Overlap-save method. In this method the size of the input data blocks is $N = L + M - 1$ and the DFTs and IDFT are of length N. Each data block consists of the last $M - 1$ data points of the previous data block followed by L new data points to form a data sequence of length $N = L + M - 1$. An N-point DFT is computed for each data block. The impulse response of the FIR filter is increased in length by appending $L - 1$ zeros and an N-point DFT of the sequence is computed once and stored. The multiplication of the two N-point DFTs $\{H(k)\}$ and $\{X_m(k)\}$ for the mth block of data yields

$$\hat{Y}_m(k) = H(k)X_m(k), \qquad k = 0, 1, \ldots, N-1 \tag{7.3.7}$$

Then the N-point IDFT yields the result

$$\hat{Y}_m(n) = \{\hat{y}_m(0)\hat{y}_m(1)\cdots\hat{y}_m(M-1)\hat{y}_m(M)\cdots\hat{y}_m(N-1)\} \tag{7.3.8}$$

Since the data record is of length N, the first $M - 1$ points of $y_m(n)$ are corrupted by aliasing and must be discarded. The last L points of $y_m(n)$ are exactly the same as the result from linear convolution and, as a consequence,

$$\hat{y}_m(n) = y_m(n), n = M, M+1, \ldots, N-1 \tag{7.3.9}$$

To avoid loss of data due to aliasing, the last $M - 1$ points of each data record are saved and these points become the first $M - 1$ data points of the subsequent record, as indicated above. To begin the processing, the first $M - 1$ points of the first record are set to zero. Thus the blocks of data sequences are

$$x_1(n) = \{\underbrace{0, 0, \ldots, 0}_{M-1 \text{ points}}, x(0), x(1), \ldots, x(L-1)\} \tag{7.3.10}$$

$$x_2(n) = \{\underbrace{x(L-M+1), \ldots, x(L-1)}_{M-1 \text{data points from } x_1(n)}, \underbrace{x(L), \ldots, x(2L-1)}_{L \text{ new data points}}\} \tag{7.3.11}$$

$$x_3(n) = \{\underbrace{x(2L-M+1), \ldots, x(2L-1)}_{M-1 \text{ data points from } x_2(n)}, \underbrace{x(2L), \ldots, x(3L-1)}_{L \text{ new data points}}\} \tag{7.3.12}$$

and so forth. The resulting data sequences from the IDFT are given by (7.3.8), where the first $M - 1$ points are discarded due to aliasing and the remaining L points constitute the desired result from linear convolution. This segmentation of the input data and the fitting of the output data blocks together to form the output sequence are graphically illustrated in Fig. 7.3.1.

Figure 7.3.1
Linear FIR
overlap-save

Overlap-ad
the size of t
zeros and c

and so on.

The IDFT
the DFTs
by append
Since
from each

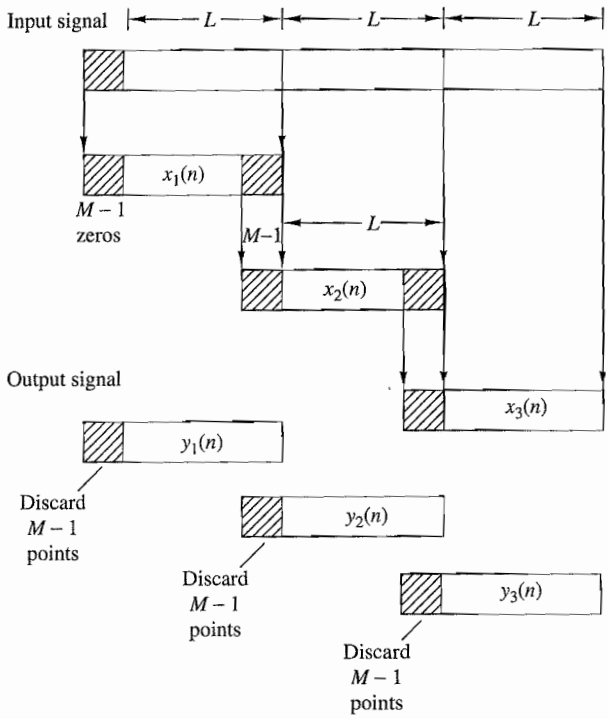


Figure 7.3.1
Linear FIR filtering by the overlap-save method.

Overlap-add method. In this method the size of the input data block is L points and the size of the DFTs and IDFT is $N = L+M-1$. To each data block we append $M-1$ zeros and compute the N-point DFT. Thus the data blocks may be represented as

$$x_1(n) = \{x(0), x(1), \ldots, x(L-1), \underbrace{0, 0, \ldots, 0}_{M-1 \text{ zeros}}\} \tag{7.3.13}$$

$$x_2(n) = \{x(L), x(L+1), \ldots, x(2L-1), \underbrace{0, 0, \ldots, 0}_{M-1 \text{ zeros}}\} \tag{7.3.14}$$

$$x_3(n) = \{x(2L), \ldots, x(3L-1), \underbrace{0, 0, \ldots, 0}_{M-1 \text{ zeros}}\} \tag{7.3.15}$$

and so on. The two N-point DFTs are multiplied together to form

$$Y_m(k) = H(k)X_m(k), \qquad k = 0, 1, \ldots, N-1 \tag{7.3.16}$$

The IDFT yields data blocks of length N that are free of aliasing, since the size of the DFTs and IDFT is $N = L + M - 1$ and the sequences are increased to N-points by appending zeros to each block.

