6

Continuous, Discrete, and Fast Fourier Transform

6.1 INTRODUCTION

In this chapter, the Fourier transform is related to the complex Fourier series presented in the previous chapter (see overview, Fig. 5.3). The Fourier transform in continuous time (or space) is referred to as the continuous Fourier transform (CFT). In Section 6.3, we develop a discrete version of the Fourier transform (DFT) and explore an efficient algorithm for calculating it this efficient algorithm is known as the fast Fourier transform (FFT). In Chapter 7, we show an example of the use of the CFT in the reconstruction of images and an application of the DFT calculating the power spectra of simulated and recorded signals.

6.2 THE FOURIER TRANSFORM

The Fourier transform is an operation that transforms data from the time (or spatial) domain into the frequency domain. In this section we demonstrate that the transform can be considered as the limiting case of the complex Fourier series.

Figure 6.1 illustrates how one can create an arbitrary function $f(t)$ from a periodic signal $f_{T_0}(t)$ by increasing its period T_0 to ∞. This thought process is the basis on which the continuous Fourier transform is derived from the complex Fourier series. For clarity, we reiterate the expressions for the complex series and its coefficients derived in Chapter 5:

$$P(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega t} \tag{6.1a}$$

with the coefficients defined as

$$c_n = \frac{1}{T} \int_T f(t) e^{-jn\omega t} dt \tag{6.1b}$$

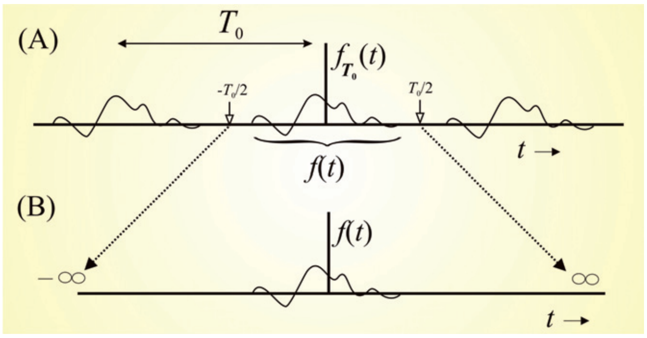


Figure 6.1 (A) Periodic function $f_{T_0}(t)$, and (B) function $f(t)$ derived from one cycle.

The first step is to establish the coefficient c_n for the series $f_{T_0}(t)$ in Figure 6.1A. Hereby we integrate over ***one cycle*** of the function $f_{T_0}(t)$, which equals $f(t)$ over the period from $-T_0/2$ to $T_0/2$:

$$c_n = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} f(t) e^{-jn\omega_0 t} dt \tag{6.2}$$

In Equation (6.2), we include the period T_0, and the fundamental angular frequency ω_0. The relationship between these parameters is given by $T_0 = \frac{1}{f_0} = \frac{2\pi}{\omega_0}$ (with f_0 the fundamental frequency in Hz). So far we are still simply applying the complex Fourier series to $f(t)$ as one cycle of $f_{T_0}(t)$. The second step is to stretch the period parameter T_0 to ∞ (which also means that the fundamental angular frequency $\omega_0 \rightarrow 0$) and to define c_n^* as

$$c_n^* = \lim_{\substack{T_0 \rightarrow \infty \\ \omega_0 \rightarrow 0}} \int_{-T_0/2}^{T_0/2} f(t) e^{-jn\omega_0 t} dt \tag{6.3}$$

The coefficient c_n^* in Equation (6.3) is very similar to c_n in Equation (6.2), but ***note that we smuggled a 1/T_0 factor out of the expression***! Stated a bit more formally, we say that c_n^* is defined by Equation (6.3), thereby avoiding a division by $T_0 \rightarrow \infty$. Because $\omega_0 \rightarrow 0$, we may consider the product $n\omega_0$ a continuous scale of the angular frequency ω(i.e., $\lim_{\omega_0 \rightarrow 0} n\omega_0 = \omega$). Further, representing the complex coefficients c_n^* in a function $F(j\omega)$, we may rewrite Equation (6.3) as

$$\boxed{F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt} \tag{6.4}$$

Equation (6.4) is the definition of the continuous time Fourier transform (CFT). In some texts, $F(\omega)$ is used instead of $F(j\omega)$. Another common notation substitutes $\omega = 2\pi f$ resulting in $F(f) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ft}\,dt$, which simply represents the results in units of Hz. Given Equation (6.4), we will now show that the inverse transform also follows from the complex Fourier series. Combining Equations (6.1b) and (6.4) and using $\omega = n\omega_0$, we have

$$c_n = \frac{1}{T_0} F(jn\omega_0) \tag{6.5}$$

Note that the $1/T_0$ factor is reintroduced. The $1/T_0$ correction maintains the consistency of the transform with its inverse.

