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

```
#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

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

(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_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

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

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 $(0 : \mathbf{n} - 1)$ 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 $(0 : \mathbf{n} - 1)$ 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 *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: $\mathbf{n} > 1$.

3: **fail** – NagError * *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 $\langle$*value*$\rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle$*value*$\rangle$.
Constraint: $\mathbf{n} > 1$.

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. $\mathbf{n} = \langle$*value*$\rangle$.

NE_TOO_MANY_FACTORS

n has more than 20 prime factors. $\mathbf{n} = \langle$*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. 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

```
/* nag_fft_hermitian (c06ebc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

#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];
        }
      /* 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("\nOriginal and corresponding complex sequence\n");
      printf("\n         Data       Real       Imag \n\n");
      for (j = 0; j < n; j++)
        printf("%3"NAG_IFMT" %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("\nComponents of discrete Fourier transform\n\n");
      for (j = 0; j < n; j++)
        printf("%3"NAG_IFMT" %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("\nOriginal sequence as restored by inverse transform\n");
      printf("\n       Original   Restored\n\n");
      for (j = 0; j < n; 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
    7
  0.34907
  0.54890
  0.74776
  0.94459
  1.13850
  1.32850
  1.51370
```

10.3 Program Results

```
nag_fft_hermitian (c06ebc) Example Program Results

Original and corresponding complex sequence

         Data       Real       Imag

  0    0.34907    0.34907    0.00000
  1    0.54890    0.54890    1.51370
  2    0.74776    0.74776    1.32850
  3    0.94459    0.94459    1.13850
  4    1.13850    0.94459   -1.13850
  5    1.32850    0.74776   -1.32850
  6    1.51370    0.54890   -1.51370

Components of discrete Fourier transform

  0    1.82616
  1    1.86862
  2   -0.01750
  3    0.50200
  4   -0.59873
  5   -0.03144
  6   -2.62557

Original sequence as restored by inverse transform

       Original   Restored

  0    0.34907    0.34907
  1    0.54890    0.54890
  2    0.74776    0.74776
  3    0.94459    0.94459
  4    1.13850    1.13850
  5    1.32850    1.32850
  6    1.51370    1.51370
```