So far we have been looking at signals as a function of time or an index in time. Just like continuous-time signals, we can view a time signal as one that consists of a range of frequencies. A signal can be observed as a trace on an oscilloscope or we can observe it through a spectrum analyzer, which displays the strength of the frequency components that make up that signal. Analytically, Fourier analysis provides us with the connection between the time-domain and frequency-domain view of the signal. It tells us that provided some conditions are satisfied; the two views are equivalent. Thus sometimes it is more convenient to describe a signal in the time-domain whereas the frequency-domain description will be more effective in other circumstances.

As we shall discover in this chapter, the frequency-domain (Fourier) representation not only gives us an alternative view of discrete-time signals, it also provides us with a way to compute certain time-domain operations like convolution and correlation more efficiently. Tools for Fourier analysis consist of the discrete Fourier series (DFS) for periodic signals and the discrete Fourier transform (DFT) for aperiodic signals. Transforming signals from the time to the frequency domain through the DFT is computationally expensive. In the early 1960s, Cooley and Tukey discovered an efficient algorithm for the computation of DFTs, called the fast Fourier transform (FFT). This discovery made real-time computation of convolution and filtering a reality. Some would say that this is when digital signal processing as a discipline was established. Since then a large variety of similar algorithms were proposed and studied. Some of them improve on the original algorithm while others tackle situations that the FFT is not designed for. Today, all DSP processors are able to compute the FFT in software sufficiently fast for most but the most demanding applications. In those situations, specific VLSI devices are available commercially.
4.1 Discrete Fourier series for discrete-time periodic signals

Any periodic discrete-time signal, with a period of \( N \), can be expressed as a linear combination of \( N \) complex exponential functions.

\[
x(n) = \sum_{k=0}^{N-1} c_k e^{j2\pi kn/N} \tag{1}
\]

where

\[
j = \sqrt{-1}
\]

and

\[
e^{j\theta} = \cos \theta + j \sin \theta
\]

Equation 1 is called the discrete-time Fourier series (DTFS).

Given the signal \( x(n) \), the Fourier coefficients can be calculated by

\[
c_k = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{2}
\]

Note that these Fourier coefficients are generally complex-valued. They provide a description of the signal in the frequency domain. The coefficient \( c_k \) has a magnitude and phase associated with a (normalized) frequency given by

\[
\omega_k = \frac{2\pi k}{N}
\]

These Fourier coefficients form the (discrete) frequency spectrum of the signal. This normalized frequency can be de-normalized if we know the sampling frequency (\( F_s \)), or

\[
\omega = \frac{2\pi}{T}
\]

the time lapse (\( T \)) between two samples which are related by

Since

\[
0 \leq \omega_k \leq 2\pi
\]

The denormalized frequency \( \omega \) takes on values in the range

\[
0 \leq \omega \leq \omega_q
\]

It is easy to verify that the sequence of coefficients given by equation (2) is periodic with a period of \( N \). This means that the frequency spectrum of a periodic signal is also periodic.

Example: 4.1

Determine the discrete spectrum of a periodic sequence \( x(n) \) with a period \( N=4 \) given by

\[
x(n) = \{0, \ 1, \ 1, \ 0\} \quad n = 0,1,2,3
\]

Solution:

From equation (2), we have

\[
c_k = \frac{1}{4} \sum_{n=0}^{3} x(n) e^{-j2\pi kn/4} \quad k = 0,\ldots,3
\]

\[
= \frac{1}{4} \left[ x(1)e^{-j\pi k/2} + x(2)e^{-j\pi k} \right]
\]
and

\[ c_0 = \frac{1}{4} [1 + 1] = \frac{1}{2} \]
\[ c_1 = \frac{1}{4} \left[ e^{-j\pi/2} + e^{-j\pi} \right] = \frac{1}{4} (-1 - j) \]
\[ c_2 = \frac{1}{4} \left[ e^{-j\pi} + e^{-j2\pi} \right] = 0 \]
\[ c_3 = \frac{1}{4} \left[ e^{-j3\pi/2} + e^{-j3\pi} \right] = \frac{1}{4} (-1 + j) \]

The magnitude of this discrete spectrum is shown in Figure 4.1.

\[ |X(k)|^2 \]

Figure 4.1
The discrete magnitude squared spectrum of the signal in Example 4.1.

