Chapter c06 – Fourier Transforms

1. Scope of the Chapter

This chapter provides C functions for a variety of computations involving Fourier transforms. All the functions use some form of the FFT algorithm.

2. Background

2.1. Complex Transforms

The *discrete Fourier transform* (DFT) of a sequence of n complex numbers $z_0, z_1, \ldots, z_{n-1}$, is defined by:

$$\hat{z}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} z_j \exp\left(-i \frac{2\pi jk}{n}\right) \tag{1}$$

for $k = 0, 1, \ldots, n-1$. Note that equation (1) makes sense for all integer k and with this extension $\hat{z}_k$ is periodic with period n, i.e., $\hat{z}_k = \hat{z}_{k \pm n}$, and in particular $\hat{z}_{-k} = \hat{z}_{n-k}$.

The original data values z_j may conversely be recovered from the transform $\hat{z}_k$ by an *inverse discrete Fourier transform*:

$$z_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \hat{z}_k \exp\left(+i \frac{2\pi jk}{n}\right) \tag{2}$$

for $j = 0, 1, \ldots, n-1$. If we take the complex conjugate of (2), we find that the sequence $\bar{z}_j$ is the DFT of the sequence $\bar{\hat{z}}_k$. Hence the inverse DFT of the sequence $\hat{z}_k$ may be obtained by taking the complex conjugates of the $\hat{z}_k$, performing a DFT, and taking the complex conjugates of the result.

Note: definitions of the discrete Fourier transform vary. Sometimes (2) is used as the definition of the DFT, and (1) as the definition of the inverse. Also the scale-factor of $1/\sqrt{n}$ may be omitted in the definition of the DFT, and replaced by $1/n$ in the definition of the inverse.

If we write $z_j = x_j + iy_j$ and $\hat{z}_k = a_k + ib_k$, then the definition of $\hat{z}_k$ may be written in terms of sines and cosines as:

$$a_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \left(x_j \cos\left(\frac{2\pi jk}{n}\right) + y_j \sin\left(\frac{2\pi jk}{n}\right) \right)$$

$$b_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \left(y_j \cos\left(\frac{2\pi jk}{n}\right) - x_j \sin\left(\frac{2\pi jk}{n}\right) \right)$$

nag_fft_complex (c06ecc) computes a single transform as defined in (1); it may be used in conjunction with nag_conjugate_complex (c06gcc) to compute an inverse transform as defined in (2). See Section 2.4 for functions which compute multiple transforms.

2.2. Real Transforms

If the original sequence is purely real valued, i.e., $z_j = x_j$, then

$$\hat{z}_k = a_k + ib_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \exp\left(-i \frac{2\pi jk}{n}\right) \tag{3}$$

and $\hat{z}_{n-k}$ is the complex conjugate of $\hat{z}_k$. Thus the DFT of a real sequence is a particular type of complex sequence, called a *Hermitian* sequence, or *half-complex* or *conjugate symmetric*, with the properties:

$$a_{n-k} = a_k \quad b_{n-k} = -b_k \quad b_0 = 0$$

and, if n is even, $b_{n/2} = 0$. Thus a Hermitian sequence of n complex data values can be represented by only n, rather than $2n$, independent real values. This fact is exploited by the functions in this chapter, in order to economize in storage and computation time.

The following storage scheme is used in this chapter: the real parts a_k for $0 \leq k \leq n/2$ are stored in normal order in the first $n/2 + 1$ locations of an array x of length n; the corresponding non-zero imaginary parts are stored in reverse order in the remaining locations of x. In other words, the real and imaginary parts of $\hat{z}_k$ are stored as follows:

	if $n = 2s$	if $n = 2s - 1$
x[0]	a_0	a_0
x[1]	a_1	a_1
x[2]	a_2	a_2
$\vdots$	$\vdots$	$\vdots$
x[$s-1$]	a_{s-1}	a_{s-1}
x[s]	a_s	b_{s-1}
x[$s+1$]	b_{s-1}	b_{s-2}
$\vdots$	$\vdots$	$\vdots$
x[$n-2$]	b_2	b_2
x[$n-1$]	b_1	b_1

nag_multiple_hermitian_to_complex (c06gsc) can be used to convert one or more Hermitian sequences from this compact storage scheme to a representation which uses two arrays of length n to store the real and imaginary parts of the full complex sequence.

From the definition of the inverse transform, we can deduce:

$$x_j = \frac{1}{\sqrt{n}}\left(a_0 + 2\sum_{k=0}^{n/2-1}\left(a_k\cos\left(\frac{2\pi jk}{n}\right) - b_k\sin\left(\frac{2\pi jk}{n}\right)\right) + a_{n/2}\right)$$

where $a_{n/2} = 0$ if n is odd.

nag_fft_real (c06eac) computes a single transform of a real sequence as defined in (3); nag_fft_hermitian (c06ebc) computes an analogous transform of a Hermitian sequence, for which the result is a purely real sequence. These functions can be used in conjunction with nag_conjugate_hermitian (c06gbc) to compute inverse transforms derived from (2). See Section 2.4 for functions which compute multiple transforms.

2.3. Real Symmetric Transforms

In many applications the sequence x_j will not only be real, but may also possess additional symmetries which we may exploit to reduce further the computing time and storage requirements.

