NAG Library Function Document

nag_fft_hermitian (c06ebc)

1 Purpose

nag_fft_hermitian (c06ebc) calculates the discrete Fourier transform of a Hermitian sequence of \( n \) complex data values. (No extra workspace required.)

2 Specification

```c
#include <nag.h>
#include <nagc06.h>
void nag_fft_hermitian (double x[], Integer n, NagError *fail)
```

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_fft_hermitian (c06ebc) calculates their discrete Fourier transform defined by

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

(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

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

this function should be preceded by a call of nag_conjugate_hermitian (c06gbc) to form the complex conjugates of the \( z_j \).

nag_fft_hermitian (c06ebc) uses the fast Fourier transform (FFT) algorithm (see Brigham (1974)). There are some restrictions on the value of \( n \) (see Section 5).

4 References


5 Arguments

1. \( x[n] \) – double
   
   **Input/Output**

   *On entry:* 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 \( (0 : n-1) \) in the function from which nag_fft_hermitian (c06ebc) is called, then for \( 0 \leq j \leq n/2 \), \( x_j \) is contained in \( x[j-1] \), and for \( 1 \leq j \leq (n-1)/2 \), \( y_j \) is contained in \( x[n-j] \). (See also Section 2.1.2 in the c06 Chapter Introduction and Section 10.)

   *On exit:* the components of the discrete Fourier transform \( \hat{x}_k \). If \( x \) is declared with bounds \( (0 : n-1) \) in the function from which nag_fft_hermitian (c06ebc) is called, then \( \hat{x}_k \) is stored in \( x[k] \), for \( k = 0, 1, \ldots, n-1 \).
2: n – Integer
   \( \text{Input} \)
   On entry: \( n \), 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: \( n > 1 \).

3: fail – NagError *
   \( \text{Input/Output} \)
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
   On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

NE_INT
   On entry, \( n = \langle \text{value} \rangle \).
   Constraint: \( n > 1 \).

NE_INTERNAL_ERROR
   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.
   An unexpected error has been triggered by this function. Please contact NAG.
   See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
   Your licence key may have expired or may not have been installed correctly.
   See Section 3.6.5 in the Essential Introduction for further information.

NE_PRIME_FACTOR
   At least one of the prime factors of \( n \) is greater than 19. \( n = \langle \text{value} \rangle \).

NE_TOO_MANY_FACTORS
   \( n \) has more than 20 prime factors. \( n = \langle \text{value} \rangle \).

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 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken is approximately proportional to \( n \times \log(n) \), but also depends on the factorization of \( n \).
\( \text{nag_fft_hermitian (c06ebc)} \) 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_fft_hermitian (c06ebc) is particularly slow if \( n \) has several unpaired prime factors, i.e., if the 'square-free' part of \( n \) has several factors.

## 10 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_fft_hermitian (c06ebc)) is printed out. It then performs an inverse transform using nag_fft_real (c06eac) and nag_conjugate_hermitian (c06gbc), and prints the sequence so obtained alongside the original data values.

### 10.1 Program Text

```c
/* nag_fft_hermitian (c06ebc) Example Program.

* Copyright 2014 Numerical Algorithms Group.
* Mark 1, 1990.
*/

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagc06.h>

int main(void)
{
    Integer exit_status = 0, j, n, n2, nj;
    NagError fail;
    double *u = 0, *v = 0, *x = 0, *xx = 0;
    INIT_FAIL(fail);
    printf("nag_fft_hermitian (c06ebc) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n"]);
    #else
    scanf("%*[\n"]);
    #endif
    #ifdef _WIN32
    while (scanf_s("%"NAG_IFMT", &n) != EOF)
    #else
    while (scanf("%"NAG_IFMT", &n) != EOF)
    #endif
    {
        if (n > 1)
            {
                if (!((u = NAG_ALLOC(n, double)) ||
                    (v = NAG_ALLOC(n, double)) ||
                    (x = NAG_ALLOC(n, double)) ||
                    (xx = NAG_ALLOC(n, double)))
                    {
                        printf("Allocation failure\n");
                        exit_status = -1;
                        goto END;
                    }
            }
            else
            {
                printf("Invalid n.\n");
                exit_status = 1;
                return exit_status;
            }
        for (j = 0; j < n; j++)
```
{  
#ifdef _WIN32
    scanf_s("%lf", &x[j]);
#else
    scanf("%lf", &x[j]);
#endif
    xx[j] = x[j];
}  

