Chapter 10
DSP in Communications

10.1 Introduction

What we have learnt so far is how to convert an analog signal into a digital signal and to process it using digital filters. The field of digital signal processing has fully matured and has found applications in diverse fields. In this chapter, we will concentrate on the application of DSP in one particular field, namely, the field of digital communications. Radio, telephony, and video are a few of the areas that are completely enveloped by modern digital and wireless communications. Radio broadcast started with analog communications. It used analog modulation techniques such as amplitude modulation (AM) and frequency modulation (FM) to transmit the message signal using radio frequencies (RF). These modulation schemes use the message signal to modulate a carrier signal in its amplitude (AM) or in its instantaneous frequency (FM) before transmission. Later digital modulation methods were introduced to serve the same purpose as the analog counterparts. Digital modulation plays an important role in modern wireless communications. The art of making very large-scale integrated (VLSI) circuits has evolved tremendously. This has enabled the design and fabrication of application-specific integrated circuits (ASIC), which in turn enables the implementation of complex digital techniques in achieving communications successfully as well as lowering the cost. Digital communication systems have many advantages over the analog counterparts. For instance, digital communications has greater immunity to noise. It is also robust to channel impairments. Another advantage is that many different data can be multiplexed and transmitted over a single channel. The various data may include voice, video, and other data, for instance. Since binary digits (bits) are used in digital communications, there is greater security in the transmitted data.

Electronic supplementary material: The online version of this article (https://doi.org/10.1007/978-3-319-76029-2_10) contains supplementary material, which is available to authorized users.

K. S. Thyagarajan, *Introduction to Digital Signal Processing Using MATLAB with Application to Digital Communications*,
https://doi.org/10.1007/978-3-319-76029-2_10

This is not feasible in analog communications. Even if there are errors in the received data, they can be detected and corrected by employing what is known as the *channel coding*, in which extra bits are added to the data bits. In analog communications, the noise in the channel will distort the message signal and is, therefore, impossible to recover the original signal. Even though digital modulation as such occupies a higher bandwidth than analog modulation, *source coding* or data compression is used to reduce the message bandwidth to start with. Digital communication link performance can be improved by using encryption and channel equalization techniques. Moreover, *field programmable gate arrays* (FPGA) enable the implementation of digital modulation and demodulation functions purely in software. This has an enormous implication because many handheld devices can perform a variety of functions well in real time using software. These features are certainly a no-no in analog communications.

In this chapter, we will describe a few DSP methods that are used in digital communications. More specifically, we will describe how digital pulses can be shaped at the transmitting side using DSP techniques to cancel what is known as the inter-symbol interference (ISI). At the receiver, another DSP function, namely, *equalization*, is used to mitigate the channel interference. In digital modulation, the receiver has to detect in each bit interval whether a binary "1" or a binary "0" is transmitted. This is achieved in an optimal fashion using the so-called *matched filter* (MF) or equivalently a *correlation filter*. We will learn how to implement such filters as well. We will also learn a few other DSP functions as applied to oversampled ADC and DAC, digital modulation schemes, and *phase-locked loop* (PLL).

10.2 Sampling Rate Conversion

Before the transmission of a message signal such as voice or music using digital modulation, the analog signal must be converted to a digital signal. As we have seen earlier, the analog-to-digital conversion (ADC) is achieved by first sampling the continuous-time signal at a minimum of Nyquist rate and then converting the analog samples to digital numbers. Typical bit widths of an ADC are between 8 and 12 bits. In wireless communications, for instance, the channel bandwidth is an extremely precious thing. Therefore, the service providers do whatever it takes to reduce the data rate of every subscriber. The first thing to do here is to reduce the bit width of the ADC. A one-bit ADC will be super. How is that possible? It is possible by using a very high sampling rate. We will first learn how to change the sampling rate and then describe the various DSP methods that incorporate different sampling rates. This is what is called *multi-rate digital signal processing*. That is, a digital signal processing that involves different sampling rates is termed multi-rate digital signal processing. In multi-rate DSP, sampling rates at certain points are increased from its native rate. This is known as *upsampling*. At other points the sampling rate is decreased from its native rate. This process is termed downsampling. Upsampling or downsampling may use

either integer sampling rate or rational sampling rate. Before we discuss ADC with a sampling rate higher than the Nyquist rate, we need to know the effect of upsampling or downsampling on the sampled signal.

10.2.1 Upsampling

Let $\{x[n], -\infty < n < \infty\}$ be a sequence that is sampled at the Nyquist rate. If the original sampling rate is increased by a positive integer factor M, then we can express the upsampled sequence $x_u[n]$ in terms of the original sequence as described by

$$x_u[n] = \begin{cases} x[n/M], n = 0, \pm M, \pm 2M, \cdots \\ 0, otherwise \end{cases} \tag{10.1}$$

From the above equation, we find that upsampling amounts to inserting M-1 zeros between every two consecutive samples. To understand better the process of upsampling, we need to describe the upsampled sequence in the Z-domain. The Z-transform of the upsampled sequence is obtained using the definition of the Z-transform and is given by

$$X_u(z) = \sum_{n=-\infty}^{\infty} x_u[n] z^{-n} \tag{10.2}$$

In terms of the Z-transform of the original sequence, (10.2) reduces to

$$X_u[z] = \sum_{n=-\infty}^{\infty} x\left[\frac{n}{M}\right] z^{-n} \tag{10.3}$$

Define $m = \frac{n}{M}$ and (10.3) becomes

$$X_u[z] = \sum_{m=-\infty}^{\infty} x[m] z^{-mM} = \sum_{m=-\infty}^{\infty} x[m]\left(z^M\right)^{-m} = X\left(z^M\right) \tag{10.4}$$

The DTFT of the upsampled sequence can then be found by using $z = e^{j\Omega}$ in (10.4), which is

$$X_u\left(e^{j\Omega}\right) = X(z)|_{z=e^{j\Omega}} = X\left(e^{jM\Omega}\right) \tag{10.5}$$

The implication of (10.5) is that the spectrum of $x_u[n]$ is the same spectrum of $x[n]$ but repeated M times in the interval $[0, 2\pi]$. In other words, what happens to $x[n]$ in the frequency domain in the interval $[0, 2\pi]$ happens M times to the upsampled sequence in that interval. That is, there are M-1 images of the spectrum $X(e^{j\Omega})$ in the spectrum of the upsampled sequence in the interval $[0, 2\pi]$. In order to preserve the integrity of the original sequence, the upsampled sequence must be lowpass filtered to reject the M-1 images.

Example 10.1 In this example, we will consider a signal consisting of three sinusoids of frequencies 950 Hz, 1800 Hz, and 1917 Hz, which is sampled at a rate of 5000 Hz. This signal is then upsampled by a factor of M = 6. We have to compute the spectra of the original and upsampled sequences as well as the spectrum of the lowpass filtered upsampled signal to verify that the images are removed from the spectrum of the upsampled signal.

Solution The actual input signal before sampling is described by

$$x(t) = \sin(2\pi f_1 t) + 2\sin(2\pi f_2 t) + 1.75\sin(2\pi f_3 t), t \geq 0 \tag{10.6}$$

where the frequencies are $f_1 = 950Hz, f_2 = 1800Hz$, and $f_3 = 1917Hz$. In Fig. 10.1 is shown the stem plots of the input sequence, upsampled sequence, and upsampled and lowpass filtered sequence in the top, middle, and bottom plots, respectively. As seen in the middle plot, there are M-1 = 5 zeros between every two consecutive samples. The corresponding spectra are shown in Fig. 10.2. There are three frequencies present in the spectrum of the original signal. Since the plot is over the interval

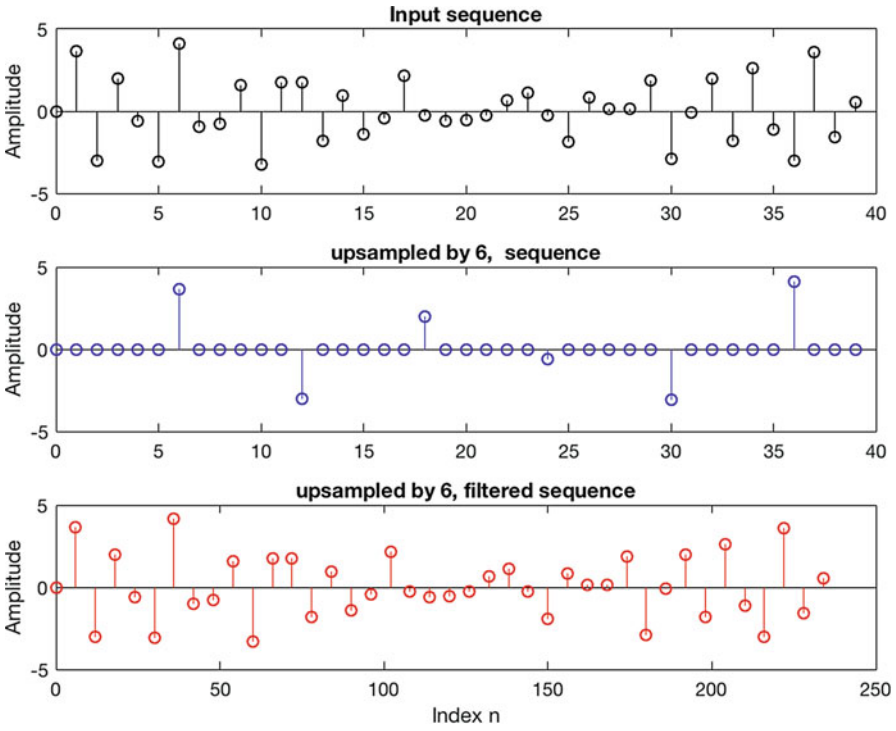

Fig. 10.1 Stem plots of the sequences: top, input sequence sampled as 5000 samples/sec; middle, upsampled sequence using an upsampling factor of 6; bottom, lowpass filtered upsampled sequence; every sixth sample is plotted

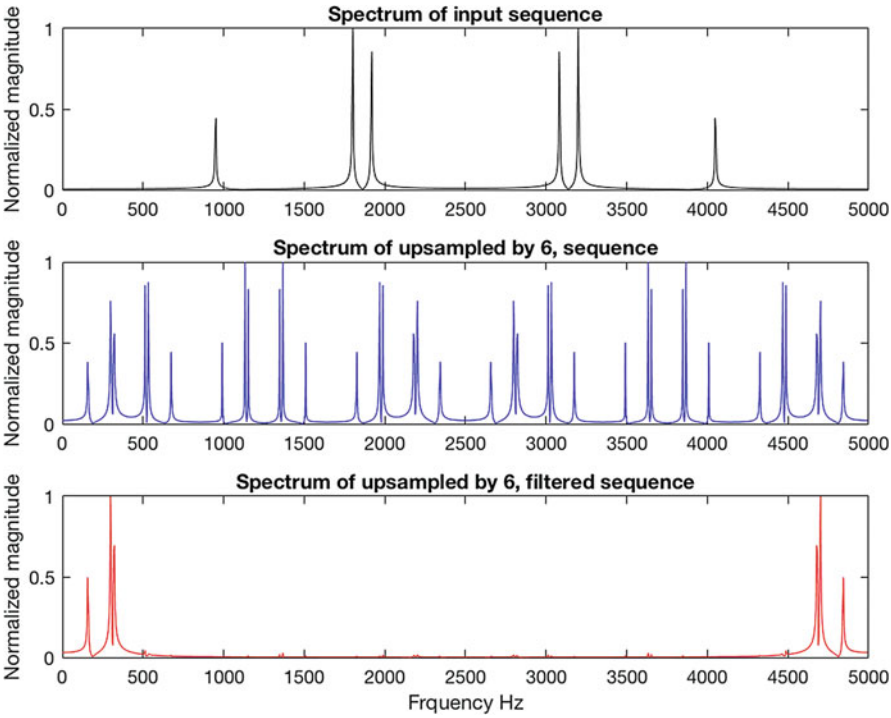

Fig. 10.2 Spectra of the sequences in Example 10.1: top, spectrum of input sequence; middle, spectrum of upsampled sequence; bottom, spectrum of lowpass filtered sequence

$[0, f_s]$, we see the mirror image of the three frequencies in the top plot. The spectrum of the upsampled sequence consists of M-1 = 5 images in the interval between zero and the sampling frequency. The bottom plot depicts the spectrum of the lowpass filtered sequence wherein the images have been removed. Since there are M-1 images in the interval between zero and the sampling frequency, the frequency is compressed. So, if Ω_c is the cutoff frequency of the original sequence, the same frequency will appear at $\frac{\Omega_c}{M}$. Therefore, to remove the images from the upsampled sequence, we should use a lowpass filter having a cutoff frequency $\frac{\Omega_c}{M}$. The MATLAB program used for this example is listed in the M-file named *Example 10_1.m*.

Upsampling Identity The upsampling process consists of first filtering the input sequence by a lowpass filter and then increasing the sampling rate by the positive integer factor M. It is equivalent to first upsampling the input sequence by the factor M followed by lowpass filtering. These two processes are shown in Fig. 10.3. Because of the identity of the two processes, one can use either one to realize upsampling of a sequence. In both cases, the image spectra are removed.

Fig. 10.3 Upsampling identity

10.2.2 Downsampling

Let us look at the process of reducing the sampling rate by a positive integer factor. The process of reducing the sampling rate is called *downsampling*. Consider a discrete-time sequence $\{x[n], -\infty < n < \infty\}$ which is assumed to be sampled at the Nyquist rate. If this sequence is downsampled by an integer factor D, we can then express the downsampled sequence $x_d[n]$ in terms of the original sequence as given by

$$x_d[n] = x[nD], D \in Z \tag{10.7}$$

To understand better the effect of downsampling on the sequence, we will have to describe the downsampled sequence in the frequency domain. The Z-transform of the downsampled sequence is obtained from the definition of Z-transform and is described by

$$X_d(z) = \sum_{n=-\infty}^{\infty} x_d[n] z^{-n} = \sum_{n=-\infty}^{\infty} x[nD] z^{-n} \tag{10.8}$$

Let

$$x'[n] = \begin{cases} x[n], n = 0, \pm D, \pm 2D, \cdots \\ 0, otherwise \end{cases} \tag{10.9}$$

In terms of the new sequence, the Z-transform of the downsampled sequence becomes

$$X_d(z) = \sum_{n=-\infty}^{\infty} x'[nD] z^{-n} \tag{10.10}$$

By using $m = nD$ in the above equation, we have

$$X_d(z) = \sum_{m=-\infty}^{\infty} x'[m] z^{-\frac{m}{D}} = \sum_{m=-\infty}^{\infty} x'[m] \left(z^{\frac{1}{D}}\right)^{-m} = X'\left(z^{\frac{1}{D}}\right) \tag{10.11}$$

We still haven't expressed the Z-transform of the downsampled sequence in terms of the Z-transform of the input sequence. In order to do that, let us define another function given in (10.12):

$$g[n] = \begin{cases} 1, n = 0, \pm D, \pm 2D, \cdots \\ 0, otherwise \end{cases} \tag{10.12}$$

The above sequence can be expressed as the inverse DTFT of the frequency points, $\left\{W_D^{kn}, 0 \le k \le D-1\right\}$, where, $W_D^k = e^{-j\frac{2\pi k}{D}}$, as given by

$$g[n] = \frac{1}{D}\sum_{k=0}^{D-1} W_D^{kn} \tag{10.13}$$

The sequence $x^{'}[n]$ can be expressed in terms of $x[n]$ and $g[n]$ as

$$x'[n] = g[n]x[n] \tag{10.14}$$

Now using (10.13) and (10.14) in (10.11), we find that

$$X'(z) = \sum_{n=-\infty}^{\infty} g[n]x[n]z^{-n} = \sum_{n=-\infty}^{\infty}\left(\frac{1}{D}\sum_{k=0}^{D-1} W_D^{nk}\right)x[n]z^{-n} \tag{10.15}$$

By interchanging the order of summation in (10.15), we have

$$X'(z) = \frac{1}{D}\sum_{k=0}^{D-1}\sum_{n=-\infty}^{\infty} x[n]\left(zW_D^{-k}\right)^{-n} = \frac{1}{D}\sum_{k=0}^{D-1} X\left(zW_D^{-k}\right) \tag{10.16}$$

Finally, using (10.16) in (10.11), we obtain the Z-transform of the downsampled sequence as

$$X_d(z) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(z^{\frac{1}{D}}W_D^{-k}\right) \tag{10.17}$$

From (10.17), the DTFT of the downsampled sequence is determined to be

$$X_d\left(e^{j\Omega}\right) = X_d(z)|_{z=e^{j\Omega}} = \frac{1}{D}\sum_{k=0}^{D-1} X\left(e^{j\left(\frac{\Omega+2\pi k}{D}\right)}\right) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(e^{j\left(\frac{\Omega-2\pi k}{D}\right)}\right) \tag{10.18}$$

From the above equation, we notice that the spectrum of the downsampled sequence is the sum of the shifted and stretched versions of the spectrum of the original sequence. Let us make it clearer by way of an example after we establish the identity of downsampling.

Downsampling Identity The process of downsampling a sequence by a positive integer factor D can be identified as first lowpass filtering the input sequence and then downsampling. This identity is equivalent to first downsampling followed by lowpass filtering. The two processes are shown in Fig. 10.4.

Example 10.2 Use the same sequence as in Example 10.1 and downsample it by a factor of D = 6. Compute the spectra of the original, downsampled, and filtered sequences and plot them.

Solution The downsampling and filtering operations are done using the MATLAB M-file named *Example 10_2.m*. The original, downsampled, and filtered sequences

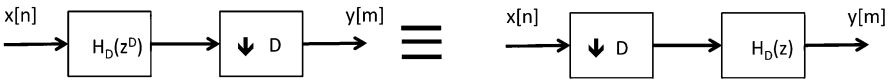

Fig. 10.4 Downsampling identity

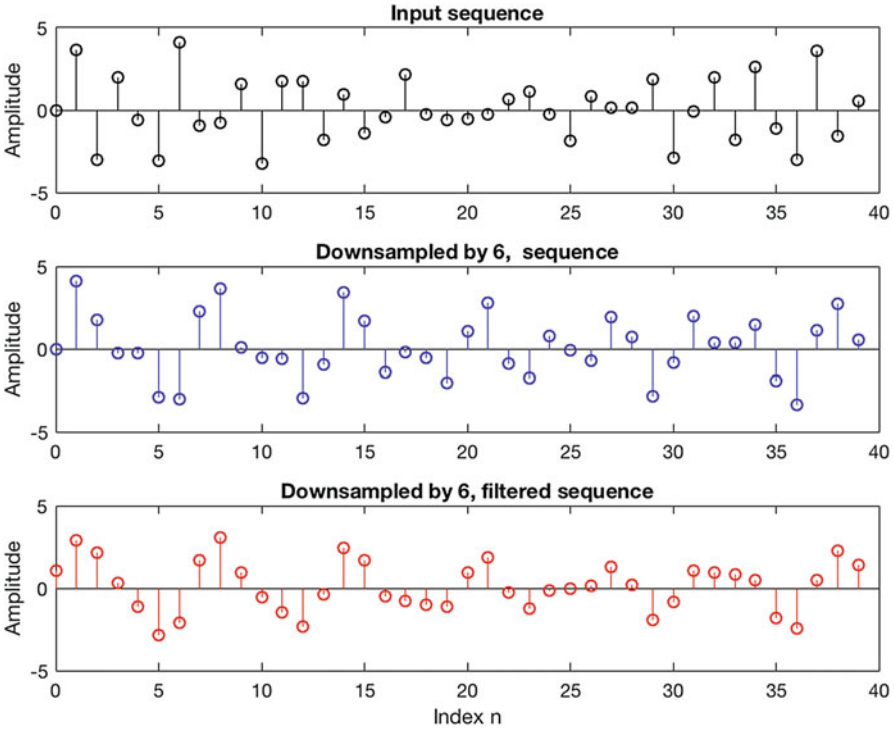

Fig. 10.5 Downsampling the sequence in Example 10.2: top, original sequence; middle, sequence downsampled by a factor of 6; bottom, lowpass filtered downsampled sequence

are shown in Fig. 10.5 in the top, middle, and bottom plots, respectively. We see from the middle plot, the sequence corresponds to every sixth sample of the input sequence. The corresponding spectra are shown in Fig. 10.6.