Using Equation (6.1a) for the complex Fourier series,

$$\begin{aligned} f_{T_0}(t) &= \sum_{n=-\infty}^{\infty} \frac{1}{T_0} F(jn\omega_0)e^{jn\omega_0 t}, \text{ using } \frac{1}{T_0} = \frac{\omega_0}{2\pi} \\ \rightarrow f_{T_0}(t) &= \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} F(jn\omega_0)e^{jn\omega_0 t}\omega_0 \end{aligned} \tag{6.6}$$

Now we let $T_0 \rightarrow \infty$, meaning that $\omega_0 \rightarrow 0$. If we change the notation $\omega_0 = \Delta\omega$ to use as a limiting variable, Equation (6.6) becomes

$$f(t) = \lim_{\substack{T_0 \to \infty \\ \omega_0 \to 0}} f_{T_0}(t) = \lim_{\Delta\omega \to 0} \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} F(jn\Delta\omega)e^{jn\Delta\omega t}\Delta\omega \tag{6.7}$$

We can interpret the sum in Equation (6.7) as calculating the area under the continuous function $F(j\omega)e^{j\omega t}$ using arbitrarily narrow slices in the limit. This interpretation generates the result for the inverse Fourier transform:

$$\boxed{f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}\,d\omega} \tag{6.8}$$

If $\omega = 2\pi f$ is substituted in Equation (6.8), we get $f(t) = \int_{-\infty}^{\infty} F(f)e^{j2\pi ft}\,df$. Equations (6.4) and (6.8) form a consistent pair for the Fourier transform and the inverse Fourier transform, respectively. With the exception of the $1/T_0$ factor, there is a direct correspondence between the transform and the complex series, whereby the transform can be considered as a series in the limit where $T_0 \rightarrow \infty$. Both the equations for the transform and its

Table 6.1 Examples of Fourier Transform Pairs

Time/spatial domain $f(t)$	Frequency domain $F(\omega)$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$\cos(\omega_0 t)$	$\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)]$
$\sin(\omega_0 t)$	$j\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]$

inverse apply for continuous time or space. Therefore, this flavor of spectral analysis is called the continuous Fourier transform (CFT).

6.2.1 Examples of CFT Pairs

Equations (6.4) and (6.8) can be used to establish Fourier transform pairs; for complicated functions, it is common practice to use tables (e.g., Table 6.1) that summarize the pairs. Here we focus on a few simple examples and associated interpretations relevant for signal analysis. First we consider the signal $\delta(t)$, known as the Dirac delta function; its Fourier transform is given by

$$F(j\omega)=\int_{-\infty}^{\infty}\delta(t)e^{-j\omega t}\,dt=e^{-j\omega 0}=1 \tag{6.9}$$

This integral is evaluated using the sifting property of the delta function (Equation (2.8)). From a signal processing standpoint, it is interesting to see that ***the time domain Dirac delta function corresponds to all frequencies in the frequency domain***.

We noticed when discussing the transforms that the equations for the Fourier transform and its inverse are fairly similar. Repeating Equations (6.4) and (6.8),

$$F(j\omega)=\int_{-\infty}^{\infty}f(t)e^{-j\omega t}\,dt \quad\Leftrightarrow\quad f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(jw)e^{j\omega t}\,d\omega \tag{6.10}$$

Clearly the transform and its inverse are identical with the exception of the $1/2\pi$ scaling factor and the sign of the imaginary exponent. This means that one can derive the inverse transform from the transform and vice versa by correcting for the scaling and the sign of the variable over which one integrates. This is the so-called ***duality property***, which results in some interesting and useful parallels between time and frequency domain representations.

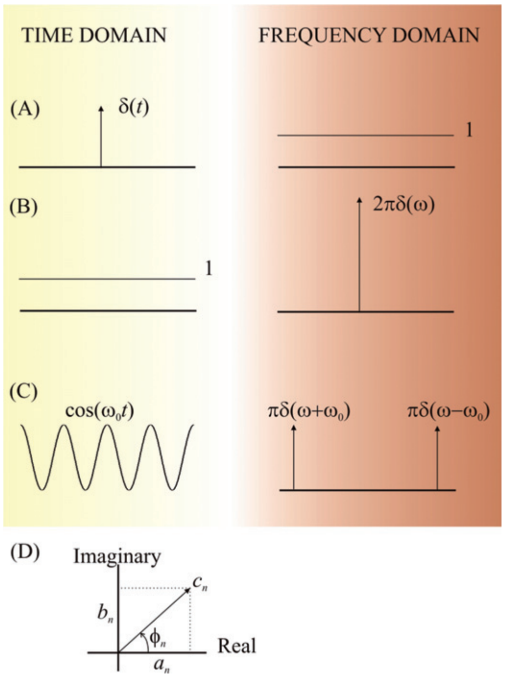


Figure 6.2 Common Fourier transform pairs. (A) A Dirac impulse function in the time domain is represented by all frequencies in the frequency domain. (B) This relationship can be reversed to show that a DC component in the time domain generates an impulse function at a frequency of zero. (C) A pure (cosine) wave shows peaks at $\pm\omega_0$ in the frequency domain. (D) In general, a coefficient c_n, being part of $F(j\omega)$, may contain both real (a_n) and imaginary (b_n) parts (represented here in a polar plot).

