NAG Library Function Document

**nag_fft_real (c06eac)**

1 **Purpose**

nag_fft_real (c06eac) calculates the discrete Fourier transform of a sequence of $n$ real data values.

2 **Specification**

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

3 **Description**

Given a sequence of $n$ real data values $x_j$, for $j = 0, 1, \ldots, n - 1$, nag_fft_real (c06eac) calculates their discrete Fourier transform defined by

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

(Note the scale factor of $\frac{1}{\sqrt{n}}$ in this definition.) The transformed values $\hat{z}_k$ 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.

The function nag_multiple_hermitian_to_complex (c06gsc) may be used to convert a Hermitian sequence to the corresponding complex sequence.

To compute the inverse discrete Fourier transform defined by

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

this function should be followed by a call of nag_conjugate_hermitian (c06gbc) to form the complex conjugates of the $\hat{z}_k$.

nag_fft_real (c06eac) 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:  **n** – Integer

   *Input*

   On entry: $n$, the number of data values.

   Constraint: $n > 1$. 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.

2:  **x[n]** – double

   *Input/Output*

   On entry: $x[j]$ must contain $x_j$, for $j = 0, 1, \ldots, n - 1$. 

Mark 25
On exit: the discrete Fourier transform stored in Hermitian form. If the components of the transform $\tilde{z}_k$ are written as $a_k + ib_k$, then for $0 \leq k \leq n/2$, $a_k$ is contained in $x[k]$, and for $1 \leq k \leq (n-1)/2$, $b_k$ is contained in $x[n - k]$. Elements of the sequence which are not explicitly stored are given by $a_{n-k} = a_k$, $b_{n-k} = -b_k$, $b_0 = 0$ and, if $n$ is even, $b_{n/2} = 0$. (See also Section 10.)

3: \textbf{fail} – NagError * 

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE\_C06\_FACTOR\_GT}

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

\textbf{NE\_C06\_TOO\_MANY\_FACTORS}

$n$ has more than 20 prime factors.

\textbf{NE\_INT\_ARG\_LE}

On entry, $n = \langle value \rangle$.
Constraint: $n > 1$.

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 $n \log(n)$, but also depends on the factorization of $n$. \texttt{nag\_fft\_real} (c06eac) is somewhat faster than average 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, \texttt{nag\_fft\_real} (c06eac) 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 and prints their discrete Fourier transform (as computed by \texttt{nag\_fft\_real} (c06eac)), after expanding it from Hermitian form into a full complex sequence. It then performs an inverse transform using \texttt{nag\_fft\_hermitian} (c06ebc) and \texttt{nag\_conjugate\_hermitian} (c06gbc), and prints the sequence so obtained alongside the original data values.

10.1 Program Text

/* \texttt{nag\_fft\_real} (c06eac) Example Program. */
/* * Copyright 2014 Numerical Algorithms Group. */
#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 *a = 0, *b = 0, *x = 0, *xx = 0;

    INIT_FAIL(fail);

    printf("nag_fft_real (c06eac) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[^
\]");
    #else
        scanf("%*[^
\]");
    #endif
    #ifdef _WIN32
        while (scanf_s("%"NAG_IFMT"", &n) != EOF)
    #else
        while (scanf("%"NAG_IFMT"", &n) != EOF)
    #endif
    {
        if (n > 1)
        {
            if (!(a = NAG_ALLOC(n, double)) ||
                !(b = 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;
            goto END;
        }
        for (j = 0; j < n; j++)
        {
            #ifdef _WIN32
                scanf_s("%lf", &x[j]);
            #else
                scanf("%lf", &x[j]);
            #endif
            xx[j] = x[j];
        }
        /* Calculate 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;
        }
        /* Calculate full complex form of Hermitian result */
        a[0] = x[0];
        b[0] = 0.0;
        n2 = (n-1)/2;
        for (j = 1; j <= n2; j++)
        {
            nj = n - j;
a[j] = x[j];
a[nj] = x[nj];
b[j] = x[nj];
b[nj] = -x[nj];
}

if (n % 2 == 0)
{
a[n2+1] = x[n2+1];
b[n2+1] = 0.0;
}

printf("\nComponents of discrete Fourier transform\n");
printf("\n Real Imag \n\n");
for (j = 0; j < n; j++)
    printf("%3"NAG_IFMT" %10.5f %10.5f\n", j, a[j], b[j]);

/* Calculate inverse transform */
/* 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", fail.message);
    exit_status = 1;
    goto END;
}

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

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(x);
NAG_FREE(xx);

return exit_status;

10.2 Program Data
nag_fft_real (c06eac) Example Program Data
7
0.34907
0.54890
0.74776
0.94459
1.13850
1.32850
1.51370
### 10.3 Program Results

**nag_fft_real (c06eac) Example Program Results**

Components of discrete Fourier transform

<table>
<thead>
<tr>
<th>Real</th>
<th>Imag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.48361</td>
</tr>
<tr>
<td>1</td>
<td>-0.26599</td>
</tr>
<tr>
<td>2</td>
<td>-0.25768</td>
</tr>
<tr>
<td>3</td>
<td>-0.25636</td>
</tr>
<tr>
<td>4</td>
<td>-0.25636</td>
</tr>
<tr>
<td>5</td>
<td>-0.25768</td>
</tr>
<tr>
<td>6</td>
<td>-0.26599</td>
</tr>
</tbody>
</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>