1 Purpose

nag_sum_fft_realherm_1d (c06pac) calculates the discrete Fourier transform of a sequence of $n$ real data values or of a Hermitian sequence of $n$ complex data values stored in compact form in a double array.

2 Specification

```c
#include <nag.h>
#include <nagc06.h>

void nag_sum_fft_realherm_1d (Nag_TransformDirection direct, double x[],
                            Integer n, NagError *fail)
```

3 Description

Given a sequence of $n$ real data values $x_j$, for $j = 0, 1, \ldots, n - 1$, nag_sum_fft_realherm_1d (c06pac) calculates their discrete Fourier transform (in the forward direction) defined by

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

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 (since $\hat{z}_0$ is real, as is $\hat{z}_{n/2}$ for $n$ even).

Alternatively, given a Hermitian sequence of $n$ complex data values $z_j$, this function calculates their inverse (backward) discrete Fourier transform defined by

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

The transformed values $\hat{x}_k$ are real.

(Note the scale factor of $1/\sqrt{n}$ in the above definitions.)

A call of nag_sum_fft_realherm_1d (c06pac) with `direct = Nag_ForwardTransform` followed by a call with `direct = Nag_BackwardTransform` will restore the original data.

nag_sum_fft_realherm_1d (c06pac) uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham (1974)) known as the Stockham self-sorting algorithm, which is described in Temperton (1983).

The same functionality is available using the forward and backward transform function pair: nag_sum_fft_real_2d (c06pvc) and nag_sum_fft_hermitian_2d (c06pwc) on setting $n = 1$. This pair use a different storage solution; real data is stored in a double array, while Hermitian data (the first unconjugated half) is stored in a Complex array.

4 References

5 Arguments

1: `direct` – Nag_TransformDirection

*Input*

*On entry:* if the forward transform as defined in Section 3 is to be computed, then `direct` must be set equal to Nag_FowardTransform.

If the backward transform is to be computed then `direct` must be set equal to Nag_BackwardTransform.

*Constraint:* `direct` = Nag_FowardTransform or Nag_BackwardTransform.

2: `x[n + 2]` – double

*Input/Output*

*On entry:* if `direct` = Nag_FowardTransform, `x[j]` must contain `x_j`, for `j = 0, 1, \ldots, n - 1`;

if `direct` = Nag_BackwardTransform, `x[2 \times k]` and `x[2 \times k + 1]` must contain the real and imaginary parts respectively of `z_k`, for `k = 0, 1, \ldots, n/2`. (Note that for the sequence `z_k` to be Hermitian, the imaginary part of `z_0`, and of `z_{n/2}` for `n` even, must be zero.)

*On exit:* if `direct` = Nag_FowardTransform, `x[2 \times k]` and `x[2 \times k + 1]` will contain the real and imaginary parts respectively of \( \hat{z}_k \), for `k = 0, 1, \ldots, n/2`;

if `direct` = Nag_BackwardTransform, `x[j]` will contain \( \hat{x}_j \), for `j = 0, 1, \ldots, n - 1`.

3: `n` – Integer

*Input*

*On entry:* `n`, the number of data values.

*Constraint:* `n` \( \geq 1 \).

4: `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 `(value)` had an illegal value.

`(value)` is an invalid value of `direct`.

**NE_INT**

On entry, `n = (value)`.

*Constraint:* `n` \( \geq 1 \).

**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.
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

nag_sum_fft_realherm_1d (c06pac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_sum_fft_realherm_1d (c06pac) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken is approximately proportional to $n \times \log(n)$, but also depends on the factorization of $n$. nag_sum_fft_realherm_1d (c06pac) 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. This function internally allocates a workspace of $3n + 100$ double values.

10 Example

This example reads in a sequence of real data values and prints their discrete Fourier transform (as computed by nag_sum_fft_realherm_1d (c06pac) with direct $=$ Nag_FowardTransform), after expanding it from complex Hermitian form into a full complex sequence. It then performs an inverse transform using nag_sum_fft_realherm_1d (c06pac) with direct $=$ Nag_BackwardTransform, and prints the sequence so obtained alongside the original data values.