Applying this Fourier transform and inverse transform relationship to the Dirac impulse $\delta(t)$, one can conclude that the time domain equivalent for a delta function in the frequency domain $\delta(-\omega)$ must be the constant function $f(t) = 1/2\pi$. Because the scaling is a constant (not depending on ω) and $\delta(-\omega) = \delta(\omega)$, one can say that $1 \Leftrightarrow 2\pi\delta(\omega)$ forms a time domain–frequency domain pair; or in a different notation,

$$F(jw) = 2\pi\delta(\omega) = \int_{-\infty}^{\infty} 1e^{-j\omega t}\, dt \tag{6.11}$$

This outcome can be validated by evaluating the inverse Fourier transform $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}[2\pi\delta(\omega)]\, e^{j\omega t}\, d\omega$. Using the sifting property of $\delta(\omega)$, it can be seen that this expression evaluates to 1. Interpreting this property from a signal processing point of view, this result indicates that an additive constant (1 in this case), or offset, representing ***a time domain DC component corresponds to a peak in the frequency domain at a frequency of zero***.

Another important example is the transform of a cosine function (cos $\omega_0 t$). This function gives us insight into how a basic pure sinusoid transforms into the frequency (Fourier) domain. Intuitively, we would expect such a function to have a singular representation in the frequency domain. We attack this problem by expressing the cosine as the sum of two complex exponentials (using Euler's relation): $\cos\omega_0 t = \frac{1}{2}[e^{-j\omega_0 t} + e^{j\omega_0 t}]$. The transform of this function becomes

$$F(j\omega) = \frac{1}{2}\int_{-\infty}^{\infty}[e^{-j\omega_0 t} + e^{j\omega_0 t}]e^{-j\omega t}\,dt = \frac{1}{2}\left[\int_{-\infty}^{\infty} e^{-j(\omega+\omega_0)t}\,dt + \int_{-\infty}^{\infty} e^{-j(\omega-\omega_0)t}\,dt\right] \quad (6.12)$$

Both integrals in this expression can be evaluated easily using the result for the exponential equation that we obtained earlier $\int_{-\infty}^{\infty} e^{-j\omega t}\,dt = 2\pi\delta(\omega)$, replacing ω with $\omega+\omega_0$ and $\omega-\omega_0$, respectively. Thus, the Fourier transform of a cosine function (a real and even symmetric function) results in

$$\cos(\omega_0 t) \quad \Leftrightarrow \quad \pi[\delta(\omega+\omega_0) + \delta(\omega-\omega_0)] \quad (6.13)$$

this is also real and even — that is, a delta function at an angular frequency of ω_0 and another at $-\omega_0$. While the impulse in the positive frequency domain is straightforward, the concept of negative frequency, originating from the complex series representation in Fourier analysis (Chapter 5, Section 5.3), defies commonsense interpretations. Following a similar logic as that applied earlier, it can be shown that a real and odd symmetric function results in an imaginary odd function:

$$\sin(\omega_0 t) \quad \Leftrightarrow \quad j\pi[\delta(\omega+\omega_0) - \delta(\omega-\omega_0)] \quad (6.14)$$

More commonly, functions that are not purely odd or even have both real and imaginary parts for each frequency component ω_n:

$$F(j\omega_n) = c_n = a_n + jb_n \quad (6.15)$$

That is, for each frequency component in the time domain, we obtain a complex number in the frequency domain. This number is proportional to the cosine and sine amplitudes.

In Chapter 7, we will show how to combine the real and imaginary parts into a metric representing the power for each frequency component in a signal.

6.3 DISCRETE FOURIER TRANSFORM AND THE FFT ALGORITHM

6.3.1 Relationship between Continuous and Discrete Fourier Transform

The continuous and discrete Fourier transforms and their inverses are related but not identical. For the discrete pair, we use a discrete time scale and a discrete frequency scale (Fig. 6.3). Because we want to apply the discrete transform to sampled real-world signals, both the time and frequency scales must also necessarily be finite. Furthermore, we can establish that both scales must be related. For example, in a signal that is observed over a 10 s interval T and sampled at an interval Δt = 1 ms (0.001 s), these parameters determine the range and precision of the discrete Fourier transform of that signal. It is intuitively clear that in a 10 s interval, we cannot reliably distinguish frequencies below a precision of $\Delta f = 1/T = 1/10$ Hz and that the maximum frequency that fits within the sample interval is $1/\Delta t = 1/0.001 = 1000$ Hz. In angular frequency terms, the precision and maximum frequency translate into a step size of $\Delta\omega = 2\pi \times 1/10$ rad/s and a range of $\Omega = 2\pi \times 1000$ rad/s (Fig. 6.3).