Since each data block is terminated with $M - 1$ zeros, the last $M - 1$ points from each output block must be overlapped and added to the first $M - 1$ points of

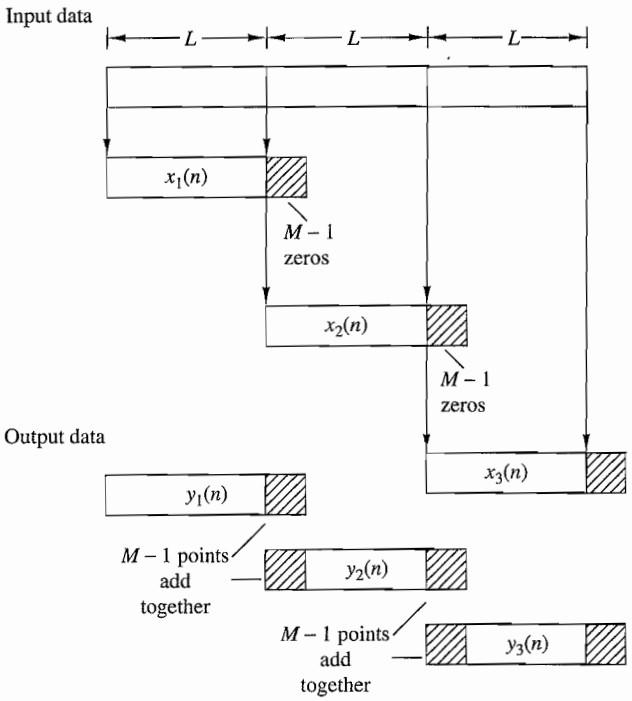


Figure 7.3.2
Linear FIR filtering by the overlap-add method.

the succeeding block. Hence this method is called the overlap-add method. This overlapping and adding yields the output sequence

$$y(n) = \{y_1(0), y_1(1), \ldots, y_1(L-1), y_1(L) + y_2(0), y_1(L+1) + y_2(1), \ldots, y_1(N-1) + y_2(M-1), y_2(M), \ldots\} \tag{7.3.17}$$

The segmentation of the input data into blocks and the fitting of the output data blocks to form the output sequence are graphically illustrated in Fig. 7.3.2.

At this point, it may appear to the reader that the use of the DFT in linear FIR filtering not only is an indirect method of computing the output of an FIR filter, but also may be more expensive computationally, since the input data must first be converted to the frequency domain via the DFT, multiplied by the DFT of the FIR filter, and finally, converted back to the time domain via the IDFT. On the contrary, however, by using the fast Fourier transform algorithm, as will be shown in Chapter 8, the DFTs and IDFT require fewer computations to compute the output sequence than the direct realization of the FIR filter in the time domain. This computational efficiency is the basic advantage of using the DFT to compute the output of an FIR filter.

7.4 Frequency Analysis of Signals Using the DFT

To compute the spectrum of either a continuous-time or discrete-time signal, the values of the signal for all time are required. However, in practice, we observe signals for only a finite duration. Consequently, the spectrum of a signal can only be

approximated from a finite data record. In this section we examine the implications of a finite data record in frequency analysis using the DFT.

If the signal to be analyzed is an analog signal, we would first pass it through an antialiasing filter and then sample it at a rate $F_s \geq 2B$, where B is the bandwidth of the filtered signal. Thus the highest frequency that is contained in the sampled signal is $F_s/2$. Finally, for practical purposes, we limit the duration of the signal to the time interval $T_0 = LT$, where L is the number of samples and T is the sample interval. As we shall observe in the following discussion, the finite observation interval for the signal places a limit on the frequency resolution; that is, it limits our ability to distinguish two frequency components that are separated by less than $1/T_0 = 1/LT$ in frequency.

Let $\{x(n)\}$ denote the sequence to be analyzed. Limiting the duration of the sequence to L samples, in the interval $0 \leq n \leq L-1$, is equivalent to multiplying $\{x(n)\}$ by a rectangular window $w(n)$ of length L. That is,

$$\hat{x}(n) = x(n)w(n) \tag{7.4.1}$$

where

$$w(n) = \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases} \tag{7.4.2}$$

Now suppose that the sequence $x(n)$ consists of a single sinusoid, that is,

$$x(n) = \cos \omega_0 n \tag{7.4.3}$$

Then the Fourier transform of the finite-duration sequence $x(n)$ can be expressed as

$$\hat{X}(\omega) = \frac{1}{2}[W(\omega - \omega_0) + W(\omega + \omega_0)] \tag{7.4.4}$$

where $W(\omega)$ is the Fourier transform of the window sequence, which is (for the rectangular window)

$$W(\omega) = \frac{\sin(\omega L/2)}{\sin(\omega/2)} e^{-j\omega(L-1)/2} \tag{7.4.5}$$

To compute $\hat{X}(\omega)$ we use the DFT. By padding the sequence $\hat{x}(n)$ with $N-L$ zeros, we can compute the N-point DFT of the truncated (L points) sequence $\{\hat{x}(n)\}$. The magnitude spectrum $|\hat{X}(k)| = |\hat{X}(\omega_k)|$ for $\omega_k = 2\pi k/N$, $k = 0, 1, \ldots, N$, is illustrated in Fig. 7.4.1 for $L = 25$ and $N = 2048$. We note that the windowed spectrum $\hat{X}(\omega)$ is not localized to a single frequency, but instead it is spread out over the whole frequency range. Thus the power of the original signal sequence $\{x(n)\}$ that was concentrated at a single frequency has been spread by the window into the entire frequency range. We say that the power has "leaked out" into the entire frequency range. Consequently, this phenomenon, which is a characteristic of windowing the signal, is called *leakage*.

