A similar arrangement can be used to store the computed DFT values. In particular, the mapping is from the index k to a pair of indices (p, q), where $0 \le p \le L-1$ and $0 \le q \le M-1$. If we select the mapping

$$k = Mp + q \tag{8.1.11}$$

the DFT is stored on a row-wise basis, where the first row contains the first M elements of the DFT $X(k)$, the second row contains the next set of M elements, and so on. On the other hand, the mapping

$$k = qL + p \tag{8.1.12}$$

results in a column-wise storage of $X(k)$, where the first L elements are stored in the first column, the second set of L elements are stored in the second column, and so on.

Now suppose that $x(n)$ is mapped into the rectangular array $x(l, m)$ and $X(k)$ is mapped into a corresponding rectangular array $X(p, q)$. Then the DFT can be expressed as a double sum over the elements of the rectangular array multiplied by the corresponding phase factors. To be specific, let us adopt a column-wise mapping for $x(n)$ given by (8.1.10) and the row-wise mapping for the DFT given by (8.1.11). Then

$$X(p, q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l, m) W_N^{(Mp+q)(mL+l)} \tag{8.1.13}$$

But

$$W_N^{(Mp+q)(mL+l)} = W_N^{MLmp} W_N^{mLq} W_N^{Mpl} W_N^{lq} \tag{8.1.14}$$

However, $W_N^{Nmp} = 1$, $W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$, and $W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$.

With these simplications, (8.1.13) can be expressed as

$$X(p, q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l, m) W_M^{mq} \right] \right\} W_L^{lp} \tag{8.1.15}$$

The expression in (8.1.15) involves the computation of DFTs of length M and length L. To elaborate, let us subdivide the computation into three steps:

1. First, we compute the M-point DFTs

$$F(l, q) \equiv \sum_{m=0}^{M-1} x(l, m) W_M^{mq}, \qquad 0 \le q \le M-1 \tag{8.1.16}$$

for each of the rows $l = 0, 1, \ldots, L-1$.

2. Second, we compute a new rectangular array $G(l, q)$ defined as

$$G(l, q) = W_N^{lq} F(l, q), \qquad \begin{matrix} 0 \le l \le L-1 \\ 0 \le q \le M-1 \end{matrix} \tag{8.1.17}$$

3. Finally, we compute the L-point DFTs

$$X(p, q) = \sum_{l=0}^{L-1} G(l, q) W_L^{lp} \tag{8.1.18}$$

for each column $q = 0, 1, \ldots, M-1$, of the array $G(l, q)$.

On the surface it may appear that the computational procedure outlined above is more complex than the direct computation of the DFT. However, let us evaluate the computational complexity of (8.1.15). The first step involves the computation of L DFTs, each of M points. Hence this step requires LM^2 complex multiplications and $LM(M-1)$ complex additions. The second step requires LM complex multiplications. Finally, the third step in the computation requires ML^2 complex multiplications and $ML(L-1)$ complex additions. Therefore, the computational complexity is

$$\begin{aligned} &\text{Complex multiplications:} \quad N(M+L+1) \\ &\text{Complex additions:} \quad N(M+L-2) \end{aligned} \tag{8.1.19}$$

where $N = ML$. Thus the number of multiplications has been reduced from N^2 to $N(M+L+1)$ and the number of additions has been reduced from $N(N-1)$ to $N(M+L-2)$.

Let us consider the computation of the $N = 2^\nu$ point DFT by the divide-and-conquer approach specified by (6.1.16) through (6.1.18). We select $M = N/2$ and $L = 2$. This selection results in a split of the N-point data sequence into two $N/2$-point data sequences $f_1(n)$ and $f_2(n)$, corresponding to the even-numbered and odd-numbered samples of $x(n)$, respectively, that is,

$$\begin{aligned} f_1(n) &= x(2n) \\ f_2(n) &= x(2n+1), \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{6.1.23}$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating $x(n)$ by a factor of 2, and hence the resulting FFT algorithm is called a decimation-in-time algorithm.

Now the N-point DFT can be expressed in terms of the DFTs of the decimated sequences as follows:

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

$$\begin{aligned} &= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} \\ &= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)} \end{aligned} \tag{6.1.24}$$

But $W_N^2 = W_{N/2}$. With this substitution, (6.1.24) can be expressed as

$$\begin{aligned} X(k) &= \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km} \\ &= F_1(k) + W_N^k F_2(k) \qquad k = 0, 1, \ldots, N-1 \end{aligned} \tag{6.1.25}$$

where $F_1(k)$ and $F_2(k)$ are the $N/2$-point DFTs of the sequences $f_1(m)$ and $f_2(m)$, respectively.

