NAG Toolbox

nag_sum_fft_real_1d_nowork (c06ea)

1 Purpose

nag_sum_fft_real_1d_nowork (c06ea) calculates the discrete Fourier transform of a sequence of n real data values. (No extra workspace required.)

Note: This function is scheduled to be withdrawn, please see c06ea in Advice on Replacement Calls for Withdrawn/Superseded Routines..

2 Syntax

```
[x, ifail] = nag_sum_fft_real_1d_nowork(x, 'n', n)
[x, ifail] = c06ea(x, 'n', n)
```

3 Description

Given a sequence of n real data values x_j, for $j = 0, 1, \ldots, n-1$, nag_sum_fft_real_1d_nowork (c06ea) calculates their discrete Fourier transform defined by

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

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{z}_k$ are complex, but they form a Hermitian sequence (i.e., $\hat{z}_{n-k}$ is the complex conjugate of $\hat{z}_k$), so they are completely determined by n real numbers (see also the C06 Chapter Introduction).

To compute the inverse discrete Fourier transform defined by

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

this function should be followed by a call of nag_sum_conjugate_hermitian_rfmt (c06gb) to form the complex conjugates of the $\hat{z}_k$.

nag_sum_fft_real_1d_nowork (c06ea) 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

x$(j+1)$ must contain x_j, for $j = 0, 1, \ldots, n-1$.

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: $\mathbf{n} > 1$.

5.3 Output Parameters

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

The discrete Fourier transform stored in Hermitian form. If the components of the transform $\hat{z}_k$ are written as $a_k + ib_k$, and if **x** is declared with bounds $(0 : \mathbf{n} - 1)$ in the function from which nag_sum_fft_real_1d_nowork (c06ea) is called, then for $0 \leq k \leq n/2$, a_k is contained in $\mathbf{x}(k)$, and for $1 \leq k \leq (n-1)/2$, b_k is contained in $\mathbf{x}(n-k)$. (See also Section 2.1.2 in the C06 Chapter Introduction and Section 10.)

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, $\mathbf{n} \leq 1$.

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_real_1d_nowork (c06ea) 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_real_1d_nowork (c06ea) 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_real_1d_rfmt (c06fa) (which requires additional double workspace) is considerably faster.

9 Example

This example reads in a sequence of real data values and prints their discrete Fourier transform (as computed by nag_sum_fft_real_1d_nowork (c06ea)), after expanding it from Hermitian form into a full complex sequence. It then performs an inverse transform using nag_sum_conjugate_hermitian_rfmt (c06gb) followed by nag_sum_fft_hermitian_1d_nowork (c06eb), and prints the sequence so obtained alongside the original data values.

9.1 Program Text

```
     function c06ea_example

fprintf('c06ea example results\n\n');

% real data
n = 7;
x = [0.34907  0.54890  0.74776  0.94459  1.13850  1.32850  1.51370];

% transform
[xt, ifail] = c06ea(x);

% get result in form useful for printing.
zt = nag_herm2complex(xt);
disp('Discrete Fourier Transform of x:');
disp(transpose(zt));

% restore by conjugating and backtransforming
xt(floor(n/2)+2:n) = -xt(floor(n/2)+2:n);
[xr, ifail] = c06eb(xt);

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

```
     c06ea example results

Discrete Fourier Transform of x:
   2.4836 + 0.0000i
  -0.2660 + 0.5309i
  -0.2577 + 0.2030i
  -0.2564 + 0.0581i
  -0.2564 - 0.0581i
  -0.2577 - 0.2030i
  -0.2660 - 0.5309i

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
```