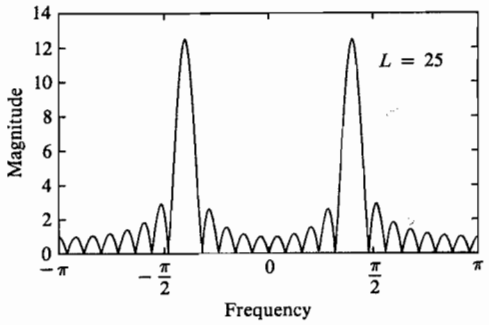


Figure 7.4.1
Magnitude spectrum for $L = 25$ and $N = 2048$, illustrating the occurrence of leakage.

Windowing not only distorts the spectral estimate due to the leakage effects, it also reduces spectral resolution. To illustrate this problem, let us consider a signal sequence consisting of two frequency components,

$$x(n) = \cos \omega_1 n + \cos \omega_2 n \tag{7.4.6}$$

When this sequence is truncated to L samples in the range $0 \leq n \leq L - 1$, the windowed spectrum is

$$\hat{X}(\omega) = \frac{1}{2}[W(\omega - \omega_1) + W(\omega - \omega_2) + W(\omega + \omega_1) + W(\omega + \omega_2)] \tag{7.4.7}$$

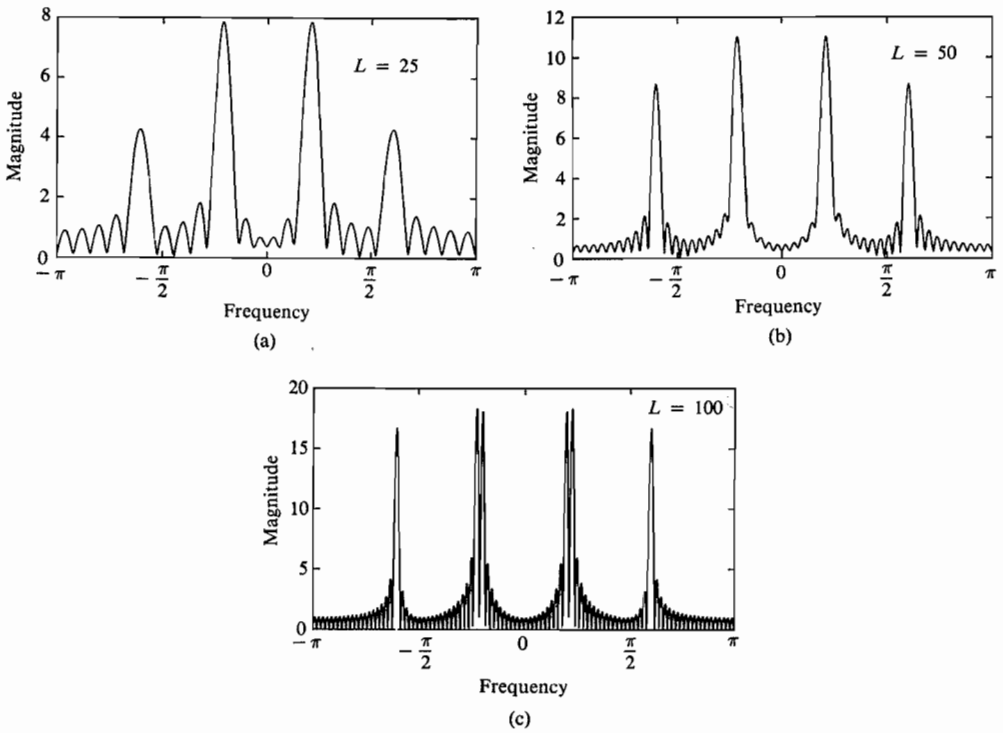


Figure 7.4.2 Magnitude spectrum for the signal given by (7.4.8), as observed through a rectangular window.

The spectrum $W(\omega)$ of the rectangular window sequence has its first zero crossing at $\omega = 2\pi/L$. Now if $|\omega_1 - \omega_2| < 2\pi/L$, the two window functions $W(\omega - \omega_1)$ and $W(\omega - \omega_2)$ overlap and, as a consequence, the two spectral lines in $x(n)$ are not distinguishable. Only if $(\omega_1 - \omega_2) \geq 2\pi/L$ will we see two separate lobes in the spectrum $\hat{X}(\omega)$. Thus our ability to resolve spectral lines of different frequencies is limited by the window main lobe width. Figure 7.4.2 illustrates the magnitude spectrum $|\hat{X}(\omega)|$, computed via the DFT, for the sequence

$$x(n) = \cos \omega_0 n + \cos \omega_1 n + \cos \omega_2 n \tag{7.4.8}$$

where $\omega_0 = 0.2\pi$, $\omega_1 = 0.22\pi$, and $\omega_2 = 0.6\pi$. The window lengths selected are $L = 25, 50$, and 100. Note that ω_0 and ω_1 are not resolvable for $L = 25$ and 50, but they are resolvable for $L = 100$.