Since $F_1(k)$ and $F_2(k)$ are periodic, with period $N/2$, we have $F_1(k + N/2) = F_1(k)$ and $F_2(k + N/2) = F_2(k)$. In addition, the factor $W_N^{k+N/2} = -W_N^k$. Hence (6.1.25) can be expressed as

$$X(k) = F_1(k) + W_N^k F_2(k) \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.1.26}$$

$$X\left(k + \frac{N}{2}\right) = F_1(k) - W_N^k F_2(k) \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.1.27}$$

Observe that the basic computation performed at every stage, as illustrated in Fig. 6.6, is to take two complex numbers, say the pair (a, b), multiply b by W_N^r, and then add and subtract the product from a to form two new complex numbers (A, B). This basic computation, which is shown in Fig. 6.7, is called a *butterfly* because the flow graph resembles a butterfly.

In general, each butterfly involves one complex multiplication and two complex additions. For $N = 2^\nu$, there are $N/2$ butterflies per stage of the computation process and $\log_2 N$ stages. Therefore, as previously indicated the total number of complex multiplications is $(N/2)\log_2 N$ and complex additions is $N \log_2 N$.

Once a butterfly operation is performed on a pair of complex numbers (a, b) to produce (A, B), there is no need to save the input pair (a, b). Hence we can

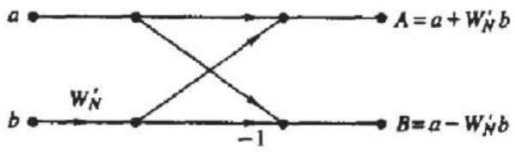


Figure 6.7 Basic butterfly computation in the decimation-in-time FFT algorithm.

Another important radix-2 FFT algorithm, called the decimation-in-frequency algorithm, is obtained by using the divide-and-conquer approach described in Section 6.1.2 with the choice of $M = 2$ and $L = N/2$. This choice of parameters implies a column-wise storage of the input data sequence. To derive the algorithm, we begin by splitting the DFT formula into two summations, one of which involves the sum over the first $N/2$ data points and the second sum involves the last $N/2$ data points. Thus we obtain

$$\begin{aligned} X(k) &= \sum_{n=0}^{(N/2)-1} x(n) W_N^{kn} + \sum_{n=N/2}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{(N/2)-1} x(n) W_N^{kn} + W_N^{Nk/2} \sum_{n=0}^{(N/2)-1} x\left(n + \frac{N}{2}\right) W_N^{kn} \end{aligned} \tag{6.1.33}$$

Since $W_N^{kN/2} = (-1)^k$, the expression (6.1.33) can be rewritten as

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \tag{6.1.34}$$

Now, let us split (decimate) $X(k)$ into the even- and odd-numbered samples. Thus we obtain

$$X(2k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N/2}^{kn} \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.1.35}$$

and

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} \left\{ \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \right\} W_{N/2}^{kn} \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.1.36}$$

where we have used the fact that $W_N^2 = W_{N/2}$.

If we define the $N/2$-point sequences $g_1(n)$ and $g_2(n)$ as

$$\begin{aligned} g_1(n) &= x(n) + x\left(n + \frac{N}{2}\right) \\ g_2(n) &= \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \qquad n = 0, 1, 2, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{6.1.37}$$

then

$$\begin{aligned} X(2k) &= \sum_{n=0}^{(N/2)-1} g_1(n) W_{N/2}^{kn} \\ X(2k+1) &= \sum_{n=0}^{(N/2)-1} g_2(n) W_{N/2}^{kn} \end{aligned} \tag{6.1.38}$$

The computation of the sequences $g_1(n)$ and $g_2(n)$ according to (6.1.37) and the subsequent use of these sequences to compute the $N/2$-point DFTs are depicted in Fig. 6.9. We observe that the basic computation in this figure involves the butterfly operation illustrated in Fig. 6.10.