<!--[if !WIN32]>

<!--[endif]-->  

/* Calculate full complex form of Hermitian sequence */
 u[0] = x[0];
 v[0] = 0.0;
 n2 = (n-1)/2;
 for (j = 1; j <= n2; j++)
 {  
     nj = n - j;
     u[j] = x[j];
     u[nj] = x[j];
     v[j] = x[nj];
     v[nj] = -x[nj];
 }

if (n % 2 == 0)
 {  
     u[n2+1] = x[n2+1];
     v[n2+1] = 0.0;
 }
 printf("Original and corresponding complex sequence\n");
 printf("Data Real Imag \n");
 for (j = 0; j < n; j++)
 printf("%3d %10.5f %10.5f %10.5f\n", j, x[j], u[j], v[j]);

/* Calculate transform */
/* nag_fft_hermitian (c06ebc).
* Single one-dimensional Hermitian discrete Fourier
* transform *
*/
 nag_fft_hermitian(n, x, &fail);
 if (fail.code != NE_NOERROR)
 {  
     printf("Error from nag_fft_hermitian (c06ebc).\n%s\n", fail.message);
     exit_status = 1;
     goto END;
 }
 printf("Components of discrete Fourier transform\n");
 for (j = 0; j < n; j++)
 printf("%3d %10.5f\n", j, x[j]);

/* Calculate inverse transform */
/* nag_fft_real (c06eac).
* Single one-dimensional real discrete Fourier transform
*/
 nag_fft_real(n, x, &fail);
 if (fail.code != NE_NOERROR)
 {  
     printf("Error from nag_fft_real (c06eac).\n%s\n", fail.message);
     exit_status = 1;
     goto END;
 }

/* nag_conjugate_hermitian (c06gbc).
* Complex conjugate of Hermitian sequence *
*/
 nag_conjugate_hermitian(n, x, &fail);
 if (fail.code != NE_NOERROR)
 {  
     printf("Error from nag_conjugate_hermitian (c06gbc).\n%s\n", fail.message);
     exit_status = 1;
     goto END;
 }

printf("Original sequence as restored by inverse transform\n");
 printf("\n Original Restored\n");
 for (j = 0; j < n; j++)
 printf("%10.5f %10.5f\n", x[j], xx[j]);

}
printf("%3"NAG_IFMT" %10.5f %10.5f\n", j, xx[j], x[j]);

END:
NAG_FREE(u);
NAG_FREE(v);
NAG_FREE(x);
NAG_FREE(xx);
return exit_status;
}

10.2 Program Data

nag_fft_hermitian (c06ebc) Example Program Data

<table>
<thead>
<tr>
<th></th>
<th>0.34907</th>
<th>0.54890</th>
<th>0.74776</th>
<th>0.94459</th>
<th>1.13850</th>
<th>1.32850</th>
<th>1.51370</th>
</tr>
</thead>
</table>

10.3 Program Results

nag_fft_hermitian (c06ebc) Example Program Results

Original and corresponding complex sequence

<table>
<thead>
<tr>
<th>Data</th>
<th>Real</th>
<th>Imag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.34907</td>
<td>0.34907</td>
</tr>
<tr>
<td>1</td>
<td>0.54890</td>
<td>0.54890</td>
</tr>
<tr>
<td>2</td>
<td>0.74776</td>
<td>0.74776</td>
</tr>
<tr>
<td>3</td>
<td>0.94459</td>
<td>0.94459</td>
</tr>
<tr>
<td>4</td>
<td>1.13850</td>
<td>0.94459</td>
</tr>
<tr>
<td>5</td>
<td>1.32850</td>
<td>0.74776</td>
</tr>
<tr>
<td>6</td>
<td>1.51370</td>
<td>0.54890</td>
</tr>
</tbody>
</table>

Components of discrete Fourier transform

<table>
<thead>
<tr>
<th></th>
<th>1.82616</th>
<th>1.86862</th>
<th>-0.01750</th>
<th>0.50200</th>
<th>-0.59873</th>
<th>-0.03144</th>
<th>-2.62557</th>
</tr>
</thead>
</table>

Original sequence as restored by inverse transform

<table>
<thead>
<tr>
<th>Original</th>
<th>Restored</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.34907</td>
</tr>
<tr>
<td>1</td>
<td>0.54890</td>
</tr>
<tr>
<td>2</td>
<td>0.74776</td>
</tr>
<tr>
<td>3</td>
<td>0.94459</td>
</tr>
<tr>
<td>4</td>
<td>1.13850</td>
</tr>
<tr>
<td>5</td>
<td>1.32850</td>
</tr>
<tr>
<td>6</td>
<td>1.51370</td>
</tr>
</tbody>
</table>