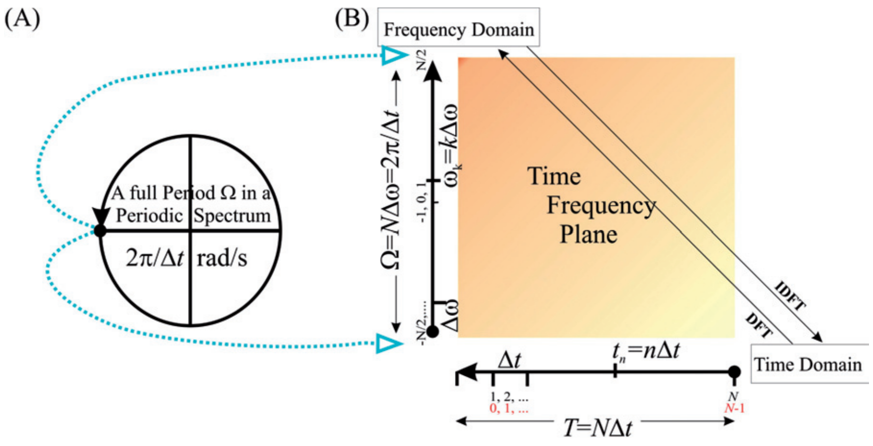


Figure 6.3 The time domain and frequency domain scales in the discrete Fourier transform. The Fourier spectrum is periodic, represented by a circular scale in (A). This circular frequency domain scale is mapped onto a line represented by the ordinate in (B). The abscissa in (B) is the time domain scale; note that on the frequency (vertical) axis, the point $-N/2$ is included and $N/2$ is not. Each sample in the time domain can be considered to represent the preceding sample interval. Depending on which convention is used, the first sample in the time domain is either counted as the 0*th* sample (indicated in red) or the first one (indicated in black).

Note: From earlier discussions about the Nyquist frequency (Chapter 2), we know that the highest frequency we can observe is actually $2\pi \times$ 500 rad/s, *half* of Ω. This point will resurface in the next chapter when the actual spectra are introduced and we find that these spectra contain two symmetric halves.

The discrete approximation $F_a(j\omega)$ of the continuous Fourier transform $F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt$ sampled over a finite interval including N samples is

$$F_a(j\omega_k) = \sum_{n=1}^{N} f(t_n) \exp(-j\omega_k t_n) \Delta t \tag{6.16}$$

Discrete time signals are usually created by sampling a continuous time process; each sample thereby represents the signal immediately preceding analog-to-digital conversion. Using this approach, we have a sampled series representing N intervals of length Δt each (Fig. 6.3). For several reasons, it is common practice to use a range for n from 0 to $N - 1$ instead of 1 to N. Furthermore, in Equation (6.17) the time (t) and angular frequency (ω) are represented by discrete variables as indicated in Figure 6.3. With $t_n = n\Delta t$ and $\omega_k = k\Delta\omega = \dfrac{k2\pi}{N\Delta t}$ (Fig. 6.3), we can write $F_a(j\omega)$ as

$$F_a(j\omega_k) = \Delta t \sum_{n=0}^{N-1} f(t_n) e^{-j\frac{2\pi}{N}kn} = \Delta t \sum_{n=0}^{N-1} f(t_n) W_N^{kn} \tag{6.17}$$

where W_N^{kn} can be thought of as a notational simplification of the exponential term. ***Smuggling Δt out of the previous expression***, changing $f(t_n)$ to $x(n)$ and $F_a(j\omega_k)$ to $X(k)$ yields the standard definition for the discrete Fourier transform (DFT):

$$\boxed{X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}} \tag{6.18}$$

Similarly, the inverse continuous Fourier transform (ICFT) can be approximated by

$$f_a(t_n) = \frac{1}{2\pi} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} F_a(j\omega_k) e^{j\omega_k t_n}\, \Delta\omega \tag{6.19}$$

Note that the upper summation limit does not include $N/2$; due to the circular scale of ω, $-N/2$ and $N/2$ are the same (Fig. 6.3). Changing the range of the summation from $-N/2 \to (N/2) - 1$ into $0 \to N - 1$ and $\Delta\omega = 2\pi/N\Delta t$ (see the axis in Fig. 6.3), Equation (6.19) yields

$$f_a(t_n) = \frac{1}{2\pi}\sum_{k=0}^{N-1} F_a(j\omega_k)e^{j\omega_k t_n}\frac{2\pi}{N\Delta t} = \frac{1}{N\Delta t}\sum_{k=0}^{N-1} F_a(j\omega_k)e^{j\omega_k t_n} \tag{6.20}$$

We now use $t_n = n\Delta t$ and $\omega_k = \dfrac{k2\pi}{N\Delta t}$ (see Fig. 6.3), ***smuggle* Δt *back***, change $f_a(t_n)$ to $x(n)$, and $F_a(j\omega_k)$ to $X(k)$, thereby obtaining the expression for the discrete inverse Fourier transform:

$$\boxed{x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn}} \tag{6.21}$$

To keep the transform ⇔ inverse transform pair consistent, the division by Δt is corrected in the inverse discrete Fourier transform (IDFT), where the expression is multiplied by Δt.

