors of n_1 , n_2 and n_3 . nag_sum_fft_hermitian_3d (c06pz) is fastest if the only prime factors of n_1 , n_2 and n_3 are 2, 3 and 5, and is particularly slow if one of the dimensions is a large prime, or has large prime factors.

Workspace is internally allocated by nag_sum_fft_hermitian_3d (c06pz). The total size of these arrays is approximately proportional to $n_1 n_2 n_3$.

9 Example

See Section 10 in nag_sum_fft_real_3d (c06py).

9.1 Program Text

```
function c06pz_example

fprintf('c06pz example results\n\n');

n1 = nag_int(3);
n2 = nag_int(3);
n3 = nag_int(4);
x = zeros(n1, n2, n3);
x(1, :, :) = [1.541, 0.161, 1.989, 0.037;
               0.346, 1.907, 0.001, 1.915;
               1.754, 0.042, 1.991, 0.151];

x(2, :, :) = [0.584, 1.004, 1.408, 0.252;
               1.284, 1.137, 0.467, 1.834;
               0.855, 0.725, 1.647, 0.096];

x(3, :, :) = [0.010, 1.844, 0.452, 1.154;
               1.960, 0.240, 1.424, 0.987;
               0.089, 1.660, 0.708, 0.872];
% Compute Transform. Reshape x into a 1-d array
[y, ifail] = c06py(n1, n2, n3, reshape(x,n1*n2*n3,1));

% Display y as a 3-d array
fprintf('\nComponents of discrete Fourier transform:\n');
reshape(y,n1/2,n2,n3)

% Compute Inverse Transform
[x1, ifail] = c06pz(n1, n2, n3, y) ;
fprintf('Original sequence as restored by inverse transform:\n');
reshape(x1, n1, n2, n3)
```

9.2 Program Results

```
c06pz example results

Components of discrete Fourier transform:

ans(:,:,1) =
      5.7547 + 0.0000i  -0.2683 - 0.4203i  -0.2683 + 0.4203i
      0.0814 + 0.0154i   0.0382 + 0.1983i   0.0674 - 0.1220i

ans(:,:,2) =
```

```

-0.2773 - 0.2370i  0.1085 - 0.7560i -0.6882 + 0.2100i
 0.0605 + 0.1563i -0.2752 + 0.2954i  0.2804 + 0.0122i

ans(:,:,3) =
0.4153 + 0.0000i  0.1753 + 0.8712i  0.1753 - 0.8712i
0.6446 - 0.4779i  1.5848 + 0.6163i -0.1134 - 1.5552i

ans(:,:,4) =
-0.2773 + 0.2370i -0.6882 - 0.2100i  0.1085 + 0.7560i
 0.0469 - 0.0772i  0.2005 + 0.0614i -0.1281 - 0.1173i

Original sequence as restored by inverse transform:

ans(:,:,1) =
1.5410    0.3460    1.7540
0.5840    1.2840    0.8550
0.0100    1.9600    0.0890

ans(:,:,2) =
0.1610    1.9070    0.0420
1.0040    1.1370    0.7250
1.8440    0.2400    1.6600

ans(:,:,3) =
1.9890    0.0010    1.9910
1.4080    0.4670    1.6470
0.4520    1.4240    0.7080

ans(:,:,4) =
0.0370    1.9150    0.1510
0.2520    1.8340    0.0960
1.1540    0.9870    0.8720

```