10.3 Oversampled ADC

As mentioned earlier, an ADC converts a continuous-time signal into a digital signal by first sampling the input signal uniformly at Nyquist rate and then quantizing the analog samples using a B-bit uniform quantizer. The typical bit width of an ADC is

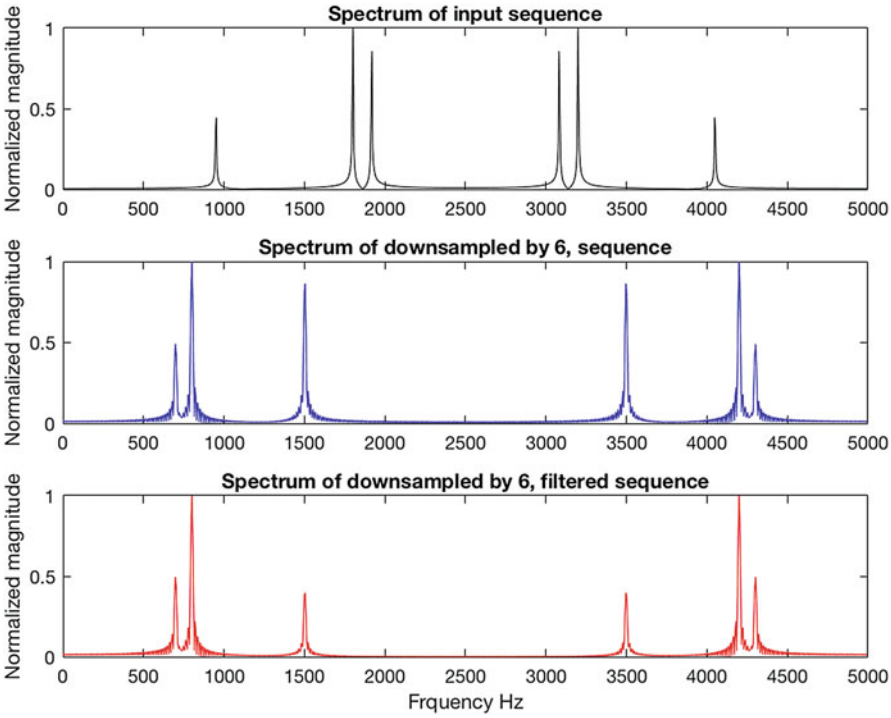

Fig. 10.6 Frequency spectra of the sequences in Fig. 10.5: top, spectrum of the original sequence; middle, spectrum of the downsampled sequence; bottom, spectrum of the downsampled sequence after lowpass filtering

in the range of 8–14 bits. Since the sampling process may introduce aliasing distortion, the input continuous-time signal is filtered by an antialiasing analog filter before sampling. Because of a narrower transition bandwidth, the required analog filter order will be very high. This makes the IC design more complex and it may also lead to instability. In order to ease the requirements on the antialiasing analog filter, one can increase the sampling rate much higher than the Nyquist rate. As we will see below, this will increase the transition bandwidth, thereby lowering the antialiasing filter order. As a result of lowering the filter order, the IC design becomes simpler and more compact. Of course, the output of the ADC must be downsampled to bring the sampling rate back to the Nyquist rate. Figure 10.7 shows the block diagram of an oversampled ADC. The input continuous-time signal is first filtered by an analog lowpass filter of a small order and then input to the ADC. The sampling rate of the ADC is assumed to be M times the Nyquist sampling rate. The output of the ADC is passed through an antialiasing digital filter before it is downsampled by the factor M. In what follows, we will give a brief analysis of the oversampled ADC.

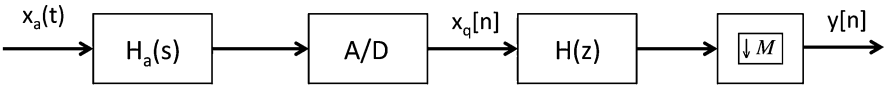

Fig. 10.7 Block diagram of an oversampled ADC

10.3.1 Transition Bandwidth Reduction

Let us assume the maximum frequency in the input continuous-time signal to be f_m. The corresponding Nyquist frequency is $2f_m$. If the actual sampling frequency F_s of the ADC is M times the Nyquist frequency, then $F_s = 2Mf_m$. This amounts to the frequency specification of the antialiasing analog filter as described by

$$H_a(f) = \begin{cases} 1, 0 \le |f| \le f_m \\ 0, Mf_m < |f| < \infty \end{cases} \tag{10.19}$$

The above equation reveals the fact that the passband edge frequency of the antialiasing filter is still the maximum frequency in the analog signal. However, the stopband edge frequency has moved to M times the maximum frequency. Therefore, the transition width, which is the difference between the stopband edge and passband edge, has increased as given by

$$\Delta f = (M-1)f_m \tag{10.20}$$

If the transition width is large, the filter order will be smaller. That's how the oversampling eases the filter order requirement.

10.3.2 Analysis of Oversampled ADC

Let us go a bit deeper and see the effect of oversampling on the ADC. Let the amplitude range of the input analog signal be

$$|x_a(t)| \le x_{max} \tag{10.21}$$

The corresponding quantization step size of a B-bit ADC is expressed as

$$q = \frac{x_{max} - x_{min}}{2^B} = \frac{2x_{max}}{2^B} = \frac{x_{max}}{2^{B-1}} \tag{10.22}$$

The noise due to quantization is uniformly distributed in the range $[-q/2, q/2]$, and its variance is determined to be

$$\sigma_q^2 = \frac{q^2}{12} = \frac{x_{max}^2}{12 \times 2^{2B-2}} = \frac{x_{max}^2}{3 \times 2^{2B}} \tag{10.23}$$

The output of the ADC is filtered by an antialiasing digital filter, followed by downsampling by the factor M. Let $y[n]$ be the downsampled signal. Since the downsampled signal has the same power as that of the signal before downsampling, we have

$$\sigma_y^2 = G\sigma_q^2 \tag{10.24}$$

where G is the power gain of the antialiasing digital filter, and it is related to the impulse response sequence of the digital filter. If the digital filter is an Nth-order FIR filter, then

$$G = \sum_{n=0}^{N} |h[n]|^2 \tag{10.25}$$

The power gain can also be obtained from the frequency domain since the power is conserved in both domains. Thus,

$$G = \frac{1}{F_s}\int_{-\frac{F_s}{2}}^{\frac{F_s}{2}} |H(f)|^2 df = \frac{1}{F_s}\int_{-\frac{F_s}{2M}}^{\frac{F_s}{2M}} df = \frac{1}{M} \tag{10.26}$$

Therefore, the variance of the noise due to quantization is reduced by the oversampling factor, which is

$$\sigma_y^2 = \frac{\sigma_q^2}{M} \tag{10.27}$$

This is really great! Not only does oversampling reduce the quantization noise, it also spreads it over the frequency range $\left[0, \frac{F_s}{2}\right]$. But the cutoff frequency of the antialiasing digital filter is $\frac{F_s}{2M}$. Therefore, it rejects the noise in the range $\left[\frac{F_s}{2M}, \frac{F_s}{2}\right]$. This reduction in the quantization noise is related to the reduction in the number of bits of the ADC as compared to the ADC that does not use oversampling. Let B_M and B denote the bit widths of the ADCs with and without oversampling. Since the quantization noise power must be the same in both ADCs for the sake of comparison, the output noise powers can be expressed as

$$\frac{\sigma_q^2}{M} = \frac{x_{max}^2}{3 \times M \times 2^{2B_M}} = \frac{x_{max}^2}{3 \times 2^{2B}} \tag{10.28}$$

From (10.28) we have

$$M \times 2^{2B_M} = 2^{2B} \tag{10.29a}$$

$$B_M = B - \frac{1}{2} log_2 M \tag{10.29b}$$

Thus, for the same given quantization noise power, an oversampled ADC needs a bit width that is less than that required by the Nyquist sampled ADC by one half times the logarithm to the base 2 of the oversampling factor.

Oversampling Factor Versus the Filter Order We can relate the oversampling factor M to the filter order for a specific filter as follows. Consider a kth-order Butterworth antialiasing analog filter with a cutoff frequency f_m. The corresponding magnitude of the frequency response is given by

$$|H_a(f)| = \frac{1}{\sqrt{1 + \left(\frac{f}{f_m}\right)^{2k}}} \tag{10.30}$$

The maximum error δ due to aliasing distortion will occur at the *folding* frequency $\frac{F_s}{2}$ and is equal to

$$\delta = \left|H_a\left(\frac{F_s}{2}\right)\right| = \frac{1}{\sqrt{1 + \left(\frac{Mf_m}{f_m}\right)^{2k}}} = \frac{1}{\sqrt{1 + M^{2k}}} \tag{10.31}$$

Then,

$$\frac{1}{\sqrt{1 + M^{2k}}} \leq \delta \tag{10.32}$$

Or

$$1 + M^{2k} \geq \delta^{-2} \Rightarrow M \geq \left(\delta^{-2} - 1\right)^{\frac{1}{2k}} \tag{10.33}$$

For instance, if $\delta = 0.005$ and $k = 3$, then from (10.33), we find that the oversampling factor $M = \lceil 5.848 \rceil = 6$. If the filter order is reduced to 2, then for the same maximum aliasing distortion, the oversampling ratio hikes up to 15. A decrease by one in the filter order increases the oversampling ratio almost by a factor of 3! The maximum aliasing error in dB can be expressed in terms of the oversampling factor and the Butterworth filter order using (10.33) and is given by

$$\delta = -10log_{10}\left(1 + M^{2k}\right) \ dB \tag{10.34}$$

Figure 10.8 shows the maximum aliasing error in dB against the oversampling factor for four values of the Butterworth filter order. As seen from the figure, the oversampling factor increases as the Butterworth filter order decreases for a fixed value of the maximum aliasing error.

Example 10.3 Let us work out an example using MATLAB to digest what we learnt about oversampled ADC. We will use the same input signal that was used in the previous example. The Nyquist sampling rate is 4000 Hz and the oversampling factor used is 15. We will use a 5-bit ADC to quantize the analog samples. The antialiasing digital filter is a fourth-order FIR filter with a normalized cutoff

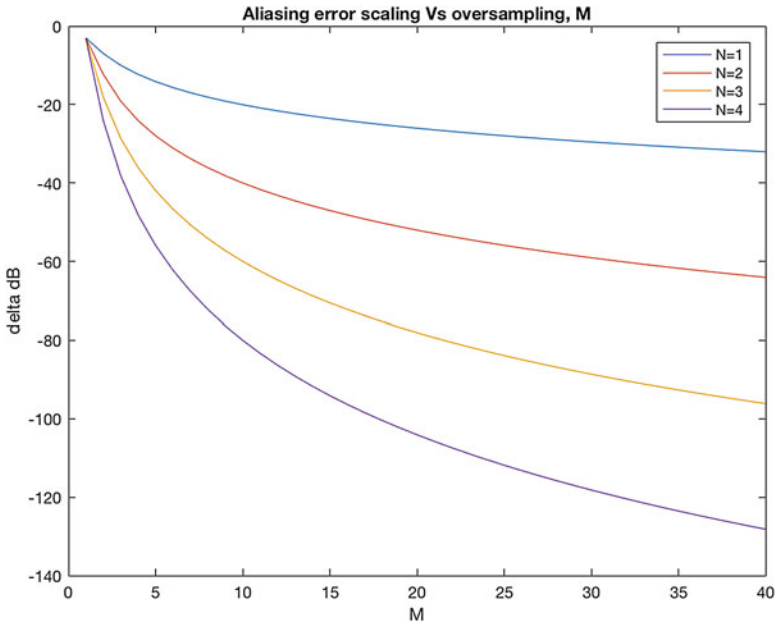

Fig. 10.8 Oversampling factor versus the maximum aliasing error in dB for four different values of the Butterworth analog filter order

frequency of 1/M. The M-file to solve this problem is named *Example 10_3.m*. The program designs a 5-bit uniform quantizer before quantizing the input signal. Figure 10.9 depicts the input-output relationship of the quantizer as a step function. The horizontal axis refers to decision regions and the vertical axis to the reconstruction levels. The actual and 5-bit quantized input sequence is shown in Fig. 10.10 as a stem plot. It appears that the 5-bit quantized version very nearly approximates the full-precision signal. The downsampled versions of the full-precision and 5-bit quantized version of the input signal are shown in Fig. 10.11 in the top and bottom plots, respectively. In Fig. 10.12 is shown the spectra of the input sequence, filtered output, and filtered downsampled output sequences in the top, middle, and bottom plots, respectively. As seen from the figure, the folding frequency of the downsampled signal reverts to 2000 Hz. As pointed out, the quantization noise of the ADC is uniformly distributed in the range $\pm\frac{q}{2}$. Figure 10.13 shows the 5-bit quantizer noise sequence along with its histogram and spectrum in the top, middle, and bottom plots, respectively. As expected, the histogram appears uniform implying that the quantizer noise is uniformly distributed. The program also computes the noise variances of the quantizer before and after filtering by the antialiasing digital filter. The noise variance or power of the 5-bit quantizer is found to be 0.0081. The corresponding SNR is 26.96 dB. The noise variance and the resulting SNR after filtering are, respectively, 0.0037 and 30.33 dB. Because of oversampling, the quantization noise power at the output of the antialiasing digital filter has decreased by more than 3 dB!

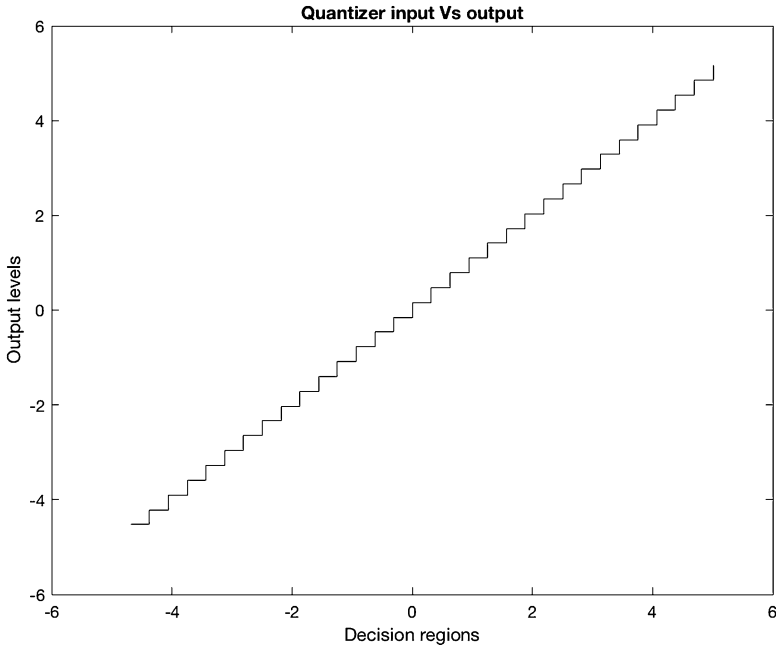

Fig. 10.9 Input/output relationship of a 5-bit uniform quantizer of Example 10.3 as a step function

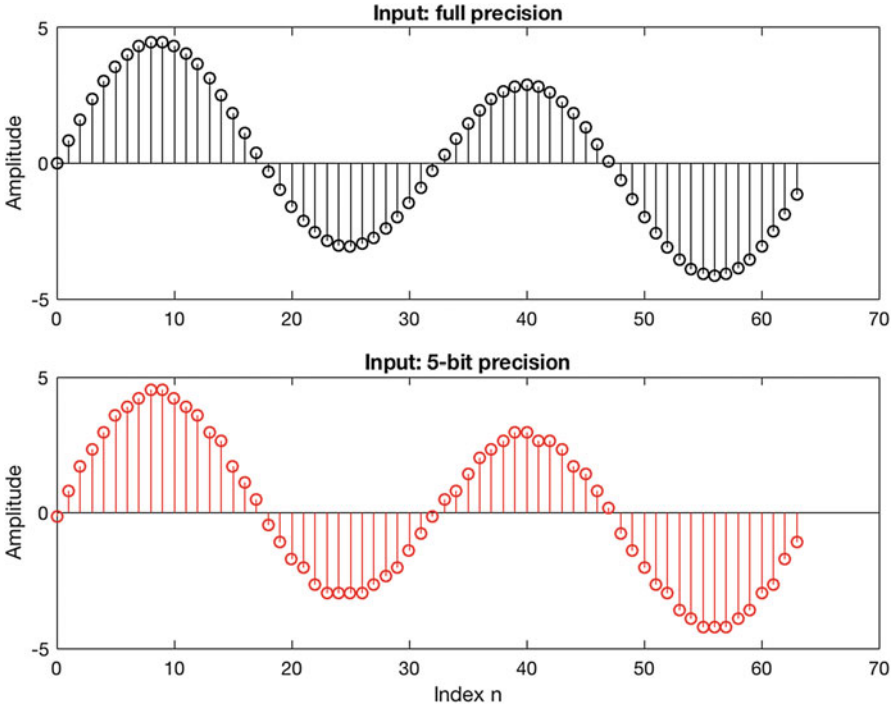

Fig. 10.10 Full-precision and 5-bit quantized input sequence: top, full-precision; bottom, 5-bit quantized sequence

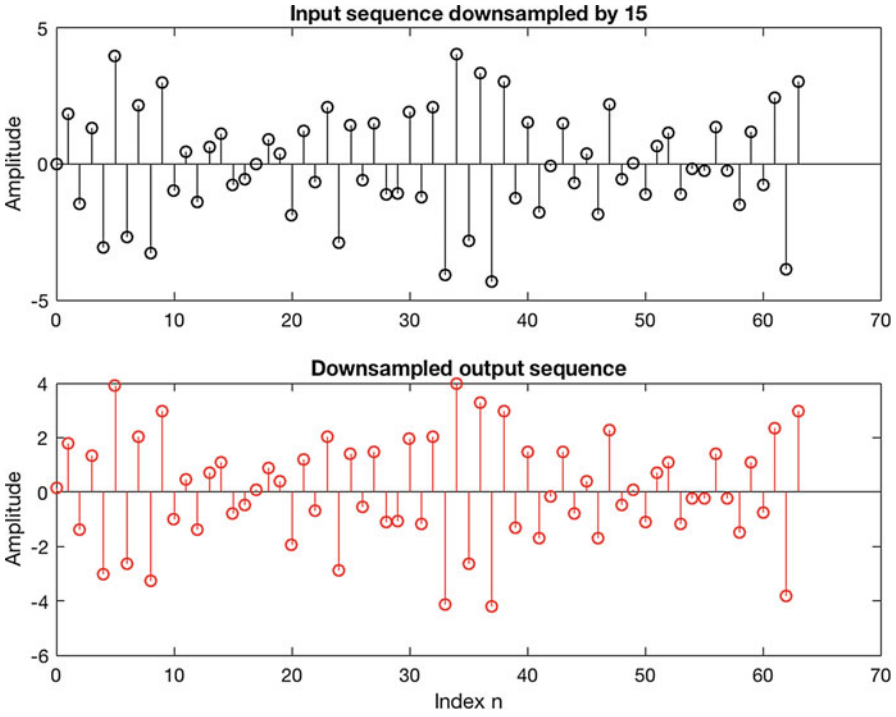

Fig. 10.11 Downsampled sequences: top, full-precision sequence; bottom: 5-bit quantized sequence

10.4 Oversampled DAC

In the previous section, we described an oversampled ADC. The purpose of using a sampling rate much higher than the Nyquist rate is to ease the requirements on the antialiasing analog prefilter by increasing the transition width. Oversampling also reduces the ADC bit width and rejects the out-of-band noise. After transmission and reception of the digital signal, it must be converted back to the analog domain. Since the sampling rate at the ADC is high, it has to be brought back to the Nyquist rate. A block diagram of an oversampled DAC is shown in Fig. 10.14. Since the process of downsampling introduces spectral images, the images must be removed using an anti-imaging lowpass analog filter. The output of the anti-imaging lowpass analog filter is the recovered analog signal. Let us exemplify the oversampled DAC operation by an example using MATLAB.