6.3.2 The Twiddle Factor

The weighting factor introduced as W_N in the preceding formulae plays an important role in the practical development of DFT algorithms including the optimized one known as the fast Fourier transform (FFT). The efficiency of this algorithm relies crucially of the fact that this factor, also known as the ***twiddle factor***, is periodic.

6.3.3 DFT versus a Base-2 FFT

The basic idea used to optimize the DFT algorithm involves using the periodicity in the twiddle factor to combine terms and therefore reduce the number of computationally demanding multiplication steps required for a given number of samples (Cooley and Tukey, 1965). Specifically, the standard formulation of the DFT of a time series with N values requires N^2 multiplications for a time series with N points, whereas the FFT requires only $N\log_2(N)$ multiplications.

For example, consider a 4-point time series: $x(0)$, $x(1)$, $x(2)$, $x(3)$, and its DFT $X(0)$, $X(1)$, $X(2)$, $X(3)$. For $N = 4$, each of the X values is calculated with

$$X(k) = \sum_{n=0}^{3} x(n)W_4^{kn} \tag{6.22}$$

If we were to perform the DFT directly from Equation (6.22), we would have four multiplications for each $X(k)$; since we have $X(0) - X(3)$, this

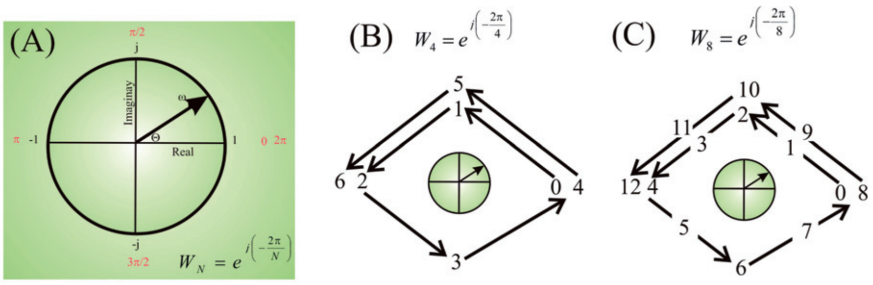

(A)
π/2
j
Imaginay
ω
Θ
π -1
1
0 2π
Real
-j
3π/2
$W_N = e^{j\left(-\frac{2\pi}{N}\right)}$
(B)
$W_4 = e^{j\left(-\frac{2\pi}{4}\right)}$
5
1
6 2
0 4
3
(C)
$W_8 = e^{j\left(-\frac{2\pi}{8}\right)}$
10
2
11
3
9
1
12 4
0 8
5
7
6

Figure 6.4 Periodicity of the twiddle factor W_N. (A) The values of Θ are indicated in red and the real and imaginary components in black. For instance, $-1 + j0$ is associated with $\Theta = \pi$, $0 + j$ is associated with $\Theta = \pi/2$, and so on. It can be seen that for $\Theta = 0$ or 2π the values are identical $(1 + j0)$ due to the periodicity of W_N. In (B) and (C) concrete examples are provided for the periodicity of a 4-point (W_4) and 8-point (W_8) algorithm. The numbers correspond to the powers of the twiddle factor (e.g., $0 \rightarrow W_4^0$; $1 \rightarrow W_4^1$; $2 \rightarrow W_4^2$); in case of $N = 4$, a cycle is completed in four steps; whereas for $N = 8$, the cycle is completed in eight steps. In the first case, (B): $W_4^0 = W_4^4$, $W_4^1 = W_4^5$, and $W_4^2 = W_4^6$. In the second example, (C): $W_8^0 = W_8^8$, $W_8^2 = W_8^{10}$, and so on.

leads to a total of $4 \times 4 = 16$ multiplications. However, since the expression in Equation (6.22) is a summation, we can split the problem into odd and even components:

$$X(k) = \sum_{r=0}^{1} x(2r) W_4^{2rk} + \sum_{r=0}^{1} x(2r+1) W_4^{(2r+1)k} \tag{6.23}$$