To reduce leakage, we can select a data window $w(n)$ that has lower sidelobes in the frequency domain compared with the rectangular window. However, as we describe in more detail in Chapter 10, a reduction of the sidelobes in a window $W(\omega)$ is obtained at the expense of an increase in the width of the main lobe of $W(\omega)$ and hence a loss in resolution. To illustrate this point, let us consider the Hanning window, which is specified as

$$w(n) = \begin{cases} \frac{1}{2}(1 - \cos \frac{2\pi}{L-1} n), & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases} \tag{7.4.9}$$

Figure 7.4.3 shows $|\hat{X}(\omega)|$ given by (7.4.4) for the window of (7.4.9). Its sidelobes are significantly smaller than those of the rectangular window, but its main lobe is approximately twice as wide. Figure 7.4.4 shows the spectrum of the signal in (7.4.8), after it is windowed by the Hanning window, for $L = 50, 75$, and 100. The reduction of the sidelobes and the decrease in the resolution, compared with the rectangular window, is clearly evident.

For a general signal sequence $\{x(n)\}$, the frequency-domain relationship between the windowed sequence $\hat{x}(n)$ and the original sequence $x(n)$ is given by the convolution formula

$$\hat{X}(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega$$

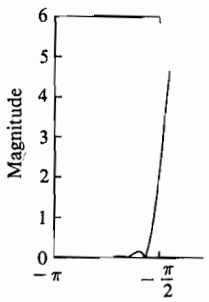


Figure 7.4.3
Magnitude spectrum of the Hanning window.

is sampled at th of the

EXA
The exp
$- F$

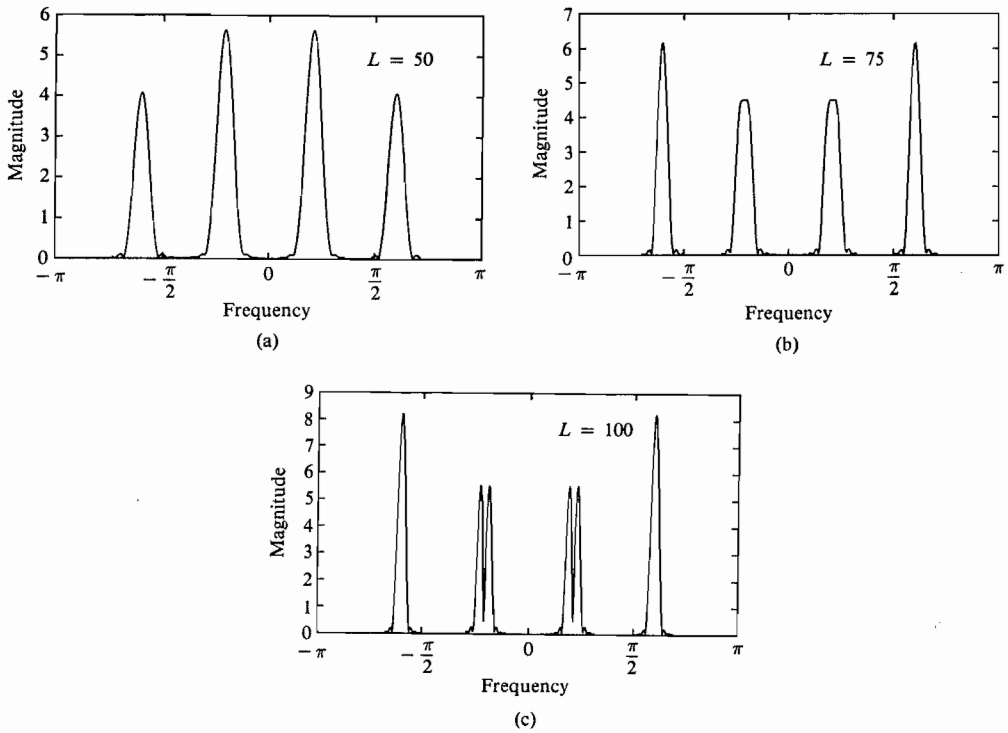


Figure 7.4.4 Magnitude spectrum of the signal in (7.4.8) as observed through a Hanning window.

The DFT of the windowed sequence $\hat{x}(n)$ is the sampled version of the spectrum $\hat{X}(\omega)$. Thus we have

$$\begin{aligned}\hat{X}(k) &\equiv \hat{X}(\omega)|_{\omega=2\pi k/N} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\theta)W\left(\frac{2\pi k}{N}-\theta\right)d\theta, \qquad k=0,1,\ldots,N-1\end{aligned} \tag{7.4.11}$$

Just as in the case of the sinusoidal sequence, if the spectrum of the window is relatively narrow in width compared to the spectrum $X(\omega)$ of the signal, the window function has only a small (smoothing) effect on the spectrum $X(\omega)$. On the other hand, if the window function has a wide spectrum compared to the width of $X(\omega)$, as would be the case when the number of samples L is small, the window spectrum masks the signal spectrum and, consequently, the DFT of the data reflects the spectral characteristics of the window function. Of course, this situation should be avoided.