Example 10.4 This example uses the same signal used in the previous example. To be self-contained, the oversampled ADC is incorporated. Its output is then converted to an analog signal. The anti-imaging lowpass analog filter is incorporated into the DAC. The M-file to execute Example 10.4 is named *Example 10_4.m*. The input to

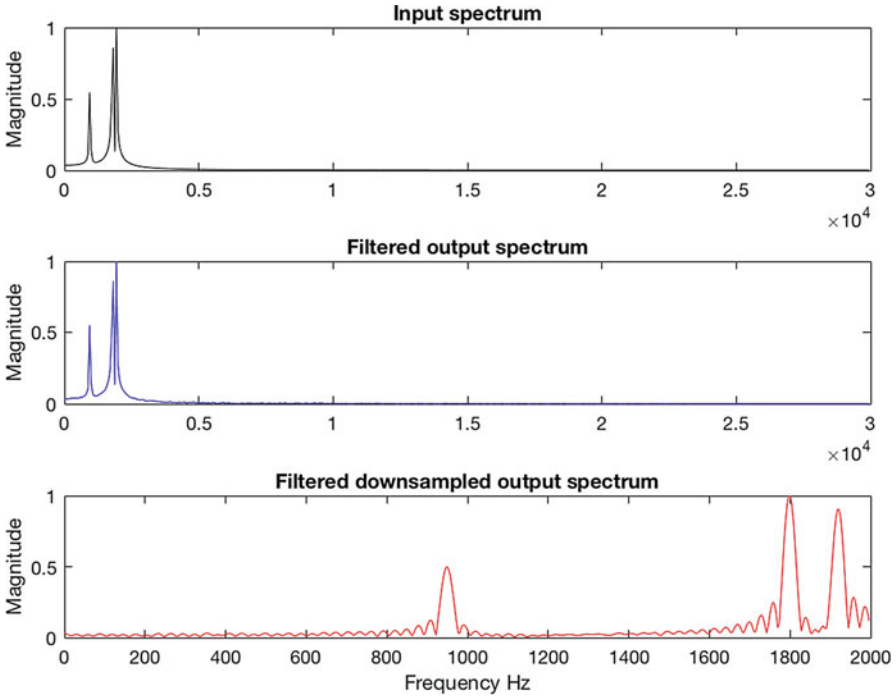

Fig. 10.12 Spectra of the sequences in Example 10.3: top, input sequence; middle, filtered output sequences; bottom, filtered and downsampled sequence

the DAC is a 5-bit quantized sequence that is oversampled by a factor of 6. The anti-imaging filter is a second-order Butterworth filter. The input analog signal and the DAC output signal are shown in Fig. 10.15 in the top and bottom plots, respectively. They seem to be pretty close. The corresponding spectra are shown in Fig. 10.16. Because of quantization, there appears some noise in the output spectrum. The SNR between the input analog signal and the DAC output signal is found to be 25.36 dB. In both oversampled ADC and DAC, DSP plays an important role in filtering the sequences using digital filters.

10.5 Cancelation of Inter-Symbol Interference

In a digital communications system, messages are represented in binary format. Each binary symbol is transmitted as a pulse. These pulses are first lowpass filtered at the transmitter before transmission to confine them to a certain specified bandwidth. As these pulses travel through the channel, they are distorted in their amplitude and phase due to the channel reactance. As a result, the pulses expand in time and so

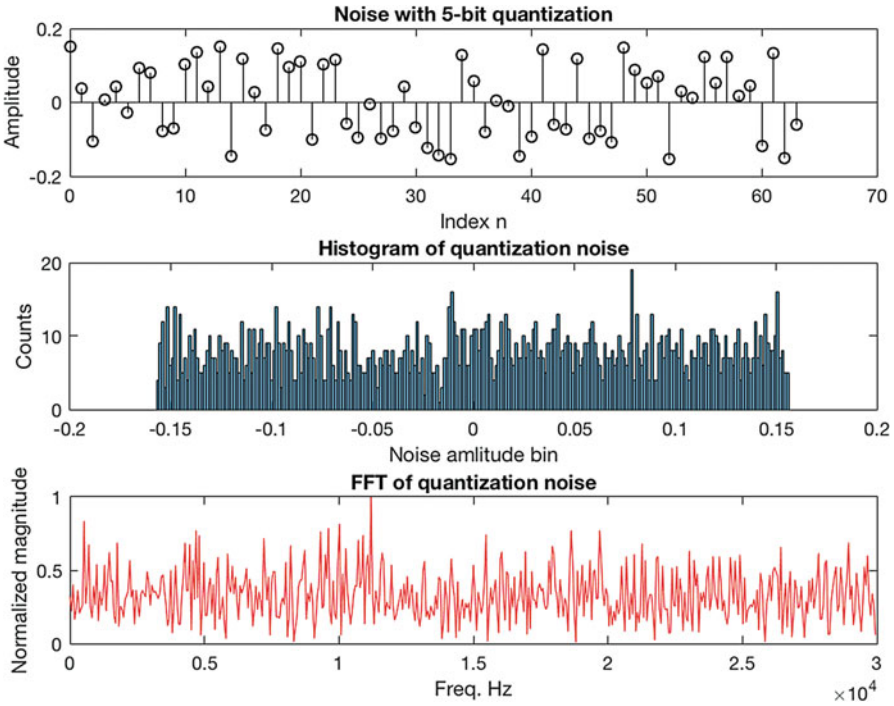

Fig. 10.13 5-bit quantizer noise: top, noise sequence; middle, histogram of the quantizer noise; bottom, spectrum of the quantizer noise

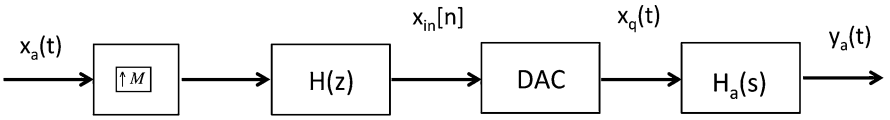

Fig. 10.14 Block diagram of an oversampled DAC

overlap between neighboring pulses. The heart of digital communications is in its precise timing. If pulses from neighboring bit intervals overlap, then error occurs in the detection of the actual transmitted pulse in each bit interval. That is to say that interference between pulses occurs due to distortion in the transmitted pulses. This is known as the *inter-symbol interference* (ISI). The transmitter/channel/receiver chain can be modeled as a cascade of three LTI systems as described by

$$H(f) = H_t(f)H_c(f)H_r(f) \tag{10.35}$$

where $H_t(f)$ represents the transmitter, $H_c(f)$ the channel, and $H_r(f)$ the receiver. The ISI can be canceled or minimized by adjusting the transmitter and receiver filters. Since the channel is not under our control, we cannot do anything with it. By a proper choice of the transmitter, we can shape the transmitted pulses in such a way

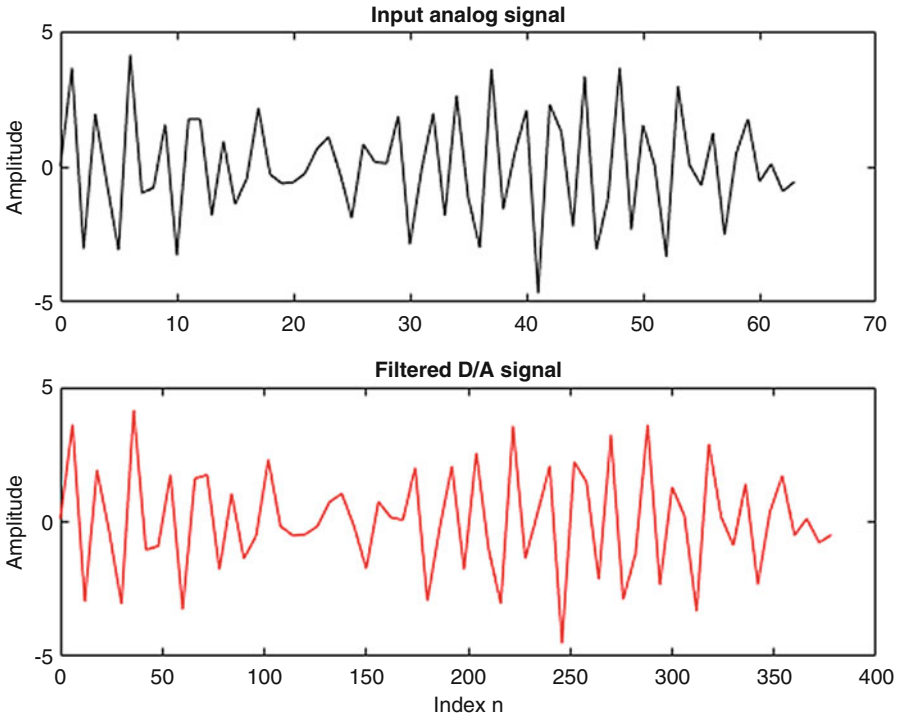

Fig. 10.15 Input analog and DAC output analog signals: top, input analog signal; bottom, DAC output

that the ISI can be minimized. This process is known as *pulse shaping* and is accomplished at the transmitter. Having taken care of the share of the transmitter, the channel effect can be canceled or minimized by a proper choice of a filter at the receiver. This is known as channel *equalization*. We have learnt phase and group delay equalization in the chapter on IIR digital filters. However, when the channel characteristics change slowly, then the equalizing digital filter coefficients should also change. This results in *adaptive equalizers* at the receiver. In any case, digital filtering is involved.

10.5.1 Pulse Shaping

According to Nyquist, if the overall system $H(f)$, from transmitter to receiver in (10.35), amounts to an ideal filter with a cutoff frequency equal to half the transmission symbol rate R_s, then there will be no ISI at the receiver. The ideal filter has the characteristics

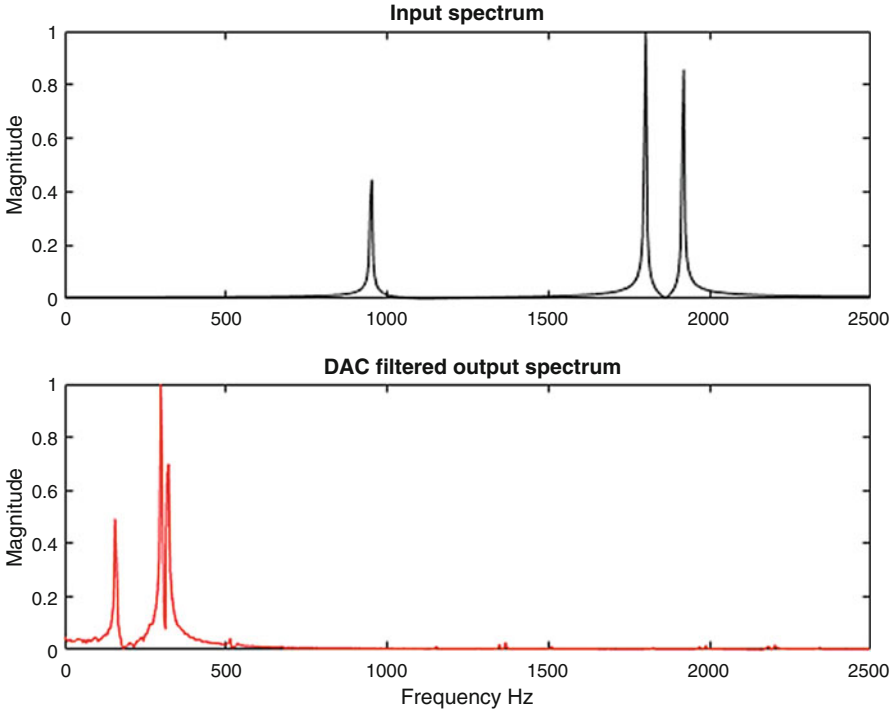

Fig. 10.16 Spectra of the signals in Fig. 10.15: top, spectrum of the input analog signal; bottom, spectrum of the DAC output analog signal

$$H(f) = \begin{cases} 1, |f| \leq W_0 \\ 0, otherwise \end{cases} \tag{10.36}$$

where

$$W_0 = \frac{1}{2T_s} = \frac{R_s}{2} \tag{10.37}$$

The corresponding impulse response of the ideal filter is given by

$$h(t) = \frac{\sin(2\pi W_0 t)}{\pi t} = 2W_0 sinc(2W_0 t), \ -\infty < t < \infty \tag{10.38}$$

The problem with the above ideal filter is that its impulse response is not zero for $t < 0$ and is, therefore, non-causal, meaning that it is not physically realizable as is. It is also susceptible to small timing errors. In effect, what we mean by the above statements is that the Nyquist bandwidth of the filter in (10.37) is not realizable. So, what are we going to do? Fortunately, one can allow some excess bandwidth,

thereby making the filter realizable. A small excess bandwidth is tolerable indeed. This is accomplished by using a filter with a frequency response described by

$$H_{RC}(f) = \begin{cases} 1, |f| \leq 2W_0 - W \\ cos^2\left(\frac{\pi}{4}\frac{|f| + W - 2W_0}{W - W_0}\right), 2W_0 - W < |f| \leq W \\ 0, |f| > W \end{cases} \tag{10.39}$$

Since the frequency response of the filter in (10.39) follows the square of a cosine function, it is called the *raised cosine* (RC) filter. In the raised cosine filter, the quantity $W - W_0$ is the *excess bandwidth* and the *roll-off* factor is defined as

$$r = \frac{W - W_0}{W_0} \tag{10.40}$$

Corresponding to (10.39), the impulse response of the RC filter can be shown to be

$$h_{RC}(t) = 2W_0 sinc(2W_0 t)\left[\frac{\cos(2\pi(W - W_0)t)}{1 - (4(W - W_0)t)^2}\right], -\infty < t < \infty \tag{10.41}$$

Have we solved anything in using RC filter? Even though the impulse response of the RC filter is non-causal, it decays very rapidly, and so it can be truncated without incurring any penalty. Thus, the RC filter becomes causal and realizable. What we are implying is that if the overall frequency response of the communications system from transmitter to receiver corresponds to the raised cosine filter response, i.e.,

$$H(f) = H_{RC}(f), \tag{10.42}$$

then there will be zero ISI because it satisfies Nyquist condition. One way to design the transmitter and receiver filters is to use the following:

$$|H_t(f)| = |H_r(f)| = \sqrt{H_{RC}(f)} \tag{10.43}$$

If (10.43) is satisfied, the overall frequency response of the communications system will not only satisfy Nyquist condition for zero ISI but will also be physically realizable.

Figure 10.17 shows the impulse response of an analog RC filter over the time interval $-0.02 \leq t \leq 0.02$ sec for three roll-off factors of 0, 0.5, and 1. As can be seen from the plots, the impulse response decays very rapidly for the roll-off factor 1. The frequency response of the RC filter corresponding to the three roll-off factors is shown in Fig. 10.18. The excess bandwidth is the largest for the case where the roll-off factor is 1. By discretizing the impulse response of the analog RC filter, we can obtain the impulse response of the corresponding digital filter. Figure 10.19 shows the discrete version of the impulse response of the RC filter, and its frequency response is shown in Fig. 10.20 for the same three values of the roll-off factor. For the discrete-time version of the RC filter, the sampling frequency is assumed to be 4.5 times the Nyquist bandwidth. We further show in Fig. 10.21 the process of

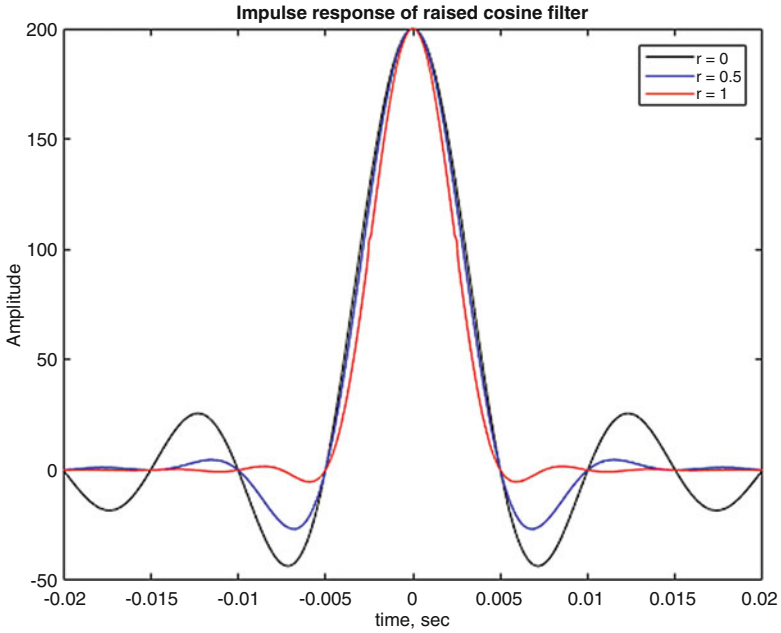

Fig. 10.17 Impulse response of the raised cosine filter in (10.41) with a Nyquist frequency of 100 Hz for three values of the roll-off factor

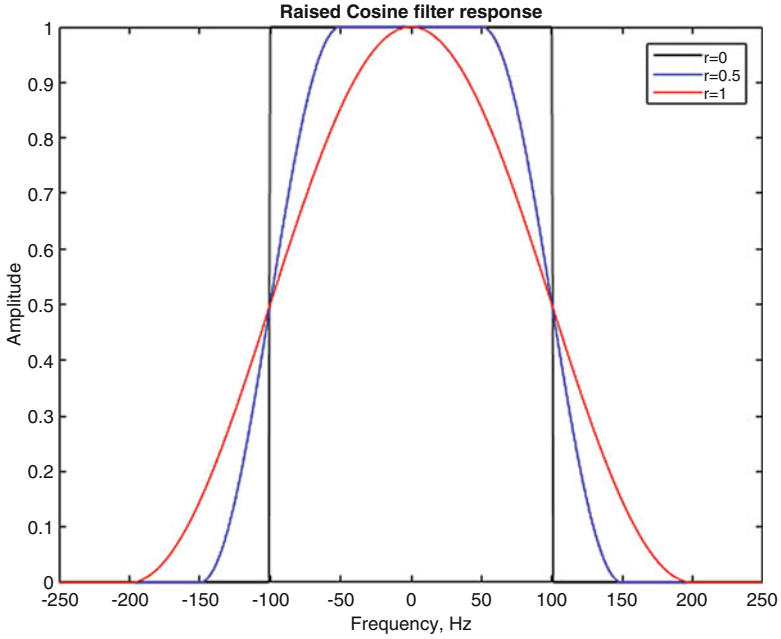

Fig. 10.18 Frequency response of the RC filter whose impulse response is shown in Fig. 10.17