The second twiddle factor can be separated into two factors:

$$W_4^{(2r+1)k} = W_4^{2rk} W_4^k \tag{6.24}$$

Expanding the summations for $X(0) - X(3)$ and combining terms we get

$$\begin{aligned}
X(0) &= x(0)W_4^0 + x(2)W_4^0 + x(1)W_4^0W_4^0 + x(3)W_4^0W_4^0 \\
&= \left[x(0) + x(2)W_4^0\right] + W_4^0\left[x(1) + x(3)W_4^0\right] \\
X(1) &= x(0)W_4^0 + x(2)W_4^2 + x(1)W_4^1W_4^0 + x(3)W_4^1W_4^2 \\
&= \left[x(0) + x(2)W_4^2\right] + W_4^1\left[x(1) + x(3)W_4^2\right] \\
X(2) &= x(0)W_4^0 + x(2)W_4^4 + x(1)W_4^2W_4^0 + x(3)W_4^2W_4^4 \\
&= \left[x(0) + x(2)W_4^4\right] + W_4^2\left[x(1) + x(3)W_4^4\right] \\
X(3) &= x(0)W_4^0 + x(2)W_4^6 + x(1)W_4^3W_4^0 + x(3)W_4^3W_4^6 \\
&= \left[x(0) + x(2)W_4^6\right] + W_4^3\left[x(1) + x(3)W_4^6\right]
\end{aligned} \tag{6.25}$$

Note that with the exception of the first line in Equation (6.25), we used $W_4^0 = 1$. In the first equation, we kept W_4^0 in the expression solely to emphasize the similarity between all the expressions for $X(k)$. Further, we exploited the fact that the twiddle factors (W) represent periodic exponentials (Fig. 6.4B) — that is,

$$W_4^4 = \exp\left\{j\left(-\frac{2\pi}{4}\right)4\right\} = 1 = W_4^0 \text{ and } W_4^6 = \exp\left\{j\left(-\frac{2\pi}{4}\right)6\right\} = -1 = W_4^2 \quad (6.26)$$

A MATLAB script representing Equation (6.25) may look like the following:

```
X(0+1)=(x(0+1)+x(2+1)*W0)+W0*(x(1+1)+x(3+1)*W0);
X(1+1)=(x(0+1)+x(2+1)*W2)+W1*(x(1+1)+x(3+1)*W2);
X(2+1)=(x(0+1)+x(2+1)*W0)+W2*(x(1+1)+x(3+1)*W0);
X(3+1)=(x(0+1)+x(2+1)*W2)+W3*(x(1+1)+x(3+1)*W2);
```

In this script, the time series is **x**, its transform is **X**, and the twiddle factors are **W0** to **W3**. Parenthetically, all indices are augmented with one because MATLAB does not allow zero indices for vectors.

You may note that there are still 12 multiplications here, an improvement over the $4 \times 4 = 16$ multiplications for the brute-force approach but more than the expected $4\log_2(4) = 8$ multiplications. However, if we take advantage of the repeated multiplications in the preceding algorithm (i.e., x(2+1)*W0, x(2+1)*W2, x(3+1)*W0, and x(3+1)*W2), we end up with $12 - 4 = 8$ multiplications.

It is easier to see the algorithm flow in a diagram where the nodes are variables and the lines represent the operations on those variables (Fig. 6.5). In this example, the input variables (left) are added in the output. In two of the cases, the input is multiplied by a twiddle factor (i.e., W_x and W_y in Fig. 6.5).

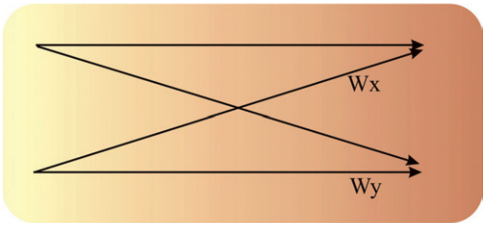


Figure 6.5 A flow diagram that forms the basis of the FFT algorithm. The base diagram is known as the ***FFT butterfly***.

6.3.4 Examples

1. A 4-point example.

The following listing is an example of a 4-point FFT MATLAB script. To clearly show the specific features of the FFT algorithm diagram in Fig. 6.6, the program has not been optimized to avoid redundant operations.

```
% pr6_1.m
% A four point FFT
clear

x(0+1)=0;                    % input time series x(n)
x(1+1)=1;                    % Indices are augmented by 1
x(2+1)=1;                    % MATLAB indices start at 1
x(3+1)=0;                    % instead of 0

W4=exp(j*2*pi/4);            % the W4 twiddle factor
W0=W4^0;                     % and the 0-3rd powers
W1=W4^1;
W2=W4^2;
W3=W4^3;

X(0+1)=(x(0+1)+x(2+1)*W0)+W0*(x(1+1)+x(3+1)*W0);
X(1+1)=(x(0+1)+x(2+1)*W2)+W1*(x(1+1)+x(3+1)*W2);
X(2+1)=(x(0+1)+x(2+1)*W0)+W2*(x(1+1)+x(3+1)*W0);
X(3+1)=(x(0+1)+x(2+1)*W2)+W3*(x(1+1)+x(3+1)*W2);

% Check with MATLAB fft command
fx=fft(x);

figure
hold
plot(X);
plot(fx,'r+');
```

2. An 8-point example.
   Evaluate the diagram in Figure 6.7 with the **MATLAB** ***script pr6_2.m***. While the signal vector indices seem rather arbitrarily ordered, a binary representation can make this indexing more straightforward. Table 6.2 provides an overview of binary numbers and how they relate to this index scrambling procedure.