4.2 Discrete Fourier transform for discrete-time aperiodic signals

When a discrete-time signal or sequence is non-periodic (or aperiodic), we cannot use the discrete Fourier series to represent it. Instead, the discrete Fourier transform (DFT) has to be used for representing the signal in the frequency domain. The DFT is the discrete-time equivalent of the (continuous-time) Fourier transforms. As with the discrete Fourier series, the DFT produces a set of coefficients, which are sampled values of the frequency spectrum at regular intervals. The number of samples obtained depends on the number of samples in the time sequence.

A time sequence \( x(n) \) is transformed into a sequence \( X(\omega) \) by the discrete Fourier transform.

\[ X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \quad k = 0, 1, ..., N - 1 \] (3)
This formula defines an \( N \)-point DFT. The sequence \( X(k) \) are sampled values of the continuous frequency spectrum of \( x(n) \). For the sake of convenience, equation 3 is usually written in the form

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \quad k = 0,1,\ldots,N-1
\]  

where

\[
W = e^{-j2\pi/N}
\]

Note that, in general, the computation of each coefficient \( X(k) \) requires a complex summation of \( N \) complex multiplications.

Since there are \( N \) coefficients to be computed for each DFT, a total of \( N^2 \) complex additions and \( N^2 \) complex multiplications are needed. Even for moderate values of \( N \), say 32; the computational burden is still very heavy. Fortunately, more efficient algorithms than direct computation are available. They are generally classified, as fast Fourier transform algorithms and some typical ones will be described later in the chapter.

### 4.3 The inverse discrete Fourier transform and its computation

The inverse discrete Fourier transform (IDFT) converts the sequence of discrete Fourier coefficients back to the time sequence and is defined as

\[
x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N} = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-kn} \quad k = 0,1,\ldots,N-1
\]  

The formulas for the forward and inverse transforms are identical except for the scaling factor of \( 1/N \) and the negation of the power in the exponential term (\( W_N^{-kn} \) instead of \( W_N^{kn} \)). So any fast algorithm that exists for the forward transform can easily be applied to the inverse transform. In the same way, specific hardware designed for the forward transform can also perform the inverse transform.

### 4.4 Properties of the DFT

In this section, some of the important properties of the DFT are summarized. Let

\[
x(n) \leftrightarrow X(k)
\]

denote the DFT between \( x(n) \) and \( X(k) \) in the discussions below.

#### 4.4.1 Periodicity of the DFT

Consider \( X(k+N) \) which is given by

\[
X(k+N) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}e^{-j2\pi Nn/N} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} = X(k)
\]
This implies that $X(k)$ is periodic with a period of $N$, even though $x(n)$ is aperiodic. Thus for a discrete-time signal, we cannot obtain the full spectrum (frequencies from negative infinity to infinity).

### 4.4.2 Linearity

If both $x_1(n)$ and $x_2(n)$ are of the same length and

$$x_1(n) \leftrightarrow X_1(k) \quad \text{and} \quad x_2(n) \leftrightarrow X_2(k)$$

then

$$x(n) = ax_1(n) + bx_2(n) \leftrightarrow X(k) = aX_1(k) + bX_2(k)$$

where $a$ and $b$ are arbitrary constants.

### 4.4.3 Parseval’s relation

This relationship states that the energy of the signal can be calculated from the time sequence or from the Fourier coefficients.