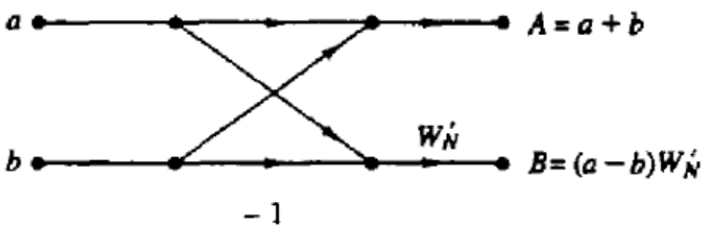


Figure 6.10 Basic butterfly computation in the decimation-in-frequency FFT algorithm.

$$H(\omega) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < \omega \le \pi \end{cases} \tag{10.1.1}$$

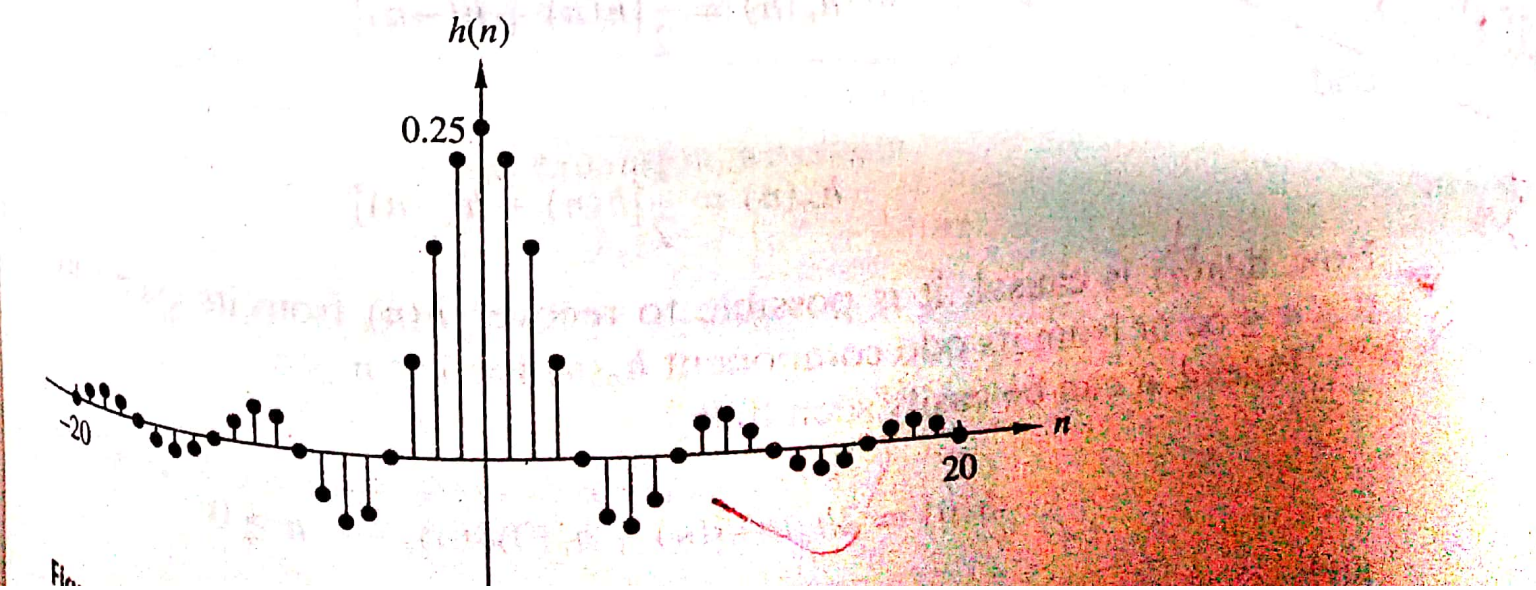
The impulse response of this filter is

$$h(n) = \begin{cases} \dfrac{\omega_c}{\pi}, & n = 0 \\ \dfrac{\omega_c}{\pi} \dfrac{\sin \omega_c n}{\omega_c n}, & n \neq 0 \end{cases} \tag{10.1.2}$$

A plot of $h(n)$ for $\omega_c = \pi/4$ is illustrated in Fig. 10.1.1. It is clear that the ideal lowpass filter is noncausal and hence it cannot be realized in practice.

One possible solution is to introduce a large delay n_0 in $h(n)$ and arbitrarily to set $h(n) = 0$ for $n < n_0$. However, the resulting system no longer has an ideal frequency response characteristic. Indeed, if we set $h(n) = 0$ for $n < n_0$, the Fourier series expansion of $H(\omega)$ results in the Gibbs phenomenon, as will be described in Section 10.2.