   In the example in Figure 6.7, the input time series is x(0), x(1), . . . , x(7). Note that the input of the algorithm is the lowercase vector x, and the output is represented by a capital X. First the input is ***scrambled*** to obtain the input to the FFT algorithm: SX(0) = x(0),

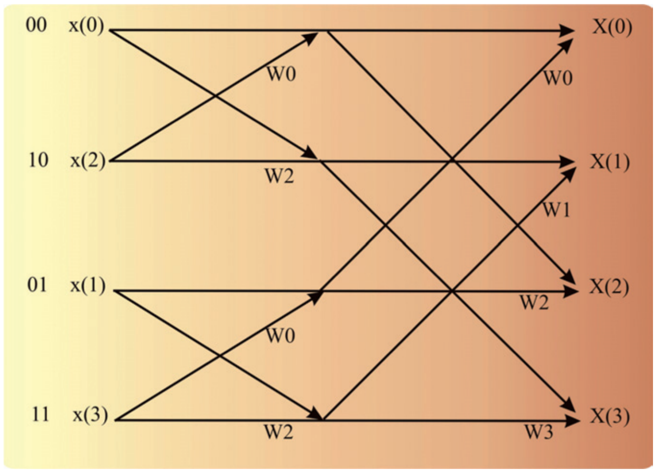
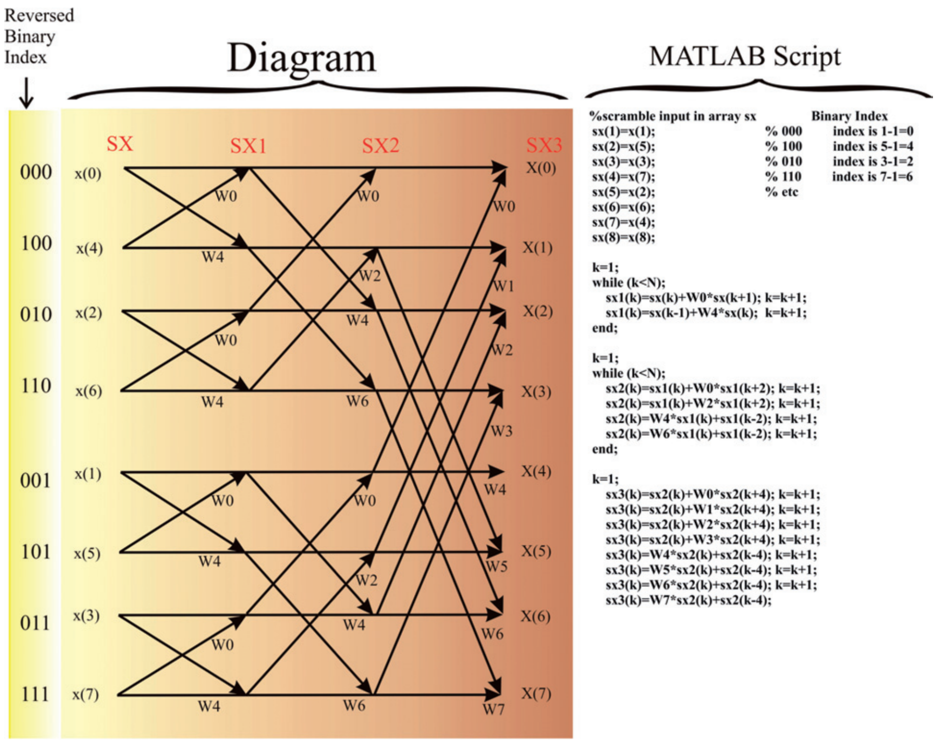


Figure 6.6 Diagram for a 4-point FFT. See the following for the diagram's implementation in MATLAB script. Here we used $W_4^0 = W_4^4$ and $W_4^2 = W_4^6$ (Fig. 6.4) to optimize the algorithm.