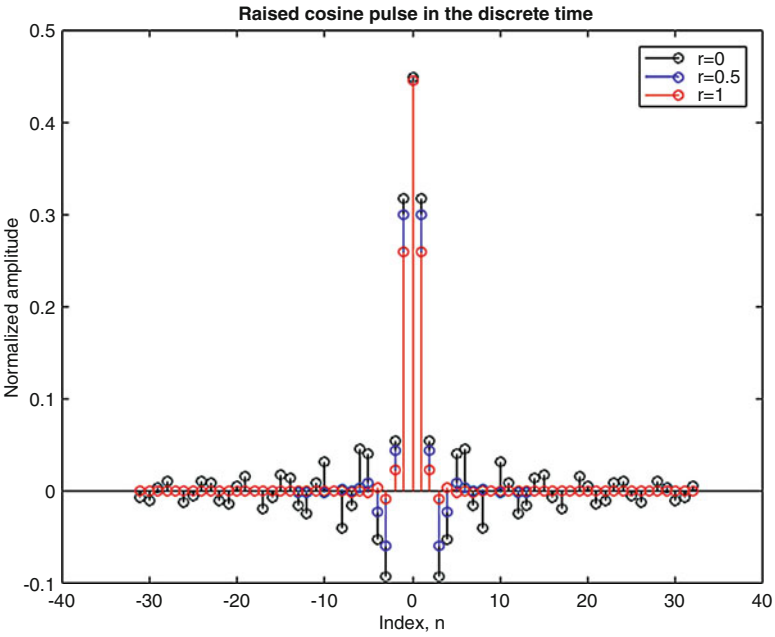

Fig. 10.19 Impulse response of the digital RC filter

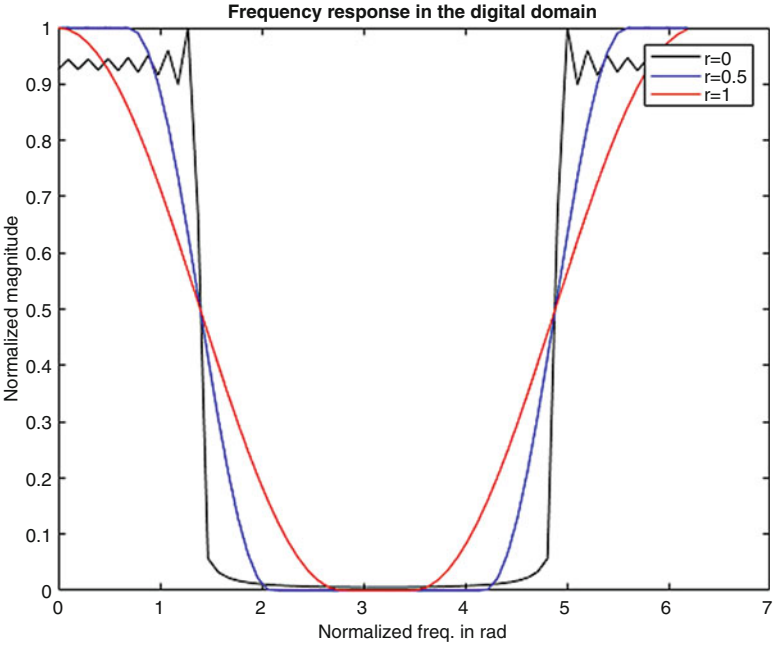

Fig. 10.20 Frequency response of the digital RC filter

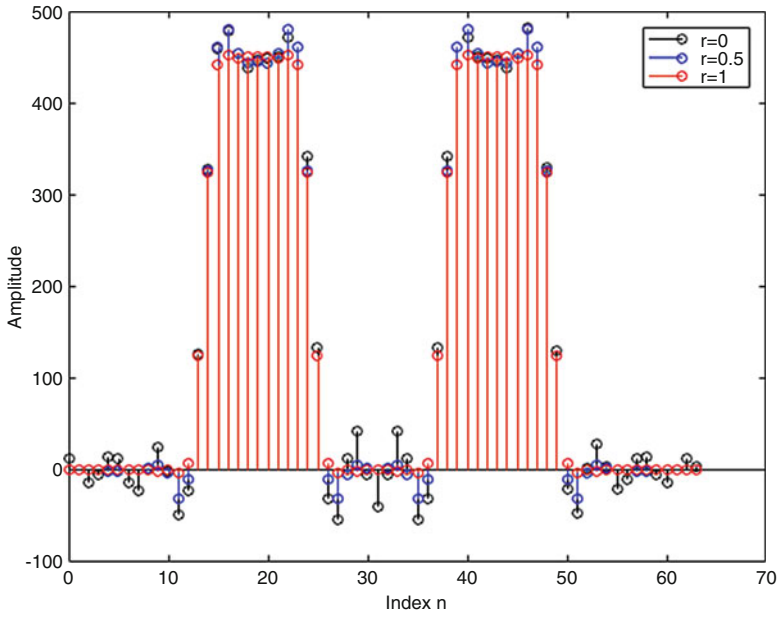

Fig. 10.21 Processing a sequence of pulses through the RC filter

filtering a sequence of rectangular pulses with the RC filter. We see no interference in the main lobe because of the RC shape of the lowpass filter. Even though there is some overlapping of the pulses in adjacent symbol intervals, there is no significant ISI due to the facts that the RC filter response decays very rapidly in the time domain and due to sinc(x) nature of the impulse response. The MATLAB M-file to obtain the impulse and frequency responses of the RC filter in both the continuous-time and discrete-time domains is named *Raised_cosine.m*.

Simulink Example for Pulse Shaping In this example we simulate the process of shaping a pulse sequence by a raised cosine filter using MATLAB's Simulink. Figure 10.22 shows the block diagram consisting of a pulse generator, whose output is filtered by an RC filter and two scopes to display the respective signals. The parameters of the pulse generator and RC filter are listed by the side of the respective blocks, as shown in the figure. The simulation time is chosen to be 1 sec. After starting the simulation, the respective outputs in the time domain are displayed on the scopes and are shown in Figs. 10.23 and 10.24. The Simulink program is named *RC_filter.slx*.

10.5.2 Equalization

Equalization is the process of correcting the ISI induced by the channel. There are linear and nonlinear equalization techniques available in the literature. We will only

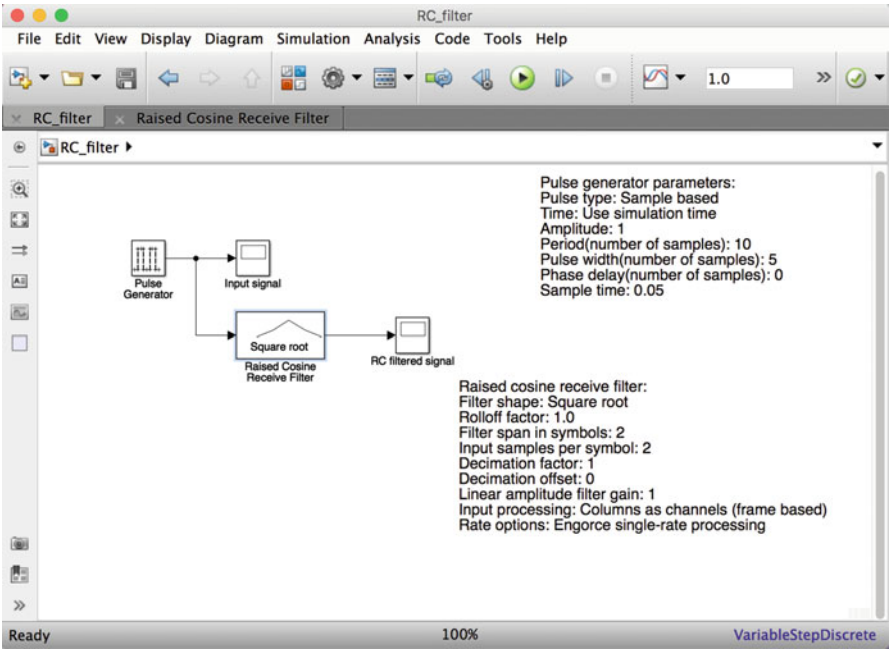

Fig. 10.22 Block diagram to simulate RC filtering of a pulse sequence

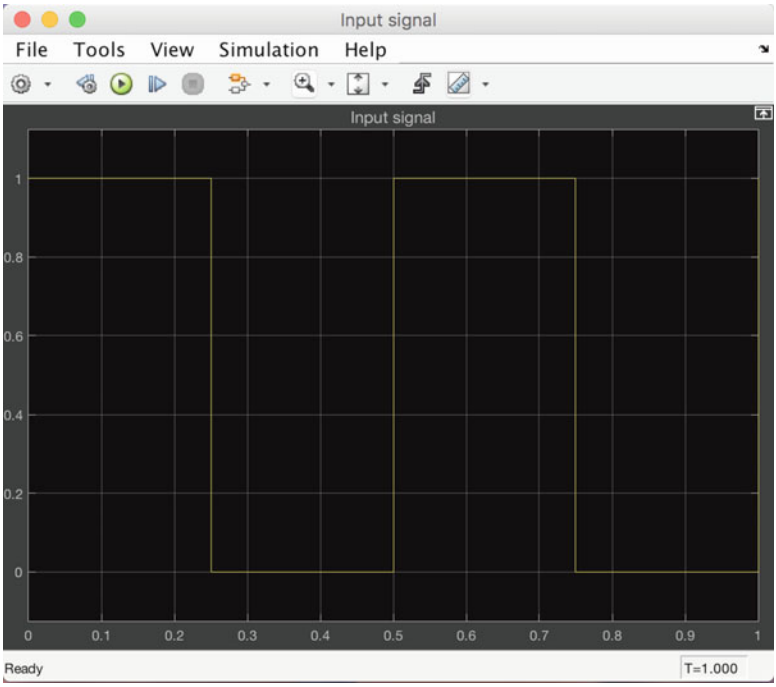

Fig. 10.23 Input pulse sequence to the RC filter

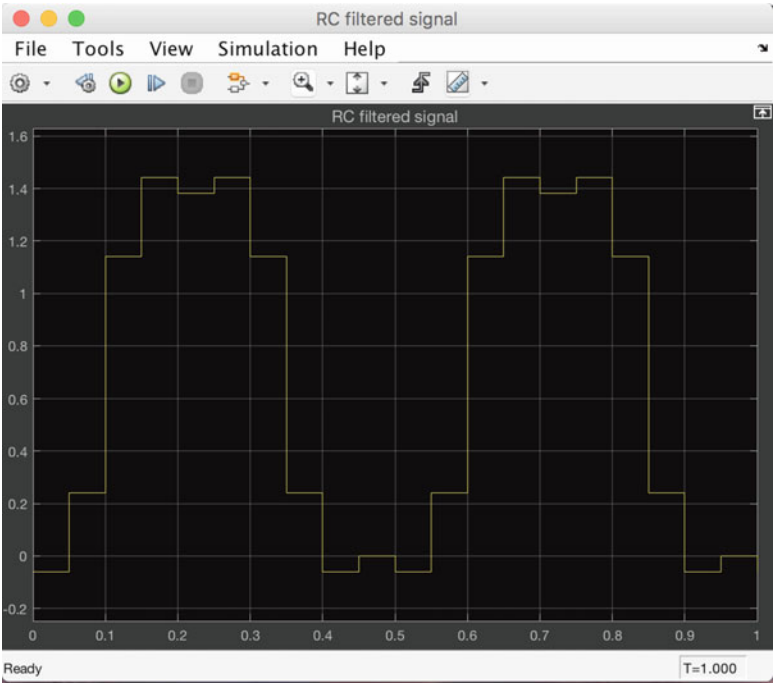

Fig. 10.24 RC-filtered sequence with no ISI

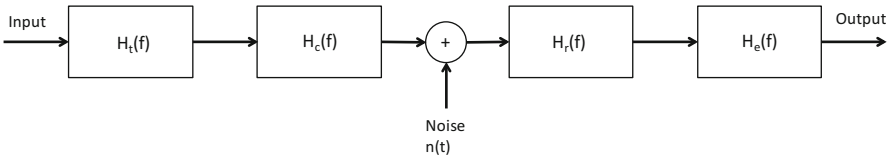

Fig. 10.25 Transmitter-receiver chain with an equalizing filter at the receiver to correct ISI due to channel impairments

deal with linear equalization procedures in this book. Mobile radio channel is nonstationary, meaning that the channel characteristics keep changing from time to time. This happens because the transmitted signal takes different paths as a result of reflection from the nearby tall buildings, hills, towers, etc. Also, since the receiver is not fixed, these reflected signals cause *fading*. This effect is known as *multipath fading*. Due to the relative motion between the transmitter and the receiver, there is the effect of Doppler spread in the received frequency. These impairments cause severe inter-symbol interference, which results in a high rate of bit errors at the receiver. As we saw earlier, if the overall system corresponds to a raised cosine filter function, then there will be no ISI. Therefore, the product of the transmitter and receiver filters equals the raised cosine filter response. In Fig. 10.25 is shown the

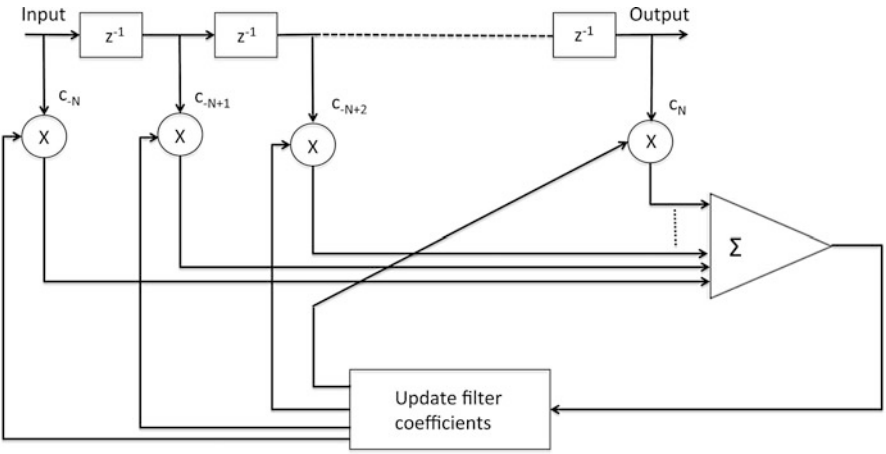

Fig. 10.26 A linear adaptive transversal filter as a channel equalizer

transmitter-channel-receiver tandem. At the receiver, an equalizing filter is employed to correct the channel impairments. The most common equalization filter is the *transversal* filter, which is an FIR filter. Due to the non-stationarity of the channel, the filter coefficients must be adapted to the changing statistics of the radio channel.

A block diagram of a linear adaptive transversal equalizer with 2 N + 1 taps is shown in Fig. 10.26. The delay in each delay element corresponds to a symbol duration. This type of equalizer is termed a *symbol-spaced equalizer*. The response of the transversal filter $y[k]$ can be expressed in terms of the input $x[n]$ as

$$y[k] = \sum_{n=-N}^{N} c_n x[k-n], \; -N \leq k \leq N \tag{10.44}$$

In compact matrix form, (10.44) can be written as

$$\boldsymbol{y} = \mathbf{X}\mathbf{c} \tag{10.45}$$

where

$$\boldsymbol{y} = [y_{-N} \cdots \cdots y_N]^T \tag{10.46a}$$

$$\boldsymbol{c} = [c_{-N} \cdots c_0 \cdots c_N]^T \text{ and} \tag{10.46b}$$

$$\boldsymbol{X} = \begin{bmatrix} x_{-N} \cdots \cdots \cdots 0 \\ x_{-N+1} \; x_{-N} \cdots \\ \vdots \\ \vdots \\ 0 \; 0 \cdots \cdots \cdots x_N \end{bmatrix} \tag{10.46c}$$

The criterion for selecting the filter coefficients $\{c_n\}$ is based on minimizing an objective function such as the mean square error (MSE) or absolute peak error. Since the channel statistics are time-variant, these coefficients are frequently changed using a suitable adaptation scheme. The number of taps in the transversal filter is usually chosen to be larger than the number of symbols involved in the ISI.

Simulink Example for Equalization In order to get a better picture of equalization to cancel ISI in digital communications, let us look at an example using MATLAB's Simulink. In this example, we consider an 8-ary QAM (quadrature amplitude modulation) scheme. In QAM, the amplitude of a carrier is modulated using discrete values. More specifically, the transmitted waveform is described by

$$x_{QAM}(t) = Ap(t)\cos(2\pi f_c t) + Bp(t)\sin(2\pi f_c t) \tag{10.47}$$

where $p(t)$ is a rectangular pulse of duration equal to the symbol duration, {A} and {B} are the sets of amplitudes with M values each, and f_c is the carrier frequency. These amplitudes have $M = 2^k$ discrete values corresponding to k-bit symbols. Let us choose M = 8. The simulation will be carried in the baseband, that is, no carrier modulation will be used. The channel introduces additive white Gaussian noise (AWGN) with a signal-to-noise ratio (SNR) of 20 dB. The channel is modeled as a four-tap FIR filter whose impulse response is described by

$$h_e[n] = 1 - 0.3z^{-1} + 0.1z^{-2} + 0.2z^{-3} \tag{10.48}$$

The adaptive transversal equalizer has 8 taps. The simulation uses least mean square (LMS) algorithm to adaptively estimate the filter taps. The signal sets are typically viewed as a constellation, where the two-dimensional vectors are described by

$$d_m = \left(\sqrt{E_s}A \quad \sqrt{E_s}B\right) \tag{10.49}$$

where E_s is the signal energy and $\{A\}$ and $\{B\}$ have M = 8 discrete values. In Fig. 10.27 is shown the block diagram of the 8-ary QAM system. The input to the LMS adapter is the signal from the AWGN block, and the desired signal is the output of the QAM modulator. The constellation diagrams of the signals before and after equalization are shown in Fig. 10.28 for an SNR of 20 dB. When the SNR is increased to 40 dB, the clusters appear more focused as seen from Fig. 10.29. The Simulink file is named *Adaptive_Equalizer.slx*. For more details on the parameters used in various blocks in Fig. 10.27, the reader may double-click each block to learn and modify the parameters.

10.5.3 Matched Filter

In digital communications, PCM binary digits are represented by pulses for transmission. In baseband communications, these pulses are transmitted as such, whereas

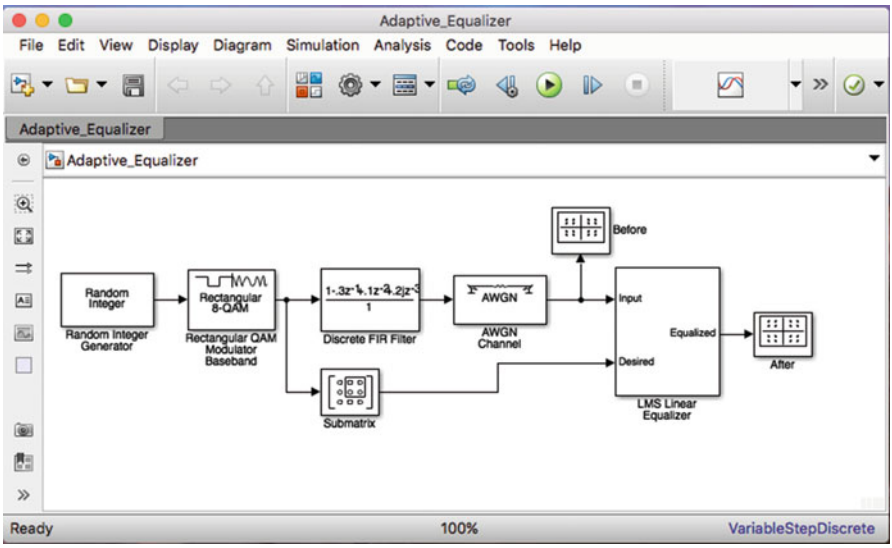

Fig. 10.27 Block diagram to simulate an 8-ary QAM system with equalization using LMS algorithm

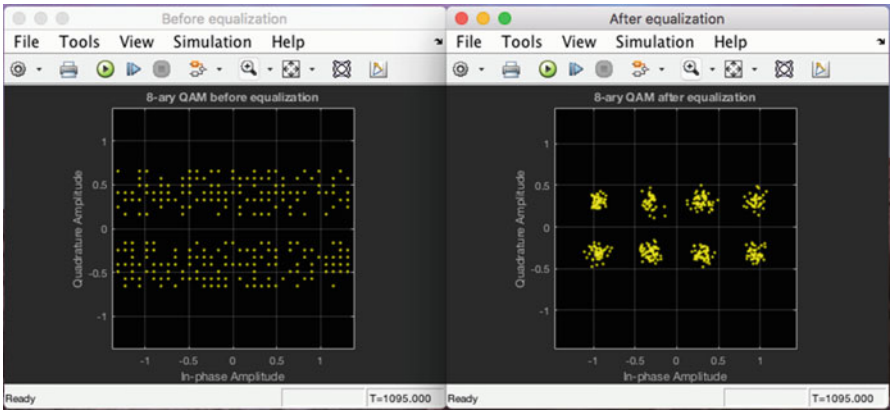

Fig. 10.28 Constellation diagram of the 8-ary QAM system in Fig. 10.27 showing the signal sets before and after equalization for an SNR of 20 dB: left, before equalization; right, after equalization

in carrier communications, these pulses modulate a carrier using different modulation schemes. We will consider baseband transmission here. The binary pulses may take one of non-return to zero (NRZ), return to zero (RZ), and Manchester code. Each one of these pulse types will affect the communications in terms of the DC component, self-clocking, error detection, bandwidth compression, noise immunity, etc. Our task here is to find out what is matched filter, why is it used in digital communications, and can it be realized as a digital filter. As pointed out earlier, the

