NAG Toolbox

nag_sum_fft_hermitian_1d_nowork (c06eb)

1 Purpose

nag_sum_fft_hermitian_1d_nowork (c06eb) calculates the discrete Fourier transform of a Hermitian sequence of n complex data values. (No extra workspace required.)

Note: This function is scheduled to be withdrawn, please see c06eb in Advice on Replacement Calls for Withdrawn/Superseded Routines..

2 Syntax

```
[x, ifail] = nag_sum_fft_hermitian_1d_nowork(x, 'n', n)
[x, ifail] = c06eb(x, 'n', n)
```

3 Description

Given a Hermitian sequence of n complex data values z_j (i.e., a sequence such that z_0 is real and z_{n-j} is the complex conjugate of z_j, for $j = 1, 2, \ldots, n-1$), nag_sum_fft_hermitian_1d_nowork (c06eb) calculates their discrete Fourier transform defined by

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \times \exp\left(-i\frac{2\pi jk}{n}\right), \quad k = 0, 1, \ldots, n-1.$$

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{x}_k$ are purely real (see also the C06 Chapter Introduction).

To compute the inverse discrete Fourier transform defined by

$$\hat{y}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \times \exp\left(+i\frac{2\pi jk}{n}\right),$$

this function should be preceded by a call of nag_sum_conjugate_hermitian_rfmt (c06gb) to form the complex conjugates of the z_j.

nag_sum_fft_hermitian_1d_nowork (c06eb) uses the fast Fourier transform (FFT) algorithm (see Brigham (1974)). There are some restrictions on the value of n (see Section 5).

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array

The sequence to be transformed stored in Hermitian form. If the data values z_j are written as $x_j + iy_j$, and if **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_hermitian_1d_nowork (c06eb) is called, then for $0 \leq j \leq n/2$, x_j is contained in $\mathbf{x}(j)$, and for $1 \leq j \leq (n-1)/2$, y_j is contained in $\mathbf{x}(n-j)$. (See also Section 2.1.2 in the C06 Chapter Introduction and Section 10.)

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **x**.

n, the number of data values. The largest prime factor of **n** must not exceed 19, and the total number of prime factors of **n**, counting repetitions, must not exceed 20.

Constraint: **n** > 1.

5.3 Output Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array

The components of the discrete Fourier transform $\hat{x}_k$. If **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_hermitian_1d_nowork (c06eb) is called, then $\hat{x}_k$ is stored in **x**(k), for $k = 0, 1, \ldots, n - 1$.

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 $= 1$

At least one of the prime factors of **n** is greater than 19.

ifail $= 2$

n has more than 20 prime factors.

ifail $= 3$

On entry, **n** ≤ 1.

ifail $= 4$

An unexpected error has occurred in an internal call. Check all function calls and array dimensions. Seek expert help.

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 subsequent inverse transform and comparing the results with the original sequence (in exact arithmetic they would be identical).

8 Further Comments

The time taken is approximately proportional to $n \times \log(n)$, but also depends on the factorization of n. nag_sum_fft_hermitian_1d_nowork (c06eb) is faster if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

On the other hand, nag_sum_fft_hermitian_1d_nowork (c06eb) is particularly slow if n has several unpaired prime factors, i.e., if the 'square-free' part of n has several factors. For such values of n, nag_sum_fft_hermitian_1d_rfmt (c06fb) (which requires an additional n elements of workspace) is considerably faster.

9 Example

This example reads in a sequence of real data values which is assumed to be a Hermitian sequence of complex data values stored in Hermitian form. The input sequence is expanded into a full complex sequence and printed alongside the original sequence. The discrete Fourier transform (as computed by nag_sum_fft_hermitian_1d_nowork (c06eb)) is printed out. It then performs an inverse transform using nag_sum_fft_real_1d_nowork (c06ea) and nag_sum_conjugate_hermitian_rfmt (c06gb), and prints the sequence so obtained alongside the original data values.

9.1 Program Text

```
     function c06eb_example

fprintf('c06eb example results\n\n');

% Hernitian data in compact form
n = 7;
x = [0.34907  0.54890  0.74776  0.94459  1.13850  1.32850  1.51370];

% convert to full complex.
z = nag_herm2complex(x);
disp('Original sequence in full complex form:');
disp(transpose(z));

% transform back to real data
[xt, ifail] = c06eb(x);

disp('Discrete Fourier Transform of x:');
disp(transpose(xt));

% restore by backtransforming to Hermitian data and conjugating
[xr, ifail] = c06ea(xt);
xr(floor(n/2)+2:n) = -xr(floor(n/2)+2:n);
fprintf('Original sequence as restored by inverse transform\n\n');
fprintf('      Original   Restored\n');
for j = 1:n
  fprintf('%3d   %7.4f    %7.4f\n',j, x(j),xr(j));
end

function [z] = nag_herm2complex(x);
  n = size(x,2);
  z(1) = complex(x(1));
  for j = 2:floor((n-1)/2) + 1
    z(j) = x(j) + i*x(n-j+2);
    z(n-j+2) = x(j) - i*x(n-j+2);
  end
  if (mod(n,2)==0)
    z(n/2+1) = complex(x(n/2+1));
  end
```

9.2 Program Results

```
     c06eb example results

Original sequence in full complex form:
   0.3491 + 0.0000i
   0.5489 + 1.5137i
   0.7478 + 1.3285i
   0.9446 + 1.1385i
   0.9446 - 1.1385i
   0.7478 - 1.3285i
   0.5489 - 1.5137i

Discrete Fourier Transform of x:
    1.8262
    1.8686
   -0.0175
    0.5020
   -0.5987
   -0.0314
   -2.6256

Original sequence as restored by inverse transform

       Original   Restored
  1    0.3491     0.3491
  2    0.5489     0.5489
  3    0.7478     0.7478
  4    0.9446     0.9446
  5    1.1385     1.1385
  6    1.3285     1.3285
  7    1.5137     1.5137
```