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2$$

### 4.4.4 Real sequences

If $x(n)$ is real, then

$$x(n) = \hat{x}^*(n) \quad \text{and} \quad X(k) = \hat{X}^*(-k)$$

i.e. the real part of $X(k)$ is an even function or is symmetrical about $k = 0$ and the imaginary part is odd function. This kind of symmetry in $X(k)$ is also known as hermitian symmetry.

### 4.4.5 Even and odd functions

If $x(n)$ is an even function or has even symmetry about $n = 0$, i.e. $x(n) = x(-n)$, then $X(k)$ will also be an even function. If $x(n)$ is an odd function, i.e. $x(n) = \hat{x}(n)$, then $X(k)$ will also be an odd function.

Furthermore, if $x(n)$ is real and even, then $X(k)$ is real and even. If $x(n)$ is real and odd, then $X(k)$ is imaginary and odd.

### 4.4.6 Convolution

$$x(n) = x_1(n) * x_2(n) \leftrightarrow X(k) = X_1(k)X_2(k)$$

Convolution in the time domain becomes a point-by-point multiplication in the frequency domain. Thus a convolution operation can be performed by first performing the DFT of each time sequence, obtain the product of the DFTs, and then inverse transform the result back to a time sequence. This is called circular convolution, which is somewhat different from the linear convolution discussed in the previous chapter. The computation of linear convolution using DFT is explored in more detail in a later section.

$$r_{12}(n) = \sum_{k=-\infty}^{\infty} x_1(k)x_2(k-n) \leftrightarrow S_{12}(k) = X_1(k)X_2(-k)$$
4.4.7 Correlation

$S_{12}(k)$ is called the cross-energy density spectrum. In the case where $x_1(n) = x_2(n)$, it is called the power spectrum. In other words, the power spectrum of a signal is the DFT of its autocorrelation sequence.

4.4.7 Time delay

$$x(n - m) \leftrightarrow W^{im} X(k)$$

A delay in the time domain is equivalent to a multiplication by a complex exponential function in the frequency domain.

4.4.9 Frequency shifting

$$W^{-h} x(n) \leftrightarrow X(k - 1)$$

This is the frequency domain equivalent of the previous property. A shift (or delay) in the frequency sequence is equivalent to the multiplication of the time sequence by a complex exponential function. Alternatively, we can say that if the time sequence is multiplied by a complex exponential, then it is equivalent to a shift in the frequency spectrum.

$$x(n) \cos(2\pi ln / N) \leftrightarrow \frac{1}{2} X(k + l) + \frac{1}{2} X(k - l)$$

4.4.10 Modulation

In the above frequency shifting property, if the complex exponential function is replaced by a real sinusoidal function, then the multiplication in the time domain is equivalent to shifting the half the power of the spectrum up and half of it down by the same amount. Multiplication by a sinusoid is the modulation operation performed in communication systems. This is illustrated in Figure 4.2.

[Diagram showing the effect of modulation on the discrete frequency spectrum]
4.4.11 **Differentiation in the frequency domain**

\[
 nx(n) \leftrightarrow \frac{dX(\omega)}{d(\omega)}
\]

Differentiation in the frequency domain is related to the multiplication of the time signal by a ramp. This property is useful in the computation of the group delay of digital filters. Details can be found in the chapter on FIR filters.

4.5 **The fast Fourier transform**

The DFT is computationally expensive. In general, an \(N\)-point DFT requires \(N\) complex multiplications and \(N-1\) complex additions. If \(N=2^r\), where \(r\) is a positive integer, then an efficient algorithm called the fast Fourier transform (FFT) can be used to compute the DFT. Cooley and Tukey first developed this algorithm in the 1960s.