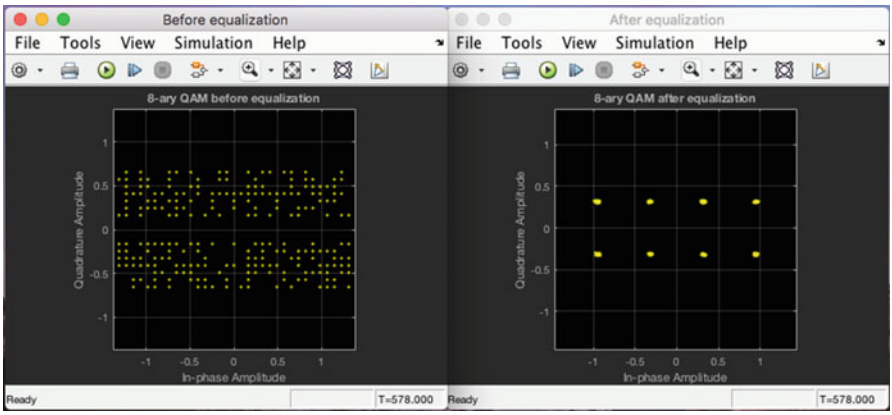

Fig. 10.29 Constellation diagram of the 8-ary QAM system in Fig. 10.27 showing the signal sets before and after equalization for an SNR of 40 dB: left, before equalization; right, after equalization

key factor in digital communications is the timing. The main task of the detector or receiver is to determine in each bit interval which binary digit – a binary "0" or a binary "1" – was transmitted. If there is no channel disturbance such as noise, then there is no problem in deciding which bit is transmitted in a given bit interval. However, the channel adds noise, namely, white Gaussian noise. Since the noise is added to the transmitted signal, this channel-induced noise is called the additive white Gaussian noise (AWGN). The received signal $r(t)$ is the sum of the transmitted signal and noise, as defined by

$$r(t) = s_i(t) + n(t), i = 1, 2; 0 \leq t \leq T \tag{10.50}$$

where T is the bit period and the transmitted signal takes the form

$$s_i(t) = \begin{cases} s_1(t), 0 \leq t \leq T, \textit{for a binary}\,'1' \\ s_2(t), 0 \leq t \leq T, \textit{for a binary}\,'0' \end{cases} \tag{10.51}$$

The processing consists of first filtering the received signal $r(t)$ followed by sampling the filtered signal $z(t)$ at the end of the bit period. Since the filter is LTI, the filtered signal is expressed as

$$z(t) = a_i(t) + n_0(t) \tag{10.52}$$

The sampled signal value $z(T)$ is then compared against a predetermined threshold value to decide which binary bit was transmitted in that bit interval. The sampled signal is described by

$$z(T) = a_i(T) + n_0(t) \tag{10.53}$$

If the threshold value is denoted by γ, then the decision amounts to

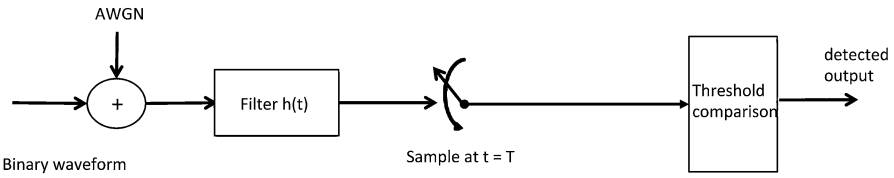

Fig. 10.30 Linear processing at the receiver to detect transmitted binary symbols

$$\widehat{s}(t) = \begin{cases} s_1(t) \text{ if } z(T) > \gamma \\ s_2(t) \text{ if } z(T) < \gamma \end{cases} \tag{10.54}$$

Figure 10.30 depicts the receiver operation. The linear time invariant (LTI) filter is implemented as a digital filter. The noise component in (10.53) is a zero-mean Gaussian random variable with a standard deviation σ_0. The probability density function (pdf) of the noise component $n_0(T)$ takes the form

$$p(n_0) = \frac{1}{\sigma_0\sqrt{2\pi}} exp\left(-\frac{n_0^2}{2\sigma_0^2}\right) \tag{10.55}$$

Since $z(T)$ is the sum of Gaussian noise and a signal component, it is also a Gaussian random variable with the same standard deviation as that of $n_0(T)$ but with a mean of $a_i(T)$. Thus, depending on which binary digit is transmitted, the pdf of $z(T)$ is given by

$$p(z|s_1) = \frac{1}{\sigma_0\sqrt{2\pi}} exp\left(-\frac{(z-a_1)^2}{2\sigma_0^2}\right) \tag{10.56a}$$

$$p(z|s_2) = \frac{1}{\sigma_0\sqrt{2\pi}} exp\left(-\frac{(z-a_2)^2}{2\sigma_0^2}\right) \tag{10.56b}$$

The two conditional pdfs in (10.56a) and (10.56b) are illustrated in Fig. 10.31 for the case $a_1 = -a_2 = 2$.

The objective of the detector is to detect which binary digit is transmitted in a given bit interval with the least amount of average bit error. As seen from Fig. 10.30, there are two variables to adjust so as to minimize the probability of a bit error. The first variable is the linear filter. By choosing the right filter, the probability of a bit error is minimized. This results in what is known as the *matched filter* (MF). The second variable is the threshold. Choosing the optimal threshold further minimizes the bit error probability. This results in *maximum likelihood receiver*.

Maximum Likelihood Receiver The decision threshold γ is chosen so as to minimize the probability of a bit error. This is achieved by maximizing the likelihood ratio

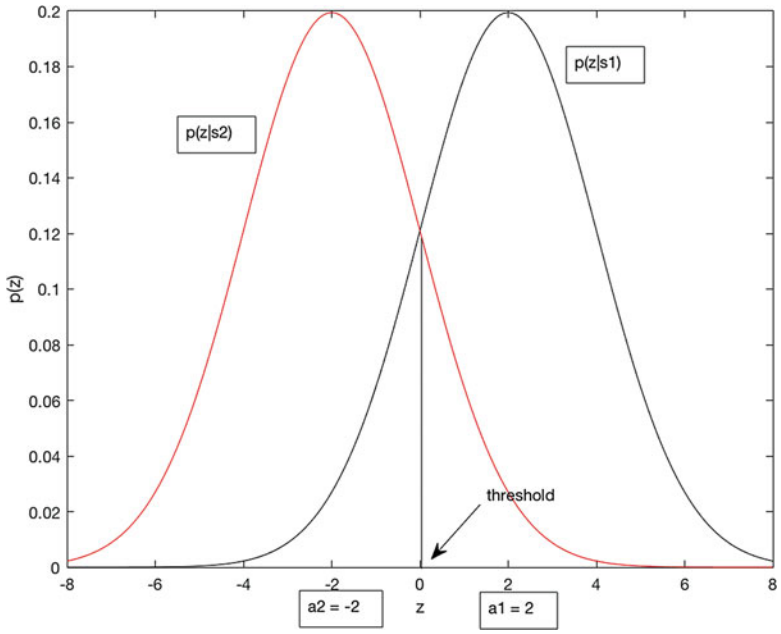

Fig. 10.31 Conditional probability density functions of the linearly processed and sampled signal

$$if\ \frac{p(z|s_1)}{p(z|s_2)} > \frac{P(s_2)}{P(s_1)}, choose\ s_1(t) \tag{10.57a}$$

$$if\ \frac{p(z|s_1)}{p(z|s_2)} < \frac{P(s_2)}{P(s_1)}, choose\ s_2(t) \tag{10.57b}$$

In the above equations, $P(s_1)$ and $P(s_2)$ are the a priori probabilities of the binary waveforms $s_1(t)$ and $s_2(t)$, respectively. Based on (10.57), the optimal threshold corresponds to the intersection of the two conditional pdfs, which is given by

$$\gamma_0 = \frac{a_1 + a_2}{2} \tag{10.58}$$

With the threshold value being determined, the probability of a bit error is determined as follows. The probability of making an error given s_1 was transmitted equals the area under the curve $p(z|s_1)$ from $-\infty$ to γ_0, which is

$$p(e|s_1) = \int_{-\infty}^{\gamma_0} p(z|s_1)dz \tag{10.59}$$

Similarly, the probability of making an error given s_2 was transmitted equals the area under the curve $p(z|s_2)$ from γ_0 to ∞, which is

$$p(e|s_2) = \int_{\gamma_0}^{\infty} p(z|s_2)dz \tag{10.60}$$

Since there are two symbols in the system, the average bit error is the weighted sum of the two conditional error probabilities in (10.59) and (10.60), which is expressed as

$$P_B = P(s_1)p(e|s_1) + P(s_2)p(e|s_2) \tag{10.61}$$

If the binary symbols are equally likely, that is, if $P(s_1) = P(s_2) = \frac{1}{2}$, then the probability of a bit error reduces to

$$P_B = \int_{\gamma_0 = \frac{a_1+a_2}{2}}^{\infty} p(z|s_2)dz = \frac{1}{\sigma_0\sqrt{2\pi}} \int_{\gamma_0}^{\infty} exp\left(-\frac{(z-a_2)^2}{2\sigma_0^2}\right)dz \tag{10.62}$$

By replacing $\frac{z-a_2}{\sigma_0}$ by x, the lower limit in (10.62) becomes $\frac{a_1-a_2}{2\sigma_0}$. Then the probability of a bit error amounts to

$$P_B = \frac{1}{\sqrt{2\pi}} \int_{\frac{a_1-a_2}{2\sigma_0}}^{\infty} exp\left(-\frac{x^2}{2}\right)dx = Q\left(\frac{a_1-a_2}{2\sigma_0}\right) \tag{10.63}$$

where $Q(x)$ is called the *complementary error function* and is defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} exp\left(-\frac{x^2}{2}\right)dx \tag{10.64}$$

In other words, the complementary error function equals the area under the normal curve with zero mean and unit variance from x to ∞. It does not have a closed form solution but is found in most standard textbooks on digital communications. It is also available in MATLAB as a built-in function *erfc*.

Matched Filter From (10.63), we observe that larger the threshold value, smaller the area under the normal curve or smaller the probability of a bit error. The matched filter maximizes the argument of the complementary error function. Let the input to the LTI filter in Fig. 10.30 be the sum of a known signal $s(t)$ and an AWGN $n(t)$. The output of the filter at the end of the bit interval T equals

$$z(T) = a_i(T) + n_0(T) \tag{10.65}$$

Therefore, the signal-to-noise ratio at $t = T$ is

$$\left(\frac{S}{N}\right)_T = \frac{a_i^2}{\sigma_0^2} \tag{10.66}$$

What we need is a filter that maximizes the SNR in (10.66). The signal component at the filter output can be related to the filter transfer function via Fourier transform by

$$a(t) = \int_{-\infty}^{\infty} H(f)S(f)e^{j2\pi ft}df \tag{10.67}$$

In (10.67), S(f) is the Fourier transform of the signal s(t). If we assume the two-sided power spectral density of the output Gaussian noise to be $\frac{N_0}{2}$ watts/Hz, then the noise power at the output of the filter is given by

$$\sigma_0^2 = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2 df \tag{10.68}$$

Using (10.66, 10.67, and 10.68), the SNR at the end of the bit interval becomes

$$\left(\frac{S}{N}\right)_T = \frac{\left|\int_{-\infty}^{\infty} H(f)S(f)e^{j2\pi fT}df\right|^2}{\frac{N_0}{2}\int_{-\infty}^{\infty}|H(f)|^2 df} \tag{10.69}$$

Using Schwartz's inequality, the numerator of (10.69) can be written as

$$\left|\int_{-\infty}^{\infty} H(f)S(f)e^{j2\pi fT}df\right|^2 \leq \int_{-\infty}^{\infty}|H(f)|^2 df\int_{-\infty}^{\infty}|S(f)|^2 df \tag{10.70}$$

Using (10.70) in (10.69), the expression for the SNR at t = T takes the form

$$\left(\frac{S}{N}\right)_T \leq \frac{2}{N_0}\int_{-\infty}^{\infty}|S(f)|^2 df \tag{10.72}$$

The maximum SNR is then equal to

$$max\left(\frac{S}{N}\right)_T = \frac{2}{N_0}\int_{-\infty}^{\infty}|S(f)|^2 df = \frac{2E}{N_0} \tag{10.73}$$

where the signal energy E is given by

$$E = \int_{-\infty}^{\infty}|S(f)|^2 df \tag{10.74}$$

The equality holds in Schwartz's inequality if the following condition is met:

$$H(f) = H_0(f) = KS^*(f)e^{-j2\pi fT} \tag{10.75}$$

The interpretation of (10.75) in the time domain from the time reversal property of the Fourier transform is that the impulse response of the matched filter is the time-reversed and right-shifted version of the signal. Mathematically speaking, it amounts to

$$h(t) = \begin{cases} Ks(T-t), 0 \leq t \leq T \\ 0, otherwise \end{cases} \tag{10.76}$$

From the above equation, we see that the filter impulse response is a replica of the transmitted pulse waveform but for the amplitude and time reversing and shifting. That is the reason the filter is known as the matched filter. Since the MF maximizes the SNR at the end of the bit interval, the maximum SNR is written as

$$\left(\frac{S}{N}\right)_T = \frac{(a_1 - a_2)^2}{\sigma_0^2} = \frac{E_d}{\frac{N_0}{2}} = \frac{2E_d}{N_0} \tag{10.77}$$

where energy difference in (10.77) stands for

$$E_d = \int_0^T (s_1 - s_2)^2 dt \tag{10.78}$$

Using (10.77) and (10.78), the probability of a bit error reduces to

$$\mathrm{P}_B = Q\left(\frac{a_1 - a_2}{2\sigma_0}\right) = Q\left(\sqrt{\frac{2E_d}{N_0}}\right) \tag{10.79}$$

Correlation Filter Another interpretation of the MF is as follows. We can express the response of the MF to the received signal at time t as

$$z(t) = \int_0^t r(\tau)s(T - t + \tau)d\tau \tag{10.80}$$

At the end of the bit interval t = T, the MF response becomes

$$z(T) = \int_0^T r(\tau)s(\tau)d\tau, \tag{10.81}$$

which is what is known as the correlation of r(t) with s(t). Therefore, the MF is also a correlation filter. To implement the MF as a correlation filter, we multiply the received signal by a replica of the transmitted signal and integrate the product over the bit interval. The output of the correlation filter at the end of the bit interval is the same as that of the MF.

MF Example Let us consider a simple example based on MATLAB to determine the impulse response and the filter response of a matched filter corresponding to a binary signal set. Let the two signals be defined by

$$s_1[n] = \begin{cases} 1, 0 \le n \le 9 \\ 0, otherwise \end{cases} \tag{10.82a}$$

$$s_2[n] = \begin{cases} -1, 0 \le n \le 9 \\ 0, otherwise \end{cases} \tag{10.82b}$$

Figure 10.32 shows the signal $s_1[n]$ in the top plot and the impulse response of the corresponding matched filter in the bottom plot. Note that the two sequences look identical because the MF impulse response is flipped and shifted to the right by the

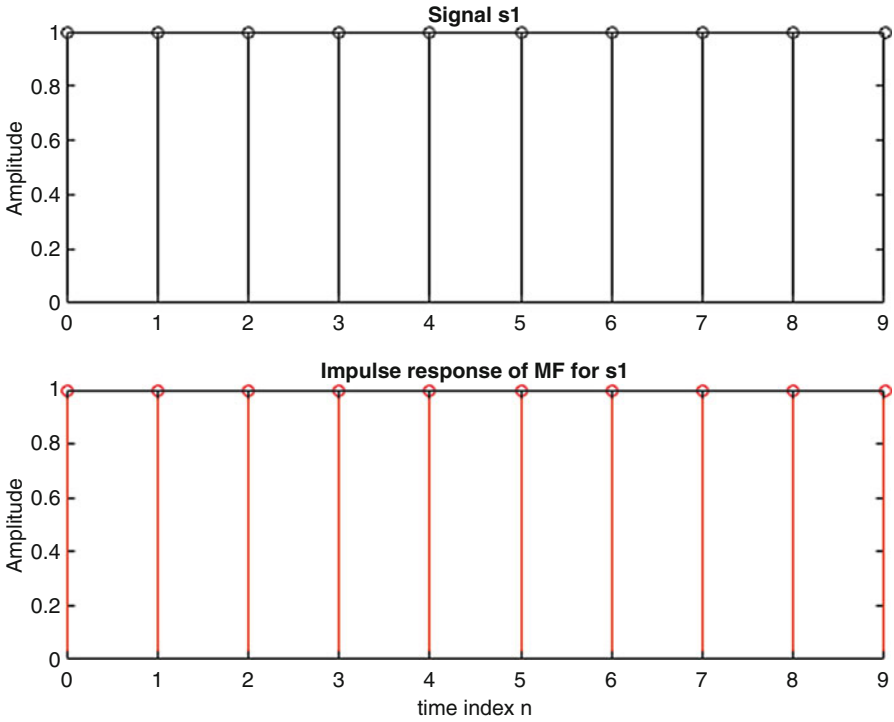

Fig. 10.32 Signal $s_1[n]$ and its MF impulse response: top, signal $s_1[n]$; bottom, corresponding MF impulse response

bit duration, which is of length ten samples. Similarly, the signal $s_2[n]$ and the corresponding MF impulse response are shown in Fig. 10.33. The responses of the two matched filters to input signal plus noise are shown in Fig. 10.34, where the top plot shows the MF response to the signal $s_1[n]$ plus noise and the bottom plot shows the MF response to the signal $s_2[n]$ plus noise. The noise is a Gaussian noise with a standard deviation of 0.25. As expected, the response is a maximum at the end of the bit interval. The responses of the corresponding correlation filters to the two signals are depicted in Fig. 10.35. The responses reach the maximum value at the end of the bit interval. Therefore, the correlation filters are equivalent to the matched filters. The M-file is named *Matched_filter.m*.

10.5.4 Phase-Locked Loop

Phase-locked loop, PLL for short, is a closed-loop control system. It acquires and tracks the phase of an incoming carrier signal and follows it so as to enable coherent demodulation. Analog modulation schemes may be amplitude modulation (AM) or

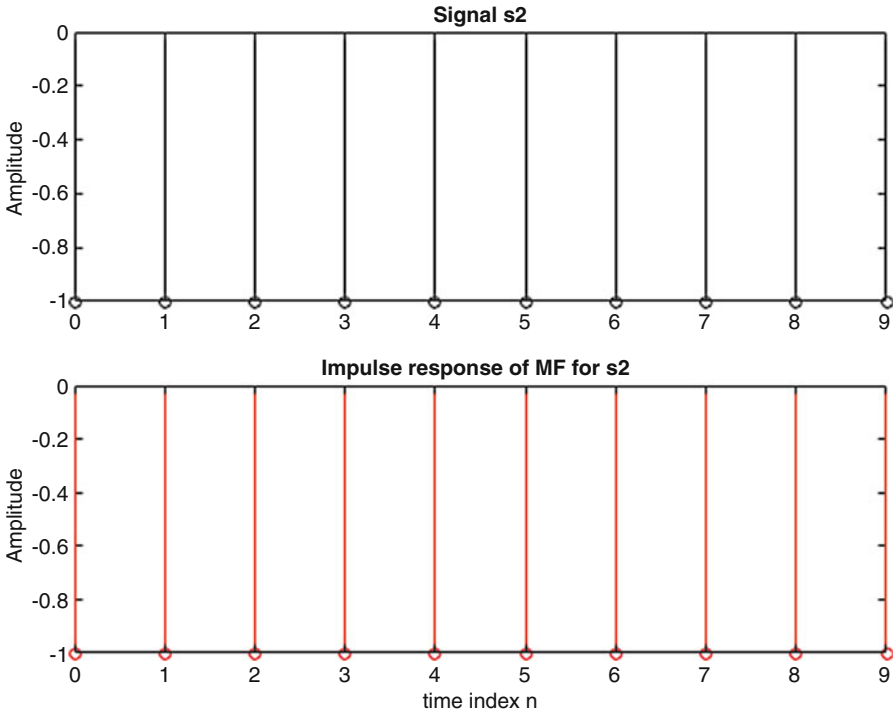

Fig. 10.33 Signal $s_2[n]$ and its MF impulse response: top, signal $s_2[n]$; bottom, corresponding MF impulse response

frequency modulation (FM). In either case, the message waveform can be recovered using coherent demodulation. Coherent demodulation requires a replica of the transmitted carrier to be available at the receiver. That is, the locally available carrier must have the same frequency and phase as that of the received carrier. The aim of the PLL is to enable coherent demodulation.