For example, if the sequence x_j is *odd* ($x_j = -x_{n-j}$), then the discrete Fourier transform of x_j contains only sine terms. Rather than compute the transform of an odd sequence, we define the *sine transform* of a real sequence by

$$\hat{x}_k = \sqrt{\frac{2}{n}}\sum_{j=1}^{n-1} x_j \sin\left(\frac{\pi jk}{n}\right)$$

which could have been computed using the Fourier transform of a real odd sequence of length $2n$. In this case the x_j are arbitrary, and the symmetry only becomes apparent when the sequence is extended.

Similarly we define the *cosine transform* of a real sequence by

$$\hat{x}_k = \sqrt{\frac{2}{n}}\left(\frac{1}{2}x_0 + \sum_{j=1}^{n-1} x_j\cos\left(\frac{\pi jk}{n}\right) + \frac{1}{2}(-1)^k x_n\right)$$

which could have been computed using the Fourier transform of a real even sequence of length $2n$.

In addition to these 'half-wave' symmetries described above, sequences arise in practice with 'quarter-wave' symmetries. We define the *quarter-wave sine transform* by

$$\hat{x}_k = \frac{1}{\sqrt{n}} \left(\sum_{j=1}^{n-1} x_j \sin\left(\frac{\pi j(2k-1)}{2n}\right) + \frac{1}{2}(-1)^{k-1} x_n \right)$$

which could have been computed using the Fourier transform of a real sequence of length $4n$ of the form

$$(0, x_1, \ldots, x_{n-1}, x_n, x_{n-1}, \ldots, x_1, 0, -x_1, \ldots, -x_{n-1}, -x_n, -x_{n-1}, \ldots, -x_1).$$

Similarly we may define the *quarter-wave cosine transform* by

$$\hat{x}_k = \frac{1}{\sqrt{n}} \left(\frac{1}{2} x_0 + \sum_{j=1}^{n-1} x_j \cos\left(\frac{\pi j(2k-1)}{2n}\right) \right)$$

which could have been computed using the Fourier transform of a real sequence of length $4n$ of the form

$$(x_0, x_1, \ldots, x_{n-1}, 0, -x_{n-1}, \ldots, -x_1, -x_0, -x_1, \ldots, -x_{n-1}, 0, x_{n-1}, \ldots, x_1).$$

2.4. Multiple Transforms

In many applications one needs to compute Fourier transforms of several sequences, all of the same length. NAG C Library functions are provided for this type of application:

nag_fft_multiple_complex (c06frc) for complex transforms
nag_fft_multiple_real (c06fpc) for real-to-Hermitian transforms
nag_fft_multiple_hermitian (c06fqc) for Hermitian-to-real transforms
nag_fft_multiple_sine (c06hac) for sine transforms
nag_fft_multiple_cosine (c06hbc) for cosine transforms
nag_fft_multiple_qtr_sine (c06hcc) for quarter-wave sine transforms
nag_fft_multiple_qtr_cosine (c06hdc) for quarter-wave cosine transforms

These functions must all be preceded by a call to nag_fft_init_trig (c06gzc) to compute the trigonometric function values which will be needed. The functions for multiple transforms are likely to be faster than making repeated calls to the functions for single transforms, but they require more workspace.

2.5. Two-dimensional Transforms

nag_fft_2d_complex (c06fuc) computes the two-dimensional generalization of (1) defined by:

$$\hat{z}_{k_1 k_2} = \frac{1}{\sqrt{n_1 n_2}} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} z_{j_1 j_2} \exp\left(-2\pi i \left(\frac{j_1 k_1}{n_1} + \frac{j_2 k_2}{n_2}\right)\right)$$

for $k_1 = 0, 1, \ldots, n_1 - 1; k_2 = 0, 1, \ldots, n_2 - 1$.

2.6. Convolutions and Correlations

One of the most important applications of the discrete Fourier transform is to the efficient computation of the discrete *convolution* or *correlation* of two vectors x and y defined by:

$$\text{convolution}: \; z_k = \sum_{j=0}^{n-1} \bar{x}_j y_{k-j}$$

$$\text{correlation}: \; w_k = \sum_{j=0}^{n-1} \bar{x}_j y_{k+j}$$

Here x and y are assumed to be periodic with period n. These computations are performed by nag_convolution_real (c06ekc), for real vectors x and y.

3. Available Functions

3.1. Fourier Transforms

Single complex transform	`c06ecc`
Single real-to-Hermitian transform	`c06eac`
Single Hermitian-to-real transform	`c06ebc`
Multiple complex transforms	`c06frc`
Multiple real-to-Hermitian transforms	`c06fpc`
Multiple Hermitian-to-real transforms	`c06fqc`
Multiple cosine transforms	`c06hbc`
Multiple sine transforms	`c06hac`
Multiple quarter-wave cosine transforms	`c06hdc`
Multiple quarter-wave sine transforms	`c06hcc`
Two-dimensional complex transform	`c06fuc`

3.2. Convolution and Correlation

Convolution or correlation of two real vectors	`c06ekc`

3.3. Utility Routines

Form conjugate of single Hermitian sequence	`c06gbc`
Form conjugate of multiple Hermitian sequences	`c06gqc`
Form conjugate of single or multiple complex sequences	`c06gcc`
Convert single or multiple Hermitian sequences to complex	`c06gsc`
Compute trigonometric function values	`c06gzc`