Table 6.2 A 3-Bit Binary Set of Numbers to Explain Scrambling in FFT Input

Binary-decimal	MATLAB index	Inverted binary-decimal	MATLAB index
000 = 0	1	000 = 0	1
001 = 1	2	100 = 4	5
010 = 2	3	010 = 2	3
011 = 3	4	110 = 6	7
100 = 4	5	001 = 1	2
101 = 5	6	101 = 5	6
110 = 6	7	011 = 3	4
111 = 7	8	111 = 7	8

SX(1) = x(4), . . . , SX(7) = x(7). The scrambling process can be presented by reversing the index in binary format. In our time series, we have 8 data points (x(0) – x(7)); to represent indexes from 0 to 7, we need 3 bits ($2^3 = 8$). We use the reverse binary values to scramble the input, for example, SX with index 1 (in binary 3-bit representation 001) becomes x with index 4 (in binary representation 100). An overview for all indexes can be found in Table 6.2. Note that in the MATLAB script example, all indexes are increased by one (SX(0) → SX(1) etc.) because MATLAB cannot work with a zero index. After the input is scrambled, the rest of the diagram shows the flow of the calculations. For instance, working backward in the diagram from X(3):



```
%scramble input in array sx                Binary Index
 sx(1)=x(1);                         % 000     index is 1-1=0
 sx(2)=x(5);                         % 100     index is 5-1=4
 sx(3)=x(3);                         % 010     index is 3-1=2
 sx(4)=x(7);                         % 110     index is 7-1=6
 sx(5)=x(2);                         % etc
 sx(6)=x(6);
 sx(7)=x(4);
 sx(8)=x(8);

 k=1;
 while (k<N);
   sx1(k)=sx(k)+W0*sx(k+1); k=k+1;
   sx1(k)=sx(k-1)+W4*sx(k);  k=k+1;
 end;

 k=1;
 while (k<N);
   sx2(k)=sx1(k)+W0*sx1(k+2); k=k+1;
   sx2(k)=sx1(k)+W2*sx1(k+2); k=k+1;
   sx2(k)=W4*sx1(k)+sx1(k-2); k=k+1;
   sx2(k)=W6*sx1(k)+sx1(k-2); k=k+1;
 end;

 k=1;
   sx3(k)=sx2(k)+W0*sx2(k+4); k=k+1;
   sx3(k)=sx2(k)+W1*sx2(k+4); k=k+1;
   sx3(k)=sx2(k)+W2*sx2(k+4); k=k+1;
   sx3(k)=sx2(k)+W3*sx2(k+4); k=k+1;
   sx3(k)=W4*sx2(k)+sx2(k-4); k=k+1;
   sx3(k)=W5*sx2(k)+sx2(k-4); k=k+1;
   sx3(k)=W6*sx2(k)+sx2(k-4); k=k+1;
   sx3(k)=W7*sx2(k)+sx2(k-4);
```

Figure 6.7 A diagram of an 8-point FFT.

$$X(3) = SX2(3) + W3 \times SX2(7)$$

with

$$SX2(3) = W6 \times SX1(3) + SX1(1) \text{ and } SX2(7) = W6 \times SX1(7) + SX1(5)$$

with

$$SX1(1) = SX(0) + W4 \times SX(1) \ldots \text{etc.}$$

As you can see, the creation of the algorithm from the diagram is tedious but not difficult. The associated part of the MATLAB script in Figure 6.7 reflects the diagram with all indexes increased by one. The purpose of this example script is to make the FFT operation transparent; therefore, this example script is neither optimized for efficiency nor particularly elegant (from a programmer's perspective).

6.4 UNEVENLY SAMPLED DATA

All of the examples presented here assume that we have sampled the data evenly (i.e., the interval Δt between the sample points of the time series is constant). Usually this is an appropriate assumption, but there are examples where uneven sampling cannot be avoided. Spike trains (Chapter 14) or time series representing heart rate are examples; in these examples, the events represent samples of an underlying process that is invisible to the experimenter. The heart rate signal is usually determined by measuring the intervals between peaks in the QRS complexes (Chapter 4, Fig. 4.7). The interval (or its inverse value) between pairs of subsequent QRS complexes is a measure of the instantaneous rate. This rate value can be positioned in a time series at the instant of either the first or second QRS complex of the pair, and because the heartbeats do occur at slightly irregular intervals, the time series is sampled unevenly. In principle, there are several solutions to this problem:

1. The unevenly sampled time series is reconstructed by using interpolation (i.e., the signal is resampled at evenly spaced intervals). The interpolation technique (e.g., linear, cubic, spline) may vary with the application. In MATLAB, resampling may be accomplished with the interp1 command or any of the other related functions. After resampling the time series, one can use standard Fourier analysis methods. The disadvantage is that the interpolation algorithm may introduce frequency components that are not related to the underlying process.
2. The measurements can be represented as the number of events in a binned trace. Because the bins are equally spaced, standard DFT/FFT can be applied. In case of low-frequency activity, the bins must be relatively wide to avoid an over-representation of empty bins. The disadvantage is that wide bins represent a low sample rate and associated low Nyquist frequency.
3. The most elegant solution is to use the so-called Lomb algorithm for estimating the spectrum. This algorithm is especially designed to deal with unevenly sampled time series directly without the assumptions demanded by interpolation and resampling techniques (e.g., Press et al., 1992).