EXAMPLE 7.4.1

The exponential signal

$$x_a(t) = \begin{cases} e^{-t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

is sampled at the rate $F_s = 20$ samples per second, and a block of 100 samples is used to

estimate its spectrum. Determine the spectral characteristics of the signal $x_a(t)$ by computing the DFT of the finite-duration sequence. Compare the spectrum of the truncated discrete-time signal to the spectrum of the analog signal.

Solution. The spectrum of the analog signal is

$$X_a(F) = \frac{1}{1 + j2\pi F}$$

The exponential analog signal sampled at the rate of 20 samples per second yields the sequence

$$\begin{aligned} x(n) &= e^{-nT} = e^{-n/20}, \qquad n \geq 0 \\ &= (e^{-1/20})^n = (0.95)^n, \quad n \geq 0 \end{aligned}$$

Now, let

$$x(n) = \begin{cases} (0.95)^n, & 0 \leq n \leq 99 \\ 0, & \text{otherwise} \end{cases}$$

The N-point DFT of the $L = 100$ point sequence is

$$\hat{X}(k) = \sum_{k=0}^{99} \hat{x}(n) e^{-j2\pi k/N}, \qquad k = 0, 1, \ldots, N-1$$

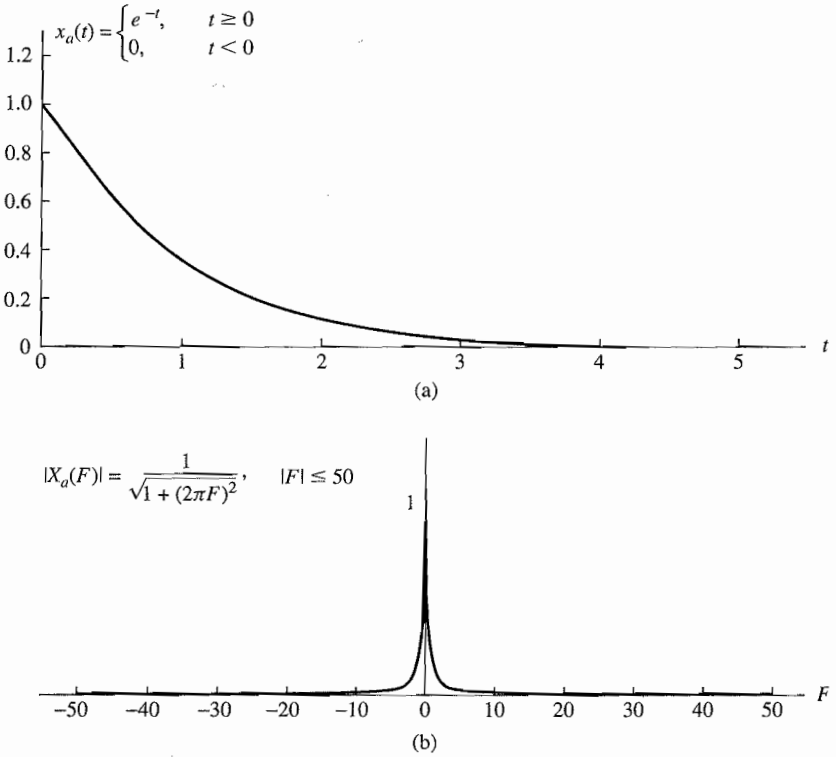


Figure 7.4.5 Effect of windowing (truncating) the sampled version of the analog signal in Example 7.4.1