Although this discussion is limited to the realization of a lowpass filter, our conclusions hold, in general, for all the other ideal filter characteristics. In brief,



Example 4.5

Consider the signal $x(t)$ whose Fourier transform is

$$X(j\omega) = \begin{cases} 1, & |\omega| < W \\ 0, & |\omega| > W \end{cases} \tag{4.18}$$

This transform is illustrated in Figure 4.9(a). Using the synthesis equation (4.8), we can

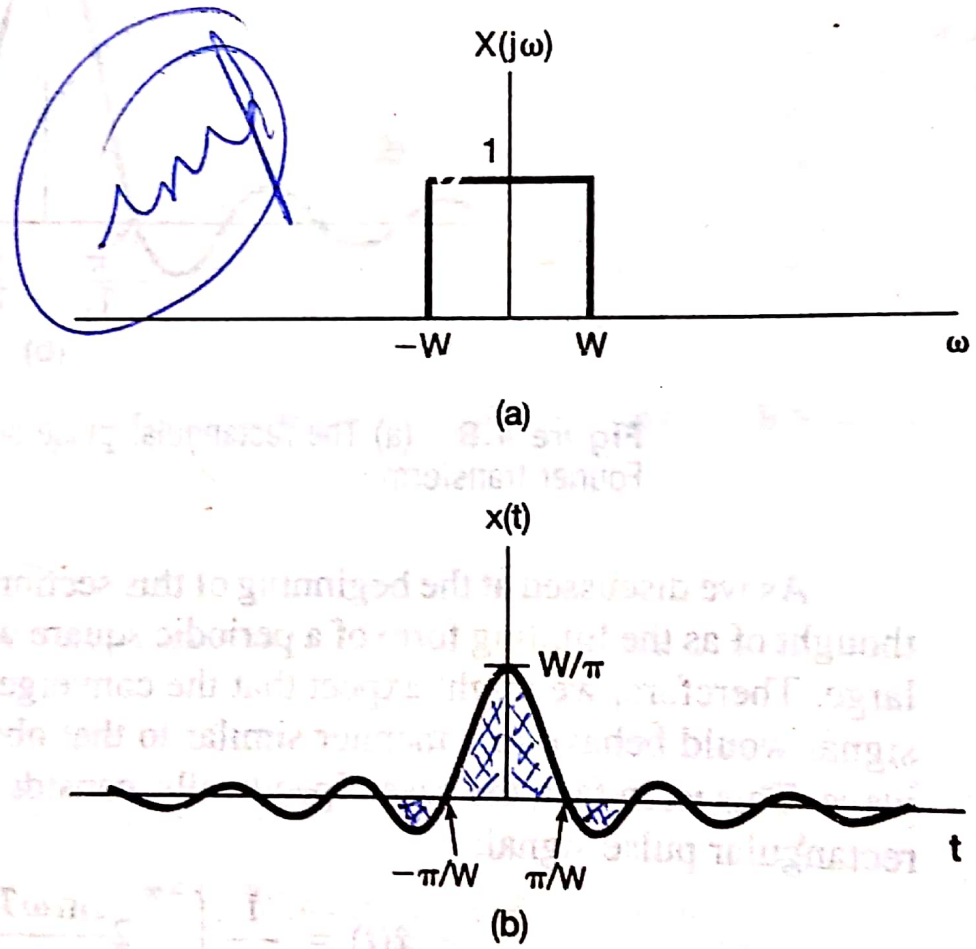


Figure 4.9 Fourier transform pair of Example 4.5: (a) Fourier transform for Example 4.5 and (b) the corresponding time function.

then determine

$$x(t) = \frac{1}{2\pi}\int_{-W}^{W} e^{j\omega t}d\omega = \frac{\sin Wt}{\pi t}, \tag{4.19}$$

which is depicted in Figure 4.9(b).

Comparing Figures 4.8 and 4.9 or, equivalently, eqs. (4.16) and (4.17) with eqs. (4.18) and (4.19), we see an interesting relationship. In each case, the Fourier transform pair consists of a function of the form $(\sin a\theta)/b\theta$ and a rectangular pulse. However, in Example 4.4, it is the *signal* $x(t)$ that is a pulse, while in Example 4.5, it is the *transform* $X(j\omega)$. The special relationship that is apparent here is a direct consequence of the *duality property* for Fourier transforms, which we discuss in detail in Section 4.3.6.

Functions of the form given in eqs. (4.17) and (4.19) arise frequently in Fourier analysis and in the study of LTI systems and are referred to as *sinc functions*. A commonly used precise form for the sinc function is

$$\text{sinc}(\theta) = \frac{\sin \pi\theta}{\pi\theta}. \tag{4.20}$$

The sinc function is plotted in Figure 4.10. Both of the signals in eqs. (4.17) and (4.19) can be expressed in terms of the sinc function:

$$\frac{2\sin \omega T_1}{\omega} = 2T_1 \,\text{sinc}\left(\frac{\omega T_1}{\pi}\right)$$

$$\frac{\sin Wt}{\pi t} = \frac{W}{\pi}\,\text{sinc}\left(\frac{Wt}{\pi}\right).$$