The basic idea of the FFT is to rewrite the DFT equation into two parts:

\[
 X(k) = \sum_{n=0}^{N/2-1} x(2n)W_N^{2nk} + W^k \sum_{n=0}^{N/2-1} x(2n+1)W_N^{2nk} \quad k = 0,1,..., N-1
\]

\[
 = X_1(k) + W^k X_2(k)
\]

The first part,

\[
 X_1(k) = \sum_{n=0}^{(N/2)-1} x(2n)W_N^{2kn}
\]

\[
 = \sum_{n=0}^{(N/2)-1} x(2n)W_N^{kn}
\]

is the DFT of the even sequence and the second part, \(X_2(k)\), is the DFT of the odd sequence. Notice that the factor \(W_N^{2nk}\) appears in both DFTs and need only be computed once. The FFT coefficients are obtained by combining the DFTs of the odd and even sequences using the formulas:

\[
 X(k) = X_1(k) + W^k X_2(k) \quad \text{for} \quad k = 0,1,..., \frac{N}{2}-1
\]

\[
 X\left(k + \frac{N}{2}\right) = X_1(k) - W^k X_2(k) \quad \text{for} \quad k = 0,1,..., \frac{N}{2}-1
\]

The complex factor \(W_N^k\) is known as the twiddle factor.

These subsequences can be further broken down into even and odd sequences until only 2-point DFTs are left. So each \(N/2\)-point DFT is obtained by combining two \(N/4\)-point DFTs, each of that is obtained by combining two \(N/8\)-point DFTs, etc. There are a total of \(r\) stages since \(N=2^r\).
Computation of two-point DFTs is trivial. The basic operation is illustrated in Figure 4.3.

\[
a = x_0 + jy_0 \\
b = x_1 + jy_1 \\
x_0' + jy_0' = a + W^k b \\
x_1' + jy_1' = a - W^k b
\]

**Figure 4.3**
The butterfly operation of the decimation-in-time FFT

It is usually known as a butterfly operation. Figure 4.4 illustrates the three stages required in the computation of an 8-point FFT. The twiddle factors are usually pre-computed and stored in memory.

**Figure 4.4**
The three stages in an 8-point decimation-in-time FFT
Note that if we want the DFT coefficients to come out at the natural order, the input sequence has to be rearranged. This reordering is known as bit reversal. We can see why it is called bit reversal if we represent the index sequence in binary form. The following table illustrates this.

<table>
<thead>
<tr>
<th>Natural order</th>
<th>Binary form</th>
<th>Bit reversed</th>
<th>Reordered index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

Compare the bit-reversed indices with that of Figure 4.4.

An efficient algorithm for bit reversal will be discussed later.

This FFT algorithm is also referred to as the radix-2 decimation-in-time FFT. Radix 2 refers to the fact that 2-point DFTs are the basic computational block in this algorithm. Decimation-in-time refers to the breaking up (decimation) of the data sequence into even and odd sequences. This is in contrast to decimation-in-frequency a little later.

### 4.5.1 Computational savings

An $N$-point FFT consists of $N/2$ butterflies per stage with $\log_2 N$ stages. Each butterfly has one complex multiplication and two complex additions. Thus there are a total of $(N/2)\log_2 N$ complex multiplications compared with $N^2$ for DFT, and $N \log_2 N$ complex additions compared with $N(N-1)$ for the DFT. A substantial saving when $N$ is large.

### 4.5.2 Decimation-in-frequency algorithm

Partitioning the data sequence into two halves, instead of odd and even sequences can derive another radix 2 FFT algorithm. Thus,

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)W_N^{kn} + \sum_{n=N/2}^{N-1} x(n)W_N^{kn}$$

$$= \sum_{n=0}^{(N/2)-1} x(n)W_N^{kn} + W_N^{Nk/2} \sum_{n=0}^{N/2-1} x(n + \frac{N}{2})W_N^{kn}$$

$$= \sum_{n=0}^{(N/2)-1} \left[ x(n) + (-1)^k x \left( n + \frac{N}{2} \right) \right] W_N^{kn}$$

The FFT coefficient sequence can be broken up into even and odd sequences and they have the form

$$X(2k) = \sum_{n=0}^{(N/2)-1} g_1(n)W_{N/2}^{kn}$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} g_2(n)W_{N/2}^{kn}$$