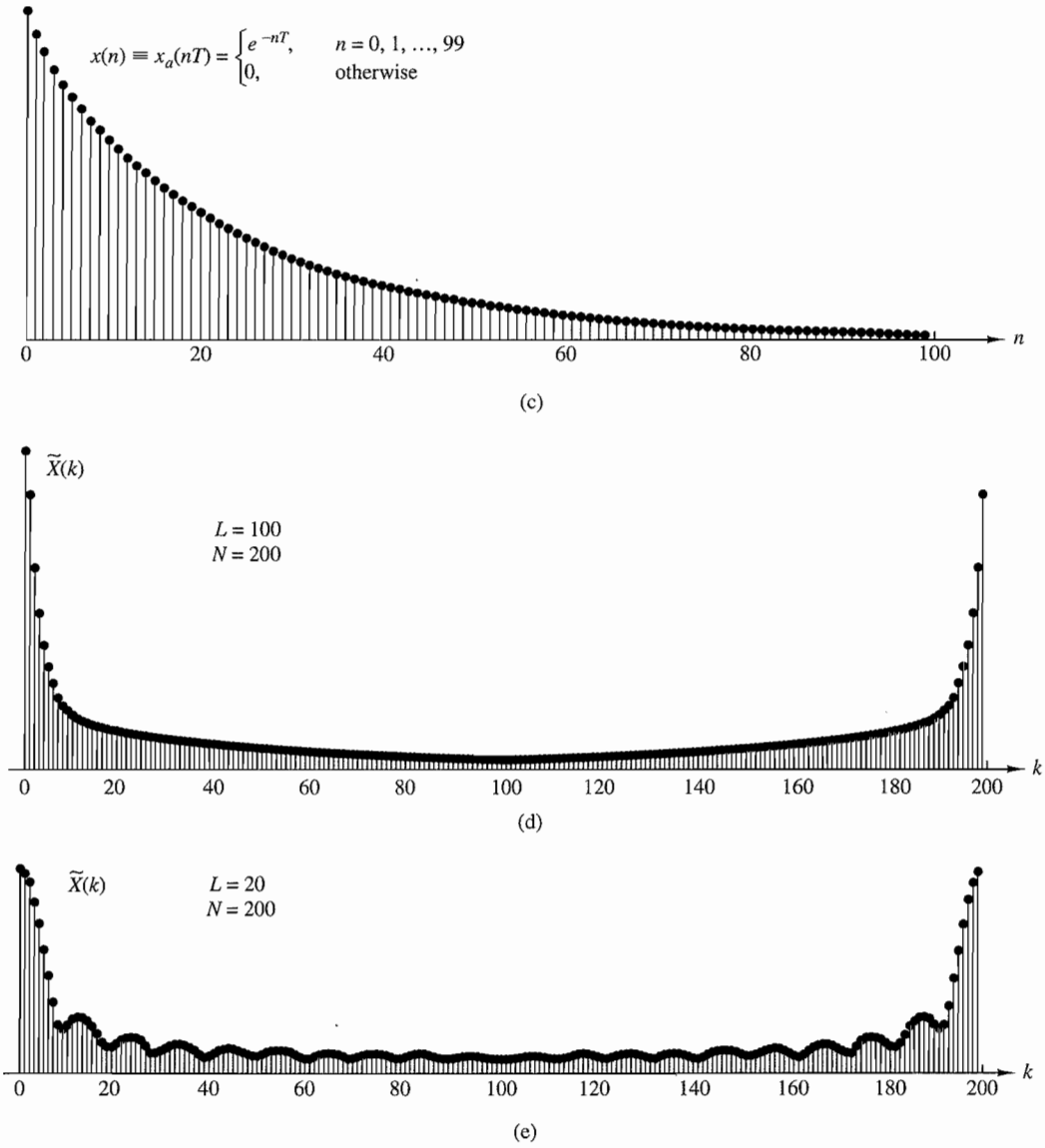


Figure 7.4.5 *Continued*

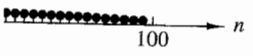
To obtain sufficient detail in the spectrum we choose $N = 200$. This is equivalent to padding the sequence $x(n)$ with 100 zeros.

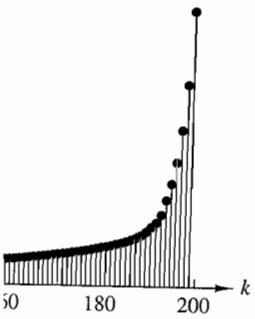
The graph of the analog signal $x_a(t)$ and its magnitude spectrum $|X_a(F)|$ are illustrated in Fig. 7.4.5(a) and 7.4.5(b), respectively. The truncated sequence $x(n)$ and its $N = 200$ point DFT (magnitude) are illustrated in Fig. 7.4.5(c) and 7.4.5(d), respectively. In this case the DFT $\{X(k)\}$ bears a close resemblance to the spectrum of the analog signal. The effect of the window function is relatively small.

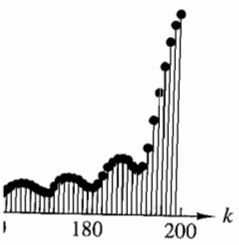
On the other hand, suppose that a window function of length $L = 20$ is selected. Then the truncated sequence $x(n)$ is given as

$$\hat{x}(n) = \begin{cases} (0.95)^n, & 0 \le n \le 19 \\ 0, & \text{otherwise} \end{cases}$$

Its $N = 200$-point DFT is illustrated in Fig. 7.4.5(e). Now the effect of the wider spectral window function is clearly evident. First, the main peak is very wide as a result of the wide spectral window. Second, the sinusoidal envelope variations in the spectrum away from the main peak are due to the large sidelobes of the rectangular window spectrum. Consequently, the DFT is no longer a good approximation of the analog signal spectrum.







is equivalent to padding

$|X_a(F)|$ are illustrated
) and its $N = 200$ point
tively. In this case the
signal. The effect of the

7.5 The Discrete Cosine Transform

The DFT represents an N-point sequence $x(n)$, $0 \le n \le N-1$, as a linear combination of complex exponentials. As a result, the DFT coefficients are, in general, complex even if $x(n)$ is real. Suppose that we wish to find an $N \times N$ orthogonal transform that expresses a real sequence $x(n)$ as a linear combination of cosine sequences. From (7.2.25) and (7.2.26), we see that this is possible if the N-point sequence $x(n)$ is real and even, that is, $x(n) = x(N-n)$, $0 \le n \le N-1$. The resulting DFT, $X(k)$, is itself real and even. This observation suggests that we could possibly derive a discrete cosine transform for any N-point real sequence by taking the $2N$-point DFT of an "even extension" of the sequence. Because there are eight ways to do this even extension, there are as many definitions of the DCT (Wang 1984, Martucci 1994). We discuss a version known as DCT-II, which is widely used in practice for speech and image compression applications as part of various standards (Rao and Huang, 1996). For simplicity, we will use the term DCT to refer to DCT-II.