10.1 Program Text

/* nag_sum_fft_realherm_1d (c06pac) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 24, 2013. */

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

int main(void)
{
    /* Scalars */
    Integer exit_status = 0, i, n;
    /* Arrays */
    double *x = 0, *x_orig, *x_back;
    /* Nag Types */
    NagError fail;
    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker */
    /* to load floating point support */
    */

Mark 25
```c
float force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

INIT_FAIL(fail);

printf("nag_sum_fft_realherm_1d (c06pac) Example Program Results\n");

/* Read dimensions of array and array values from data file. */
#ifdef _WIN32
scanf_s("%*[\n]%"NAG_IFMT"%*[\n]", &n);
#else
scanf("%*[\n]%"NAG_IFMT"%*[\n]", &n);
#endif
#if defined __GNUC__
if (!(x = NAG_ALLOC(n+2, double)) ||
    !(x_orig = NAG_ALLOC(n, double)) ||
    !(x_back = NAG_ALLOC(n+2, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
for (i = 0; i < n; i++) {#ifdef _WIN32
    scanf_s("%lf", &x_orig[i]);
#else
    scanf("%lf", &x_orig[i]);
#endif
    x[i] = x_orig[i];
}
#else
for (i = 0; i < n; i++)
    x[i] = x_orig[i];
#endif

/* Compute discrete Fourier transform of real array x using
 * nag_sum_fft_realherm_1d (c06pac).
 */
nag_sum_fft_realherm_1d(Nag_ForwardTransform, x, n, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sum_fft_realherm_1d (c06pac).\nfail.message\n", exit_status = 1;
    goto END;
}
for (i = 0; i < n + 2; i++) x_back[i] = x[i];

/* Compute inverse discrete Fourier transform of Hermitian array x using
 * nag_sum_fft_realherm_1d (c06pac).
 */
nag_sum_fft_realherm_1d(Nag_BackwardTransform, x_back, n, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sum_fft_realherm_1d (c06pac).\nfail.message\n", exit_status = 2;
    goto END;
}

printf("\n%2s%7s%22s%17s\n","i","x","z = FFT(x)","InvFFT(z)");
for (i = 0; i < n; i++)
    if (i<n/2) {
        printf("%"NAG_IFMT"
            %8.5f (%8.5f, %8.5f ) %8.5f\n", i, x_orig[i],
            x[2*i], x[2*i+1], x_back[i]);
    } else {
        printf("%"NAG_IFMT"%8.5f (%8.5f, %8.5f ) %8.5f\n", i, x_orig[i],
            x[2*(n-i)], -x[2*(n-i)+1], x_back[i]);
    }
END:
NAG_FREE(x);
```

NAG_FREE(x_orig);
NAG_FREE(x_back);
return exit_status;
}

10.2 Program Data

nag_sum_fft_realherm_1d (c06pac) Example Program Data
7
0.34907
0.54890
0.74776
0.94459
1.13850
1.32850
1.51370 : x[0:n-1]

10.3 Program Results

nag_sum_fft_realherm_1d (c06pac) Example Program Results

<table>
<thead>
<tr>
<th>i</th>
<th>x</th>
<th>z = FFT(x)</th>
<th>InvFFT(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.34907</td>
<td>(2.48361, 0.00000)</td>
<td>0.34907</td>
</tr>
<tr>
<td>1</td>
<td>0.54890</td>
<td>(-0.26599, 0.53090)</td>
<td>0.54890</td>
</tr>
<tr>
<td>2</td>
<td>0.74776</td>
<td>(-0.25768, 0.20298)</td>
<td>0.74776</td>
</tr>
<tr>
<td>3</td>
<td>0.94459</td>
<td>(-0.25636, 0.05806)</td>
<td>0.94459</td>
</tr>
<tr>
<td>4</td>
<td>1.13850</td>
<td>(-0.25636, -0.05806)</td>
<td>1.13850</td>
</tr>
<tr>
<td>5</td>
<td>1.32850</td>
<td>(-0.25768, -0.20298)</td>
<td>1.32850</td>
</tr>
<tr>
<td>6</td>
<td>1.51370</td>
<td>(-0.26599, -0.53090)</td>
<td>1.51370</td>
</tr>
</tbody>
</table>