A PLL can be implemented either in analog form or digital form. We will first describe the analog PLL and then discuss the digital version later. A PLL consists of a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). This is shown in Fig. 10.36. The phase detector produces a signal, which is the difference in phase between the incoming and locally generated signals. The loop filter filters the output of the phase detector to pass the slowly varying phase component and reject the high-frequency component. The VCO generates a replica of the incoming carrier signal based on the output signal of the loop filter. The input to the VCO is a measure of the difference in phase between that of the incoming signal and VCO output. This input then drives the phase of the VCO output in the direction of the phase of the incoming carrier. Since a PLL is a closed-loop system, the phase error tends to zero a little after the start of the PLL. Once the phase error is zero, the PLL is said to be in lock. Then it tracks the phase of the incoming carrier.

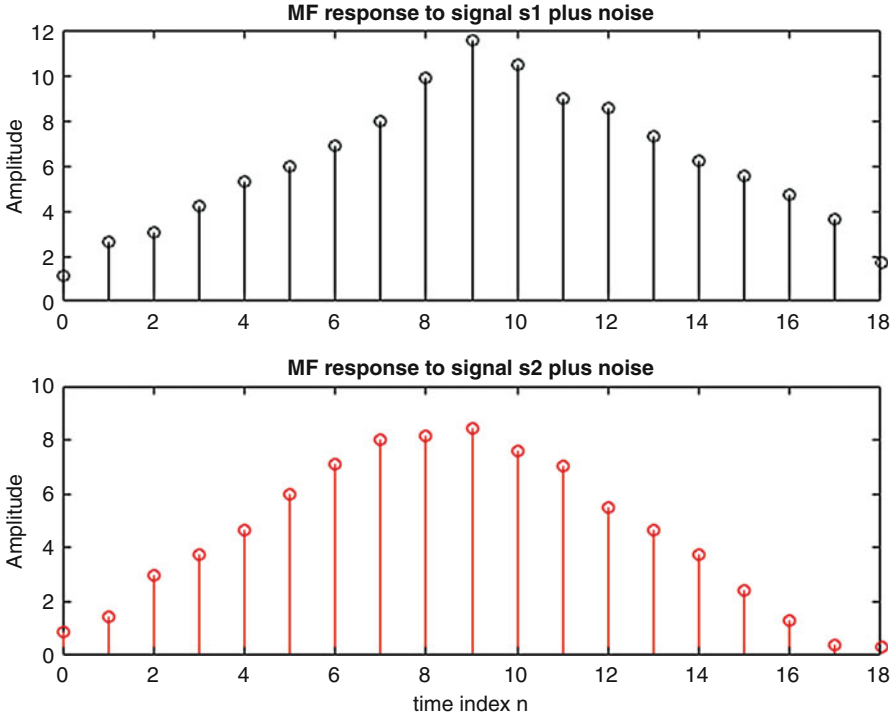

Fig. 10.34 Response of the MF to input signal plus noise: top, response of MF 1 to signal $s_1[n]$ plus Gaussian noise; bottom, response of MF 2 to signal $s_2[n]$ plus Gaussian noise. Noise standard deviation in both cases is 0.25

Analysis of a PLL As seen from Fig. 10.36, $r(t)$ is the received signal; $x(t)$ is the output of the VCO; the output of the phase detector is $e(t)$, which is the product of $r(t)$ and $x(t)$; and $v(t)$ is the output of the loop filter, which is the input to the VCO. Let the incoming carrier signal be described by

$$r(t) = \sin(2\pi f_c t + \theta(t)) \tag{10.83}$$

where the nominal frequency of the carrier is f_c and $\theta(t)$ is its phase, which is a slowly varying signal. Let the VCO output be defined as

$$x(t) = 2\cos(2\pi f_c t + \varphi(t)) \tag{10.84}$$

The phase detector accepts both $r(t)$ and $x(t)$ as inputs and outputs the product of the two input signals as described below.

$$e(t) = r(t)x(t) = 2\sin(\omega_c t + \theta(t))\cos(\omega_c t + \varphi(t)) \tag{10.85}$$

Using the trigonometric identity, the phase detector output can be written as

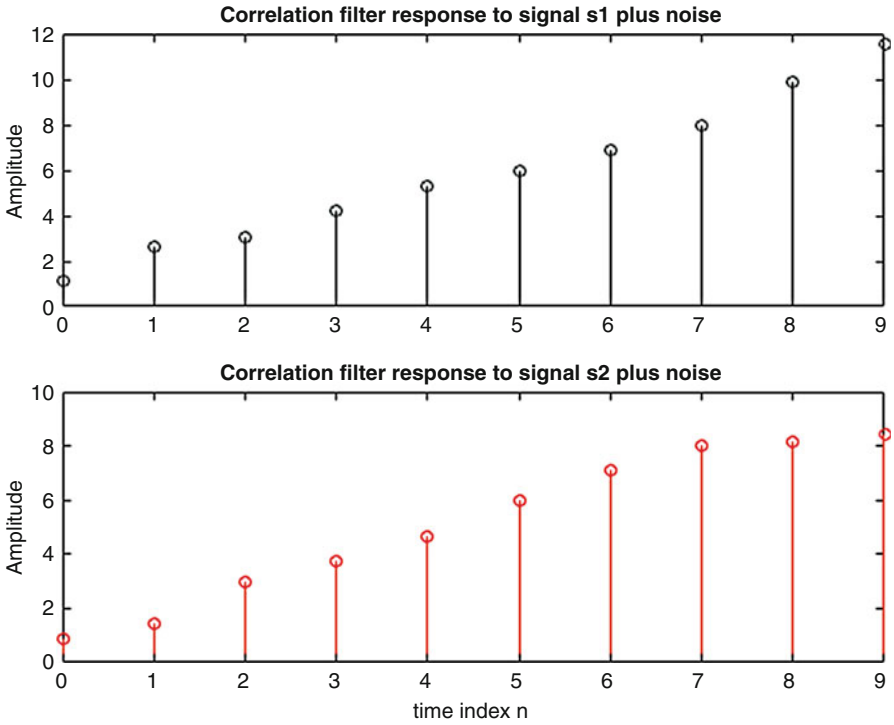

Fig. 10.35 Response of the correlation filter: top, response of correlation filter 1 to input signal $s_1[n]$ plus noise; bottom, response of correlation filter 2 to input signal $s_2[n]$ plus noise. The standard deviation of the noise is 0.25

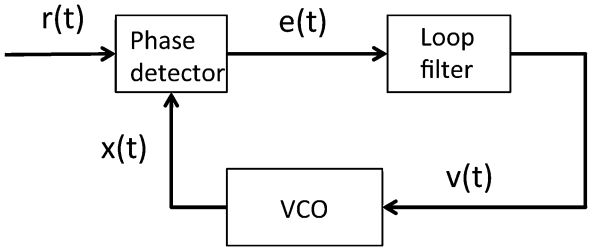

Fig. 10.36 Block diagram of a PLL

$$e(t) = \sin(\theta(t) - \varphi(t)) + \sin(2\omega_c t + \theta(t) + \varphi(t)) \tag{10.86}$$

Since the aim of the PLL is to track the phase of the incoming signal, the loop filter is designed to be a lowpass filter, which rejects the signal at twice the carrier frequency and passes the lowpass signal $\sin(\theta(t) - \varphi(t))$. The VCO generates a frequency deviation that is proportional to its input voltage $v(t)$. When $v(t) = 0$, the

VCO nominal frequency is ω_c. Denoting the instantaneous frequency deviation from ω_c by $\Delta\omega(t)$, we have

$$\Delta\omega(t) = \frac{d\phi(t)}{dt} = Kv(t) \tag{10.87}$$

where K is a VCO gain in rad/volt. Note that frequency is the derivative of the phase. If the impulse response of the loop filter is denoted by $g(t)$, then, since the loop filter is LTI, its response is the convolution of its input and impulse response. That is,

$$v(t) = e(t)*g(t) \tag{10.88}$$

Under locked condition, the phase error is very small and so

$$e(t) \simeq \theta(t) - \varphi(t) \tag{10.89}$$

Therefore, (10.87) becomes

$$\Delta\omega(t) = K(\theta(t) - \varphi(t))*g(t) \tag{10.90}$$

This gives rise to the linearized PLL, which is shown in Fig. 10.37. Using (10.87) in (10.90), we have

$$\frac{d\varphi(t)}{dt} + K\varphi(t)*g(t) = K\theta(t)*g(t) \tag{10.91}$$

By applying the Laplace transform on both sides of (10.91) and using the differentiation and convolution properties of the Laplace transform, (10.91) can be written as

$$s\Phi(s) + K\Phi(s)G(s) = K\Theta(s)G(s) \tag{10.92}$$

From (10.92) the closed-loop transfer function of the linearized PLL is expressed as

$$H(s) \equiv \frac{\Phi(s)}{\Theta(s)} = \frac{KG(s)}{s + KG(s)} \tag{10.93}$$

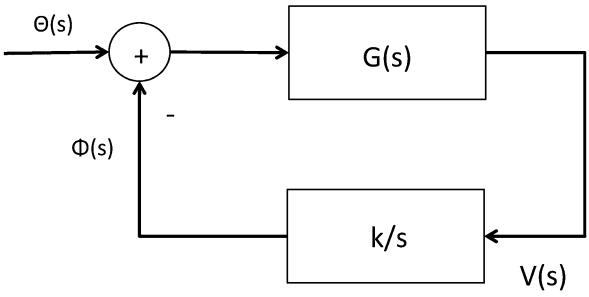

Fig. 10.37 Linearized PLL in the Laplace domain

In (10.92) and (10.93), $\Theta(s)$ and $\Phi(s)$ are the Laplace transforms of $\theta(t)$ and $\varphi(t)$, respectively. The linearized PLL in the Laplace domain is shown in Fig. 10.37. For $s = j\omega$, the frequency response of the linearized PLL takes the form

$$H(\omega) = \frac{KG(\omega)}{j\omega + KG(\omega)} \tag{10.94}$$

Steady-State Phase Error The Laplace transform of the phase error is given by

$$E(s) = \mathcal{L}\{e(t)\} = \Theta(s) - \Phi(s) \tag{10.95}$$

From (10.93),

$$\Phi(s) = \frac{KG(s)\Theta(s)}{s + KG(s)} \tag{10.96}$$

Using (10.96) in (10.95), the phase error in the Laplace domain is found to be

$$E(s) = \Theta(s)\left\{1 - \frac{KG(s)}{s + KG(s)}\right\} = \frac{s\Theta(s)}{s + KG(s)} \tag{10.97}$$

The steady-state phase error is the phase error as t tends to infinity. Using one of the properties of the Laplace transform, the steady-state phase error is obtained from

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{s^2\Theta(s)}{s + KG(s)} \tag{10.98}$$

Step Phase Response Let us assume that the PLL is in phase lock. When a unit step phase is then applied to the PLL at t = 0, the input phase in the Laplace domain is given by

$$\Theta(s) = \frac{1}{s} \tag{10.99}$$

If $G(0) \neq 0$, then the steady-state phase error becomes

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} \frac{1}{s} \frac{s^2}{s + KG(s)} = 0 \tag{10.100}$$

From (10.100), it is clear that the PLL tracks the input step phase.

PLL Response to a Frequency Step What happens if an abrupt step in the input frequency occurs? Will the PLL be able to track a frequency step? Let us investigate. Incidentally, a frequency step change could indicate a Doppler shift in the incoming signal frequency. This shift may be due to a relative motion between the transmitter and receiver. Since phase is the integral of the frequency, it changes linearly with respect to time when the frequency change is a step function. Using the integral in

time property of the Laplace transform, the input phase due to a frequency step change $\Delta\omega$ is given by

$$\Theta(s) = \frac{\Delta\omega}{s^2} \tag{10.101}$$

The steady-state phase error is found to be

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{s^2 \frac{\Delta\omega}{s^2}}{s + KG(s)} = \frac{\Delta\omega}{KG(0)} \tag{10.102}$$

The steady-state error depends on G(0). There are three possible types of loop filter, which are allpass, lowpass, and lead-lag filters and are defined below in that order.

$$G(s) = 1 \Rightarrow G(0) = 1 \tag{10.103a}$$

$$G(s) = \frac{\beta}{s+\beta} \Rightarrow G(0) = 1 \tag{10.103b}$$

$$G(s) = \left(\frac{\beta}{\alpha}\right)\frac{s+\alpha}{s+\beta} \Rightarrow G(0) = 1 \tag{10.103c}$$

In any case, G(0) = 1. Therefore, the steady-state phase error becomes

$$\lim_{t\to\infty} e(t) = \frac{\Delta\omega}{K} \tag{10.104}$$

and so the PLL tracks a step change in the input frequency. Even though the steady-state phase error is a constant and not zero, the PLL tracks a step change in input frequency with a constant phase error. Let us clarify the above discussion with a couple of examples.

Example 10.5a Step Phase: In this example, let us calculate the response of the PLL to a step phase input with the following specs. $G(s) = \frac{1}{s+1}$, and the VCO output is $X(s) = \frac{K}{s}V(s)$. Since we are going to implement this PLL in S/W, we will use a digital lowpass loop filter. The integrator, which models the VCO, in the discrete-time domain is simply a delayed accumulator. Let the VCO gain K = 0.1 and let the phase step of 0.95 rad be applied at the time index n = 510. The input phase, the VCO output phase, and the phase error are shown in Fig. 10.38 in top, middle, and bottom plots, respectively. As seen from the figure, the VCO phase undergoes a transient state and reaches a steady state after about ten sampling intervals. Similarly, the phase error reaches a steady state after about ten sampling intervals. The VCO input and output are shown in Fig. 10.39 in the top and bottom plots, respectively.

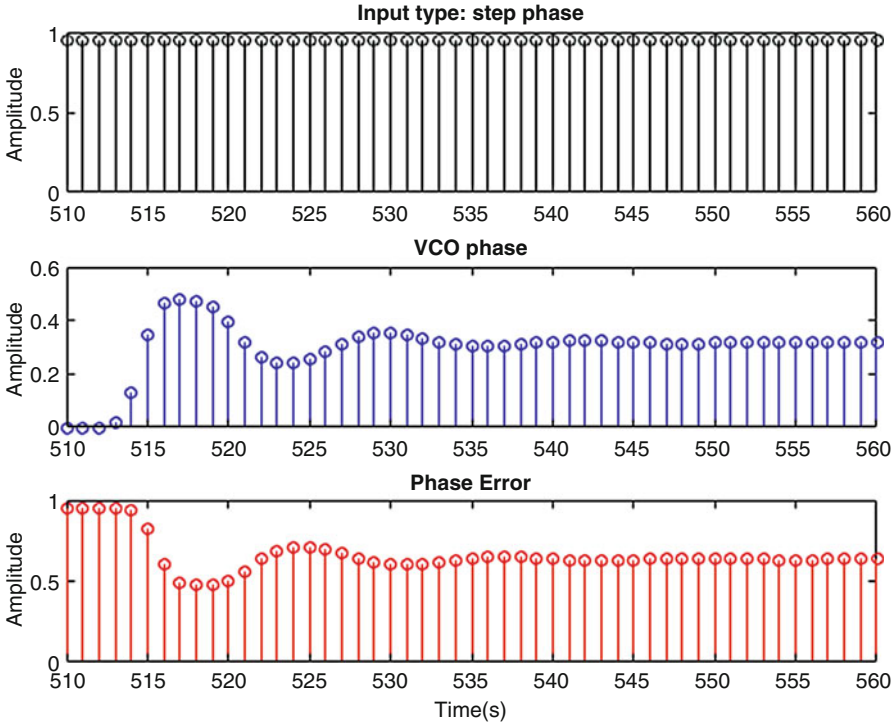

Fig. 10.38 Input phase and VCO phase in Example 10.5a: top, input step phase; middle, VCO output phase; bottom, phase error

Example 10.5b Pulse Phase: In this example we consider an input pulse phase to the PLL with the same loop filter and VCO as in the previous example. In Fig. 10.40 are shown the input phase, the VCO phase, and the phase error in the top, middle, and bottom plots, respectively. The VCO phase is a distorted pulse and so is the phase error. Similar to the previous case, the VCO input and output are shown in the top and bottom plots of Fig. 10.41.

Example 10.5c Ramp Phase: As a third example, let the input to the PLL be a ramp phase. The phase changes from 0 to 1 rad over 20 samples. Note that the instantaneous frequency is the time derivative of the phase. Since the phase varies linearly with time, the frequency will be a constant equal to the slope of the phase. Figure 10.42 shows the input phase, the VCO phase, and the phase error in the top, middle, and bottom plots, respectively. The VCO input and output are shown in the top and bottom plots in Fig. 10.43. The M-file used for the three cases of Example 10.5 is named *Example 10_5.m*.

Simulink Example to Simulate a Linearized Analog PLL To get a hands-on experience in working with PLL, let us look at simulating a linearized analog PLL using MATLAB's Simulink. In this example, the PLL functions in the baseband.

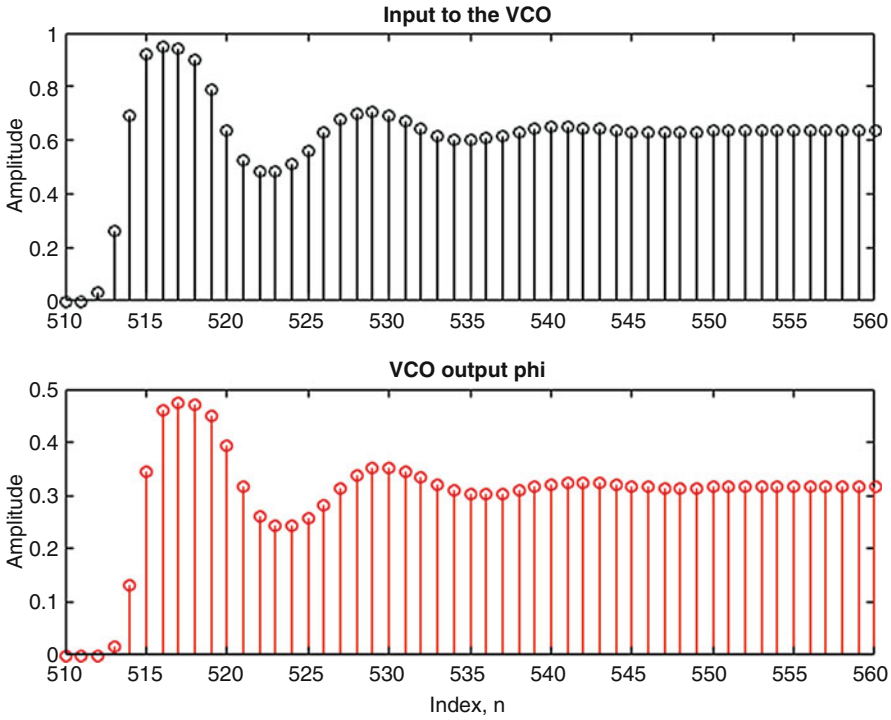

Fig. 10.39 Input and output phase of VCO in Example 10.5a: top, VCO input; bottom, VCO output phase

Under the Communications System Toolbox, we will find the subsystem named Synchronization, and under Synchronization, we have Components in which we will find the block named Linearized Baseband PLL. This block has one input and three outputs. The input is the signal whose phase is to be tracked. The three outputs are the phase detector (PD), the loop filter (Filt), and the voltage-controlled oscillator (VCO). The loop filter parameters to be entered in the PLL Block Parameters are the coefficients of the numerator and denominator polynomials of the lowpass filter transfer function. The coefficients correspond to the descending powers of the Laplace variable s. The filter chosen is a third-order Butterworth analog filter with a passband edge of 100 rad/s. The VCO parameter is its gain or input sensitivity in Hz/V. The input to the PLL block is a baseband sinusoidal source, which is found under Simulink – Source category. We have the option to use either sample based or time based. We will use sample based as the parameter under "sine type." We will also choose 100 samples/period under "samples per period" and an offset of 10 samples. The sample time is 0.01 s. The reader can view all the parameter options available as well as what are selected by double clicking each block in the simulation diagram. We can also add white Gaussian noise to the signal and the sum is fed to the PLL. Figure 10.44 shows the simulation block diagram used in this example.

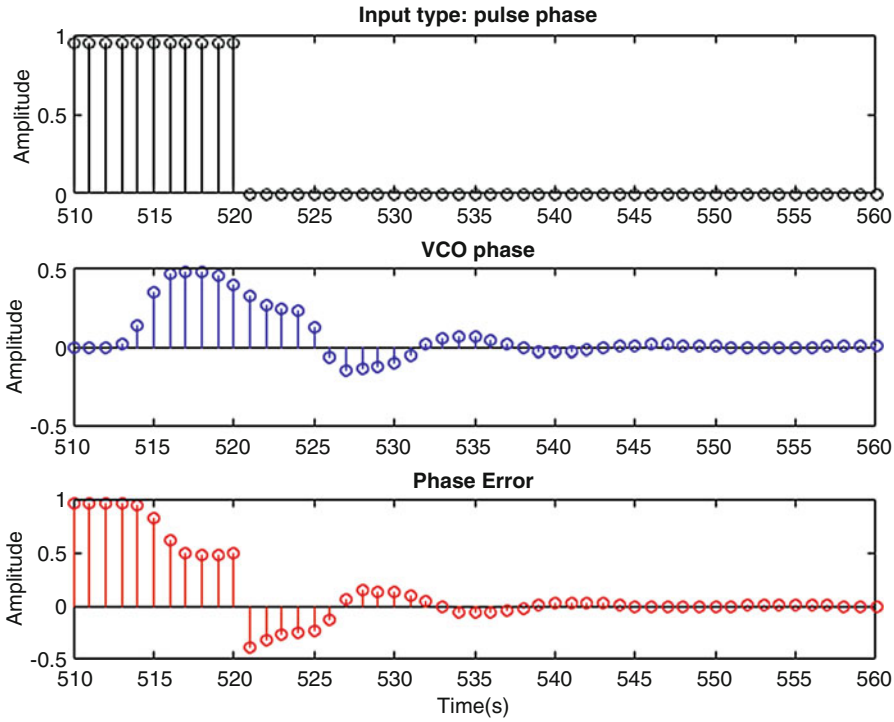

Fig. 10.40 Input phase and VCO phase in Example 10.5b: top, input pulse phase; middle, VCO output phase; bottom, phase error

After connecting all the blocks, we need to save the diagram in a file. The simulation time is chosen to be 10 s. To start the simulation, we have to click the green right arrow. If there are no errors, the simulation will start, and the results will be displayed on the respective scopes. First, let us simulate the PLL without noise. The input signal is shown in Fig. 10.45. As seen from the figure, there are 10 cycles over the 10 sec duration. The VCO output is shown in Fig. 10.46. As expected, it takes a few samples before the locking condition occurs. Next, we add white Gaussian noise with a power of 0.01 W and start the simulation. The VCO output is shown in Fig. 10.47 when noise is added. Due to the presence of noise, the VCO takes more time to track the incoming phase. The MATLAB file to simulate the linearized analog PLL is named *Linear_PLL.slx*.

Digital Phase Lock Loop A digital phase lock loop (DPLL) achieves the same purpose as the analog counterpart but with many advantages. A DPLL has a superior performance over an analog PLL. It is able to acquire and track much faster than an analog PLL. It is much more reliable and has lower size and cost. The VCO of an analog PLL is highly sensitive to temperature and power supply variations. Therefore, it needs not only an initial calibration but also frequent adjustments. This will

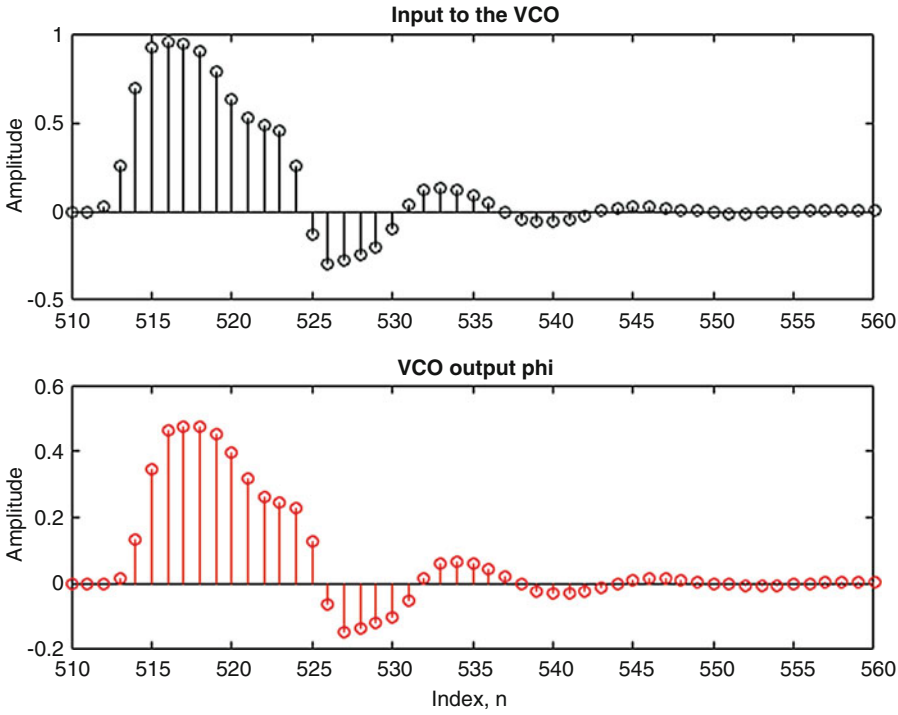

Fig. 10.41 Input and output phase of VCO in Example 10.5b: top, VCO input; bottom, VCO output phase

be a problem in consumer products such as a cellular phone. DPLL has no such problem. The phase detector in an analog PLL uses analog multipliers, which are sensitive to drift in DC voltage, whereas DPLL does not suffer from this problem. An analog PLL does not function well at low frequencies because the lowpass loop filter is analog. A larger time period is necessary for a better frequency resolution, which in turn reduces the locking speed. DPLL does not have this problem either. Moreover, since an analog PLL uses analog multipliers and analog filter, self-acquisition is slow and unreliable. DPLL has a faster locking speed. With so many advantages over an analog PLL, it is certainly desirable to use a DPLL instead. This also gives us the motivation to look into DPLL.

A simple block diagram of a DPLL is shown below in Fig. 10.48. There are different DPLLs available in the literature. One of them is called the Nyquist DPLL. In this type of PLL, the input analog sinusoidal signal is uniformly sampled at least at the Nyquist rate, and the analog samples are quantized to B-bits to form the input digital signal. It is then digitally multiplied by the output of the digital-controlled oscillator (DCO) to form the error sequence. This error sequence is then filtered by a lowpass digital filter. The output of the lowpass digital filter then controls the DCO frequency. Figure 10.49 shows the block diagram of a Nyquist DPLL.

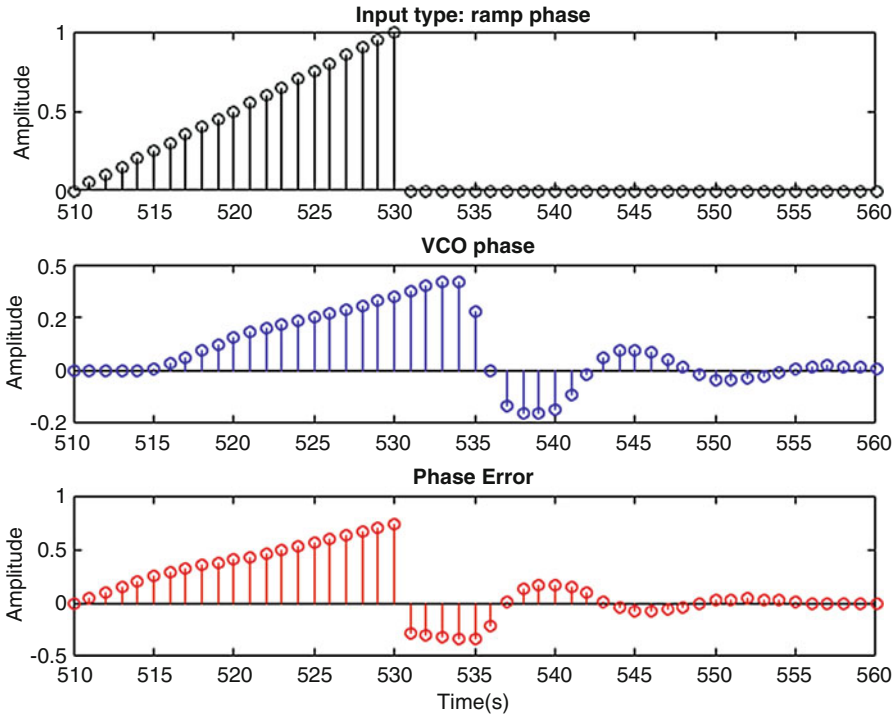

Fig. 10.42 Input phase and VCO phase in Example 10.5c: top, input ramp phase; middle, VCO output phase; bottom, phase error

Software-Based DCO Unlike the analog VCO, the DCO in the DPLL in Fig. 10.49 uses software or algorithm to generate the sinusoid. It uses the basic idea behind the analog VCO in the following manner. In the continuous-time or analog domain, the VCO output is described by

$$y(t) = B\cos\left(\omega_c t + K\int_0^t v(\tau)d\tau\right) \tag{10.105}$$

In the discrete-time domain, we can write the above VCO output as

$$y[n] = B\cos\left(\frac{2\pi f_c}{f_s}n + K\sum_{i=0}^{n-1} v[i]\right) \tag{10.106}$$

where f_s is the sampling frequency and the summation inside the argument of the cosine function corresponds to the integral of the VCO input. The sequence in (10.106) is then converted to a square wave, which is obtained by

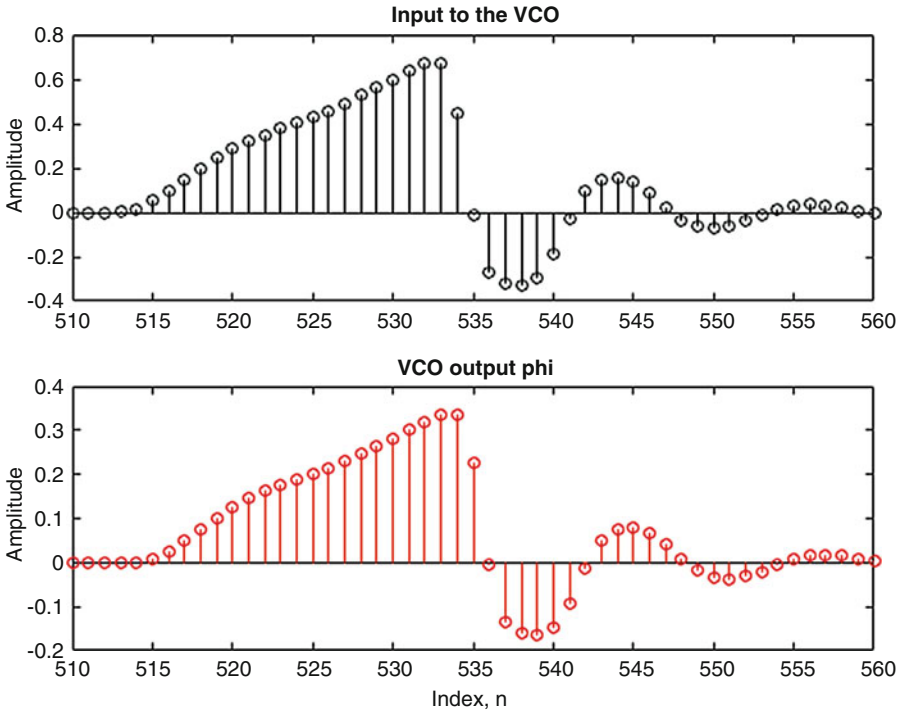

Fig. 10.43 Input and output phase of VCO in Example 10.5c: top, VCO input; bottom, VCO output phase

$$y[n] = sq\left(\frac{2\pi f_c}{f_s}n + K\sum_{i=0}^{n-1} v[i]\right) \tag{10.107}$$

where the function sq(x) is defined as

$$sq(x) = \begin{cases} 1, 0 \le x < \pi \\ -1, \pi \le x < 2\pi \end{cases} \tag{10.108}$$

and it is periodic as indicated below.

$$sq(x + 2m\pi) = sq(x), m \in Z \tag{10.109}$$

An Example to Illustrate the Function in Equation 10.108 Let us illustrate the conversion of the sequence in (10.107) into a square sequence using MATLAB. For the sake of illustration, we choose the sinusoidal frequency to be 100 Hz with a sampling frequency of 1000 Hz. Let the DCO gain be equal to 1 rad/volt. The program to run is named *SQ.m*. The discrete square sequence is shown in Fig. 10.50

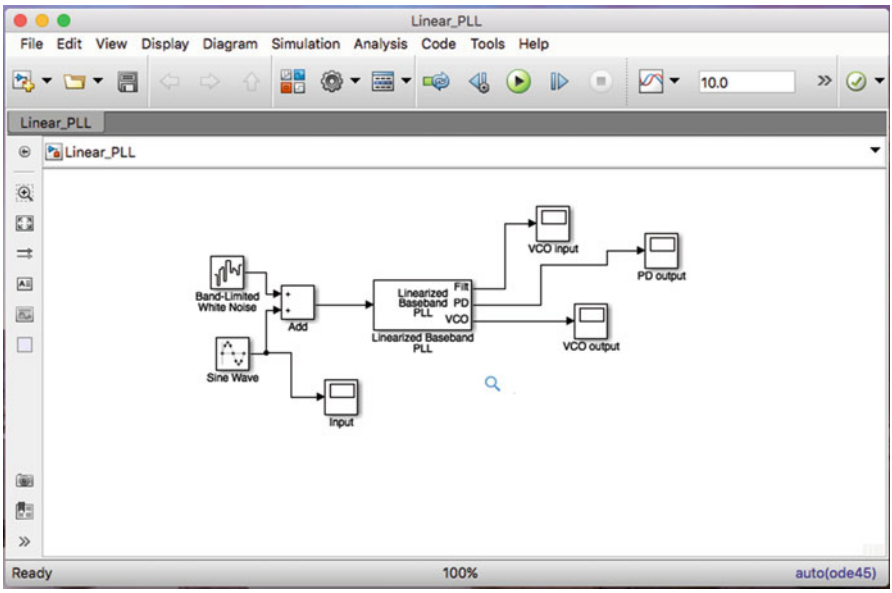

Fig. 10.44 Block diagram to simulate a linearized analog PLL using MATLAB's Simulink

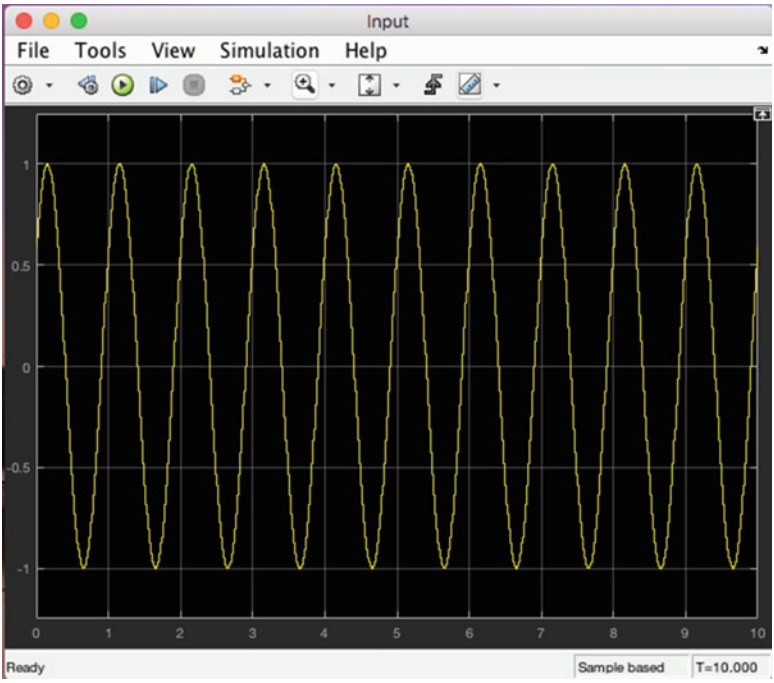

Fig. 10.45 Input analog sinusoid

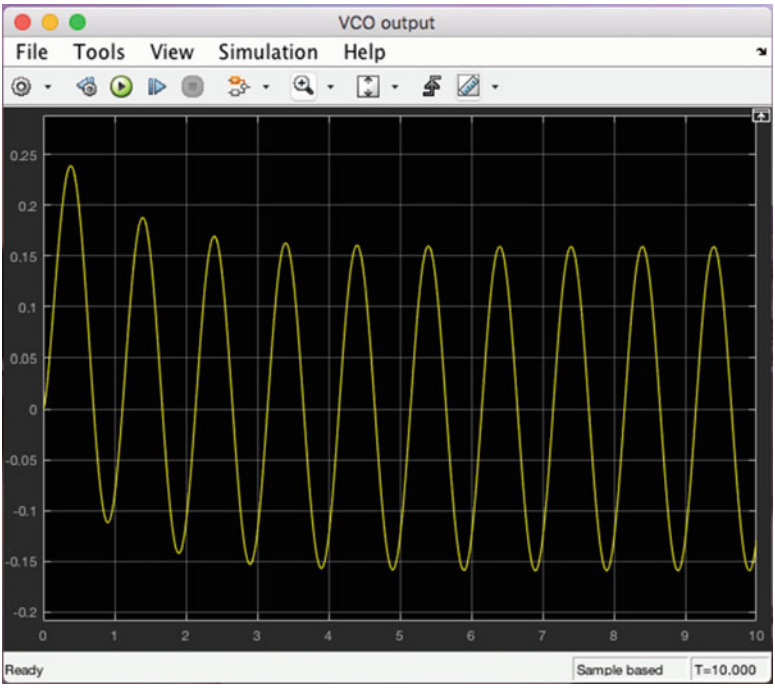

Fig. 10.46 VCO output with no input noise

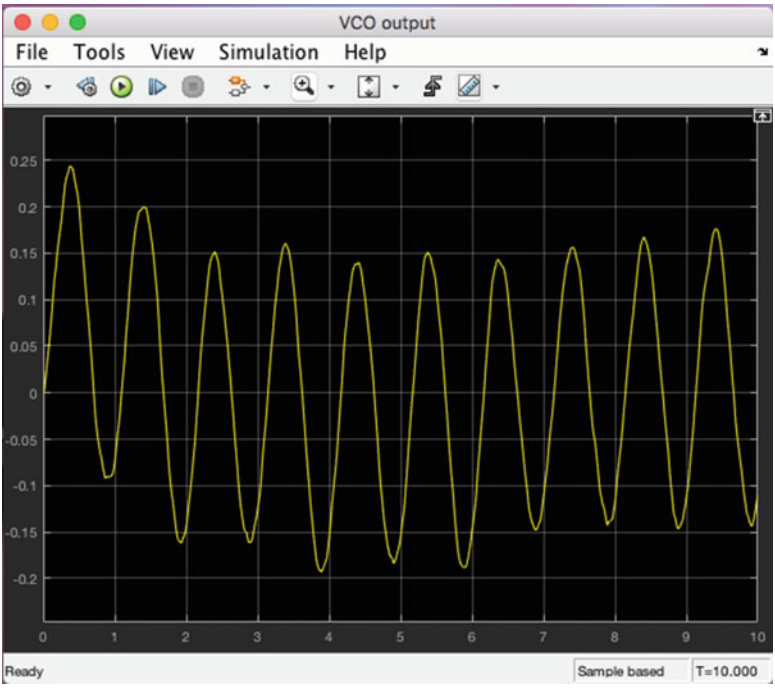

Fig. 10.47 VCO output with a white Gaussian noise of 0.01 W of power

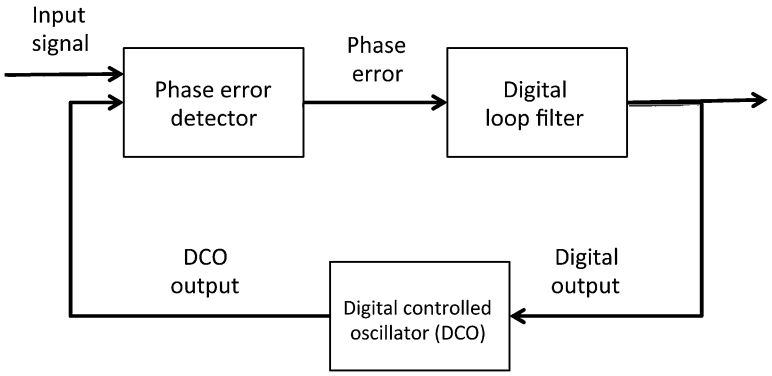

Fig. 10.48 Block diagram of a typical DPLL

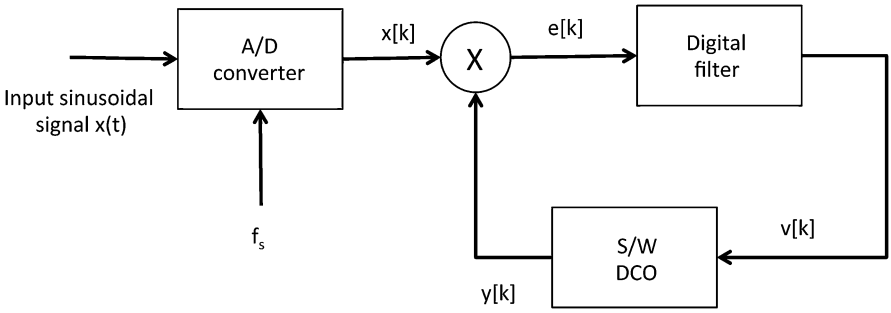

Fig. 10.49 Block diagram of a Nyquist DPLL

as a stem plot. The same sequence when the DCO gain is K = 2 is shown in Fig. 10.51 for comparison.

Simulink Example of a DPLL In this example, we will simulate a DPLL using MATLAB's Simulink. The block diagram for the simulation is shown in Fig. 10.52. It is similar to the one used in the simulation of an analog PLL. The main difference is in the PLL block. It is called charge-pump PLL. The phase detector outputs a square waveform as opposed to a continuous-time signal. The loop filter is a lowpass analog filter. The input signal is the same sinusoid used in the previous Simulink example. The loop filter is a third-order Butterworth lowpass filter with a passband edge of 10 rad/s. The VCO gain is 1.25 rad/volt. The block diagram for the simulation is shown in Fig. 10.52. The details of the parameters can be found by double clicking the respective block. The outputs of the phase detector, loop filter, and the VCO are shown in Figs. 10.53, 10.54, and 10.55, respectively. The VCO seems to track the incoming phase. The same three outputs are displayed in Figs. 10.56, 10.57, and 10.58 when the input signal is corrupted by an additive white Gaussian noise with a power of 0.01. Again, the VCO tracks the phase of the incoming signal.

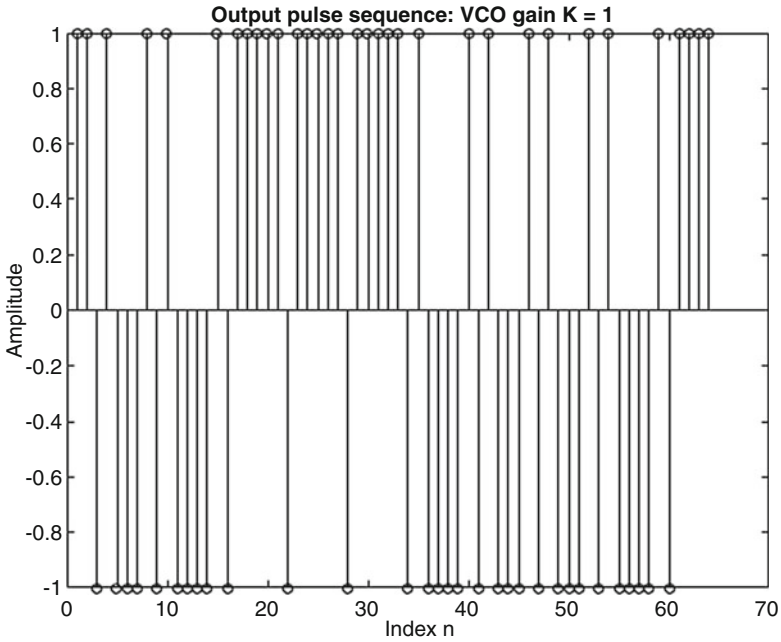

Fig. 10.50 The output sequence of the sq(x) function with the DCO gain equal to 1

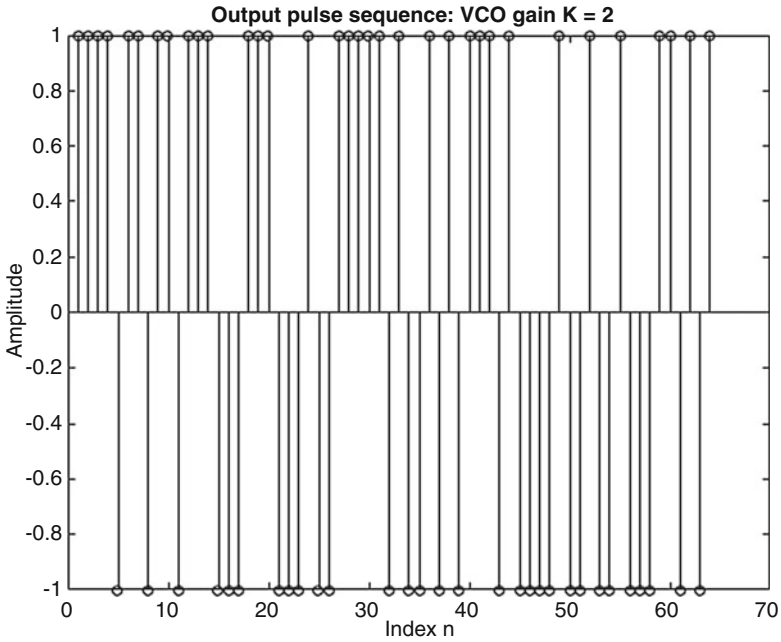

Fig. 10.51 The same output sequence of the sq(x) function with the DCO gain equal to 2

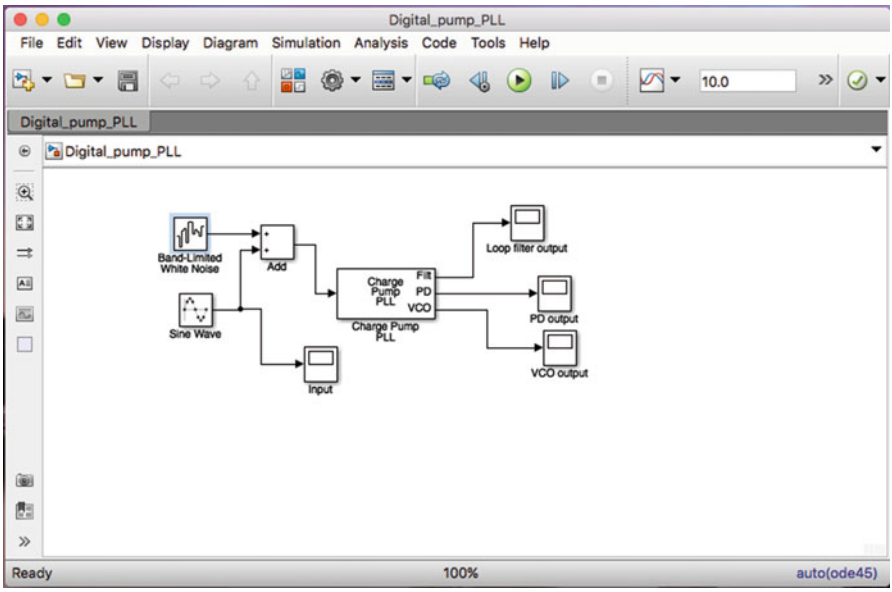

Fig. 10.52 Simulation block diagram of a charge-pump PLL

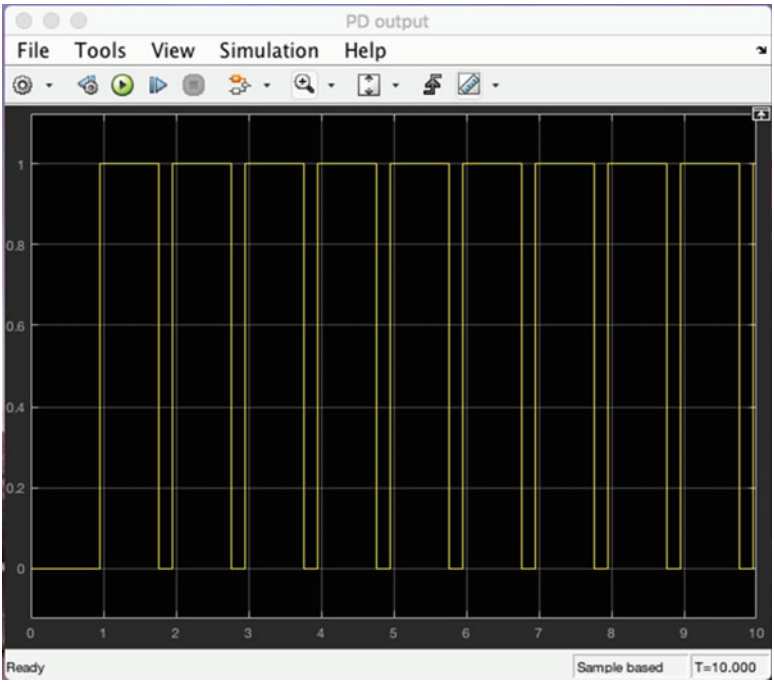

Fig. 10.53 Output of the phase detector when no noise is present

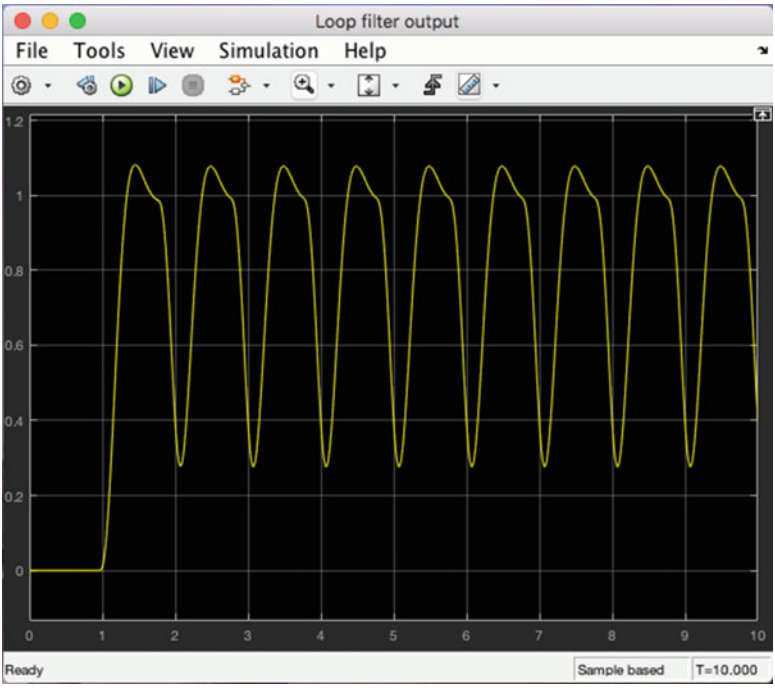

Fig. 10.54 Output of the loop filter when no noise is present

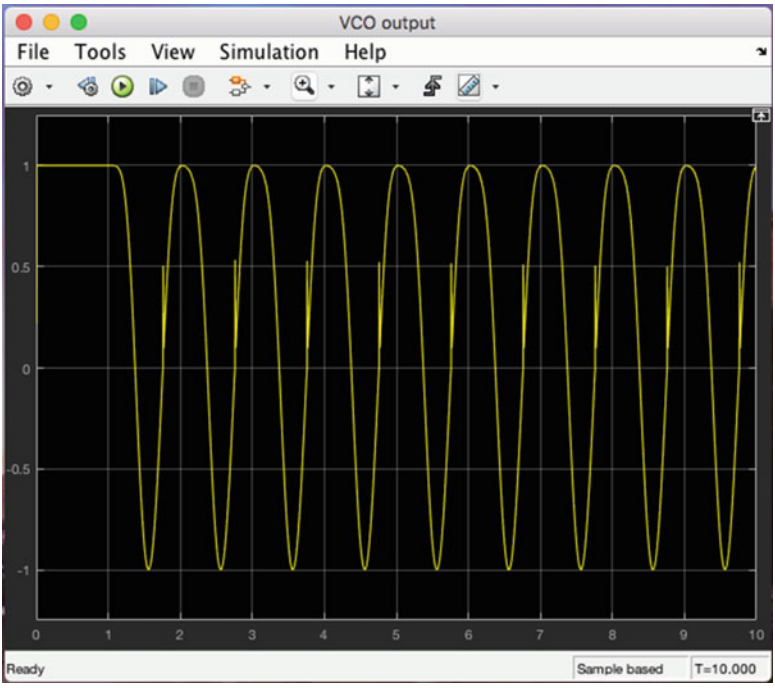

Fig. 10.55 Output of the VCO when no noise is present

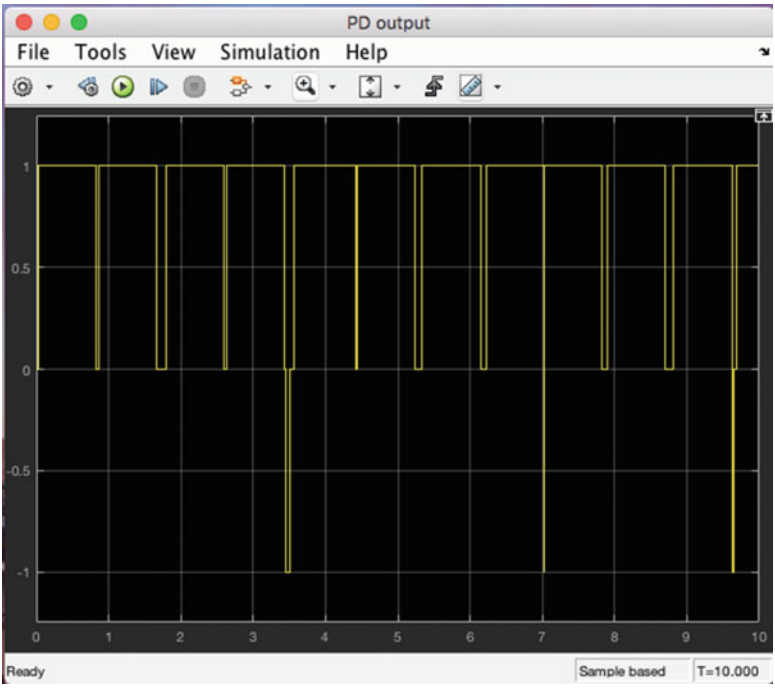

Fig. 10.56 Output of the phase detector when AWGN with power 0.01 is present

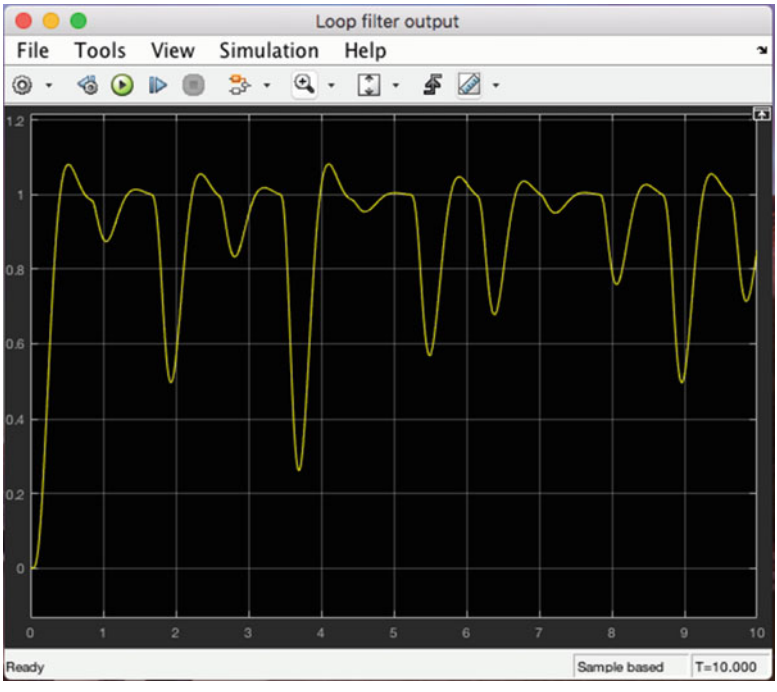

Fig. 10.57 Output of the loop filter when AWGN with power 0.01 is present

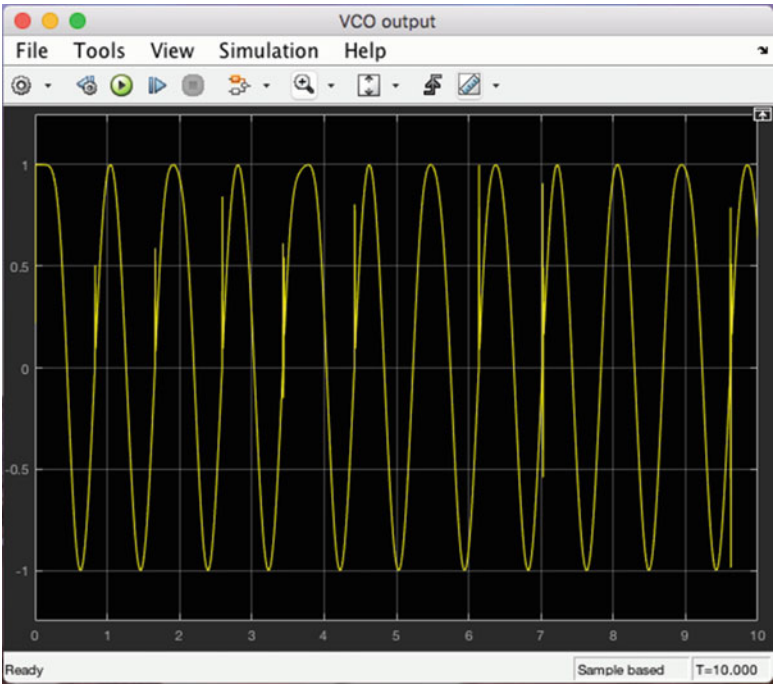

Fig. 10.58 Output of the VCO when AWGN with power 0.01 is present

10.5.5 OFDM

OFDM stands for orthogonal frequency-division multiplexing. It is also a multicarrier modulation scheme. In a conventional frequency-division multiplexing, data from different subscribers each modulate a subcarrier to form