7.5.1 Forward DCT

Let $s(n)$ be a $2N$-point even symmetric extension of $x(n)$ defined by

$$s(n) = \begin{cases} x(n), & 0 \le n \le N-1 \\ x(2N-n-1), & N \le n \le 2N-1 \end{cases} \tag{7.5.1}$$

The sequence $s(n)$ has even symmetry about the "half-sample" point $n = N + (1/2)$ (see Figure 7.5.1). The $2N$-point DFT of $s(n)$ is given by

$$S(k) = \sum_{n=0}^{2N-1} s(n) W_{2N}^{nk}, \quad 0 \le k \le 2N-1 \tag{7.5.2}$$

Substitution of (7.5.1) in (7.5.2) yields

$$S(k) = \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n=N}^{2N-1} x(2N-n-1) W_{2N}^{nk} \tag{7.5.3}$$

8.2 Applications of FFT Algorithms

The FFT algorithms described in the preceding section find application in a variety of areas, including linear filtering, correlation, and spectrum analysis. Basically, the FFT algorithm is used as an efficient means to compute the DFT and the IDFT.

In this section we consider the use of the FFT algorithm in linear filtering and in the computation of the crosscorrelation of two sequences. The use of the FFT in spectrum estimation is considered in Chapter 14. In addition we illustrate how to enhance the efficiency of the FFT algorithm by forming complex-valued sequences from real-valued sequences prior to the computation of the DFT.

8.2.1 Efficient Computation of the DFT of Two Real Sequences

The FFT algorithm is designed to perform complex multiplications and additions, even though the input data may be real valued. The basic reason for this situation is that the phase factors are complex and hence, after the first stage of the algorithm, all variables are basically complex valued.

In view of the fact that the algorithm can handle complex-valued input sequences, we can exploit this capability in the computation of the DFT of two real-valued sequences.

Suppose that $x_1(n)$ and $x_2(n)$ are two real-valued sequences of length N, and let $x(n)$ be a complex-valued sequence defined as

$$x(n) = x_1(n) + jx_2(n), \qquad 0 \le n \le N-1 \tag{8.2.1}$$

The DFT operation is linear and hence the DFT of $x(n)$ can be expressed as

$$X(k) = X_1(k) + jX_2(k) \tag{8.2.2}$$

The sequences $x_1(n)$ and $x_2(n)$ can be expressed in terms of $x(n)$ as follows:

$$x_1(n) = \frac{x(n) + x^*(n)}{2} \tag{8.2.3}$$

$$x_2(n) = \frac{x(n) - x^*(n)}{2j} \tag{8.2.4}$$

Hence the DFTs of $x_1(n)$ and $x_2(n)$ are

$$X_1(k) = \frac{1}{2}\{\text{DFT}[x(n)] + \text{DFT}[x^*(n)]\} \tag{8.2.5}$$

$$X_2(k) = \frac{1}{2j}\{\text{DFT}[x(n)] - \text{DFT}[x^*(n)]\} \tag{8.2.6}$$

Recall that the DFT of $x^*(n)$ is $X^*(N-k)$. Therefore,

$$X_1(k) = \frac{1}{2}[X(k) + X^*(N-k)] \tag{8.2.7}$$

$$X_2(k) = \frac{1}{j2}[X(k) - X^*(N-k)] \tag{8.2.8}$$

Thus, by performing a single DFT on the complex-valued sequence $x(n)$, we have obtained the DFT of the two real sequences with only a small amount of additional computation that is involved in computing $X_1(k)$ and $X_2(k)$ from $X(k)$ by use of (8.2.7) and (8.2.8).

8.2.2 Efficient Computation of the DFT of a $2N$-Point Real Sequence

Suppose that $g(n)$ is a real-valued sequence of $2N$ points. We now demonstrate how to obtain the $2N$-point DFT of $g(n)$ from computation of one N-point DFT involving complex-valued data. First, we define

$$\begin{aligned} x_1(n) &= g(2n) \\ x_2(n) &= g(2n+1) \end{aligned} \tag{8.2.9}$$

Thus we have subdivided the $2N$-point real sequence into two N-point real sequences. Now we can apply the method described in the preceding section.

Let $x(n)$ be the N-point complex-valued sequence

$$x(n) = x_1(n) + jx_2(n) \tag{8.2.10}$$

From the results of the preceding section, we have

$$\begin{aligned} X_1(k) &= \frac{1}{2}[X(k) + X^*(N-k)] \\ X_2(k) &= \frac{1}{2j}[X(k) - X^*(N-k)] \end{aligned} \tag{8.2.11}$$

Finally, we must express the $2N$-point DFT in terms of the two N-point DFTs, $X_1(k)$ and $X_2(k)$. To accomplish this, we proceed as in the decimation-in-time FFT algorithm, namely,

$$\begin{aligned} G(k) &= \sum_{n=0}^{N-1} g(2n) W_{2N}^{2nk} + \sum_{n=0}^{N-1} g(2n+1) W_{2N}^{(2n+1)k} \\ &= \sum_{n=0}^{N-1} x_1(n) W_N^{nk} + W_{2N}^k \sum_{n=0}^{N-1} x_2(n) W_N^{nk} \end{aligned}$$

Consequently,

$$\begin{aligned} G(k) &= X_1(k) + W_2^k N X_2(k), \qquad k = 0, 1, \ldots, N-1 \\ G(k+N) &= X_1(k) - W_2^k N X_2(k), \qquad k = 0, 1, \ldots, N-1 \end{aligned} \tag{8.2.12}$$

Thus we have computed the DFT of a $2N$-point real sequence from one N-point DFT and some additional computation as indicated by (8.2.11) and (8.2.12).

8.2.3 Use of the FFT Algorithm in Linear Filtering and Correlation

An important application of the FFT algorithm is in FIR linear filtering of long data sequences. In Chapter 7 we described two methods, the overlap-add and the overlap-save methods for filtering a long data sequence with an FIR filter, based on the use of the DFT. In this section we consider the use of these two methods in conjunction with the FFT algorithm for computing the DFT and the IDFT.

Let $h(n)$, $0 \leq n \leq M-1$, be the unit sample response of the FIR filter and let $x(n)$ denote the input data sequence. The block size of the FFT algorithm is N, where $N = L + M - 1$ and L is the number of new data samples being processed by the filter. We assume that for any given value of M, the number L of data samples is selected so that N is a power of 2. For purposes of this discussion, we consider only radix-2 FFT algorithms.

The N-point DFT of $h(n)$, which is padded by $L-1$ zeros, is denoted as $H(k)$. This computation is performed once via the FFT and the resulting N complex numbers are stored. To be specific we assume that the decimation-in-frequency FFT algorithm is used to compute $H(k)$. This yields $H(k)$ in bit-reversed order, which is the way it is stored in memory.

In the overlap-save method, the first $M-1$ data points of each data block are the last $M-1$ data points of the previous data block. Each data block contains L new data points, such that $N = L + M - 1$. The N-point DFT of each data block is performed by the FFT algorithm. If the decimation-in-frequency algorithm is employed, the input data block requires no shuffling and the values of the DFT occur in bit-reversed order. Since this is exactly the order of $H(k)$, we can multiply the DFT of the data, say $X_m(k)$, with $H(k)$, and thus the result

$$Y_m(k) = H(k)X_m(k)$$

is also in bit-reversed order.

The inverse DFT (IDFT) can be computed by use of an FFT algorithm that takes the input in bit-reversed order and produces an output in normal order. Thus there is no need to shuffle any block of data in computing either the DFT or the IDFT.

If the overlap-add method is used to perform the linear filtering, the computational method using the FFT algorithm is basically the same. The only difference is that the N-point data blocks consist of L new data points and $M-1$ additional zeros. After the IDFT is computed for each data block, the N-point filtered blocks are overlapped as indicated in Section 7.3.2, and the $M-1$ overlapping data points between successive output records are added together.

Let us assess the computational complexity of the FFT method for linear filtering. For this purpose, the one-time computation of $H(k)$ is insignificant and can be ignored. Each FFT requires $(N/2)\log_2 N$ complex multiplications and $N \log_2 N$ additions. Since the FFT is performed twice, once for the DFT and once for the IDFT, the computational burden is $N \log_2 N$ complex multiplications and $2N \log_2 N$ additions. There are also N complex multiplications and $N-1$ additions required to compute $Y_m(k)$. Therefore, we have $(N \log_2 2N)/L$ complex multiplications per output data point and approximately $(2N \log_2 2N)/L$ additions per output data point.

The overlap-add method requires an incremental increase of $(M-1)/L$ in the number of additions.

By way of comparison, a direct-form realization of the FIR filter involves M real multiplications per output point if the filter is not linear phase, and $M/2$ if it is linear phase (symmetric). Also, the number of additions is $M-1$ per output point (see Sec. 10.2).

It is interesting to compare the efficiency of the FFT algorithm with the direct form realization of the FIR filter. Let us focus on the number of multiplications, which are more time consuming than additions. Suppose that $M = 128 = 2^7$ and $N = 2^\nu$. Then the number of complex multiplications per output point for an FFT size of $N = 2^\nu$ is

$$c(\nu) = \frac{N \log_2 2N}{L} = \frac{2^\nu(\nu+1)}{N-M+1}$$

$$\approx \frac{2^\nu(\nu+1)}{2^\nu - 2^7}$$

The values of $c(\nu)$ for different values of ν are given in Table 8.3. We observe that there is an optimum value of ν which minimizes $c(\nu)$. For the FIR filter of size $M = 128$, the optimum occurs at $\nu = 10$.

We should emphasize that $c(\nu)$ represents the number of complex multiplications for the FFT-based method. The number of real multiplications is four times this number. However, even if the FIR filter has linear phase (see Sec. 10.2), the number of computations per output point is still less with the FFT-based method. Furthermore, the efficiency of the FFT method can be improved by computing the DFT of two successive data blocks simultaneously, according to the method just described. Consequently, the FFT-based method is indeed superior from a computational point of view when the filter length is relatively large.

The computation of the cross correlation between two sequences by means of the FFT algorithm is similar to the linear FIR filtering problem just described. In practical applications involving crosscorrelation, at least one of the sequences has finite duration and is akin to the impulse response of the FIR filter. The second sequence may be a long sequence which contains the desired sequence corrupted by additive noise. Hence the second sequence is akin to the input to the FIR filter.

TABLE 8.3 Computational Complexity

Size of FFT $\nu = \log_2 N$	$c(\nu)$ Number of Complex Multiplications per Output Point
9	13.3
10	12.6
11	12.8
12	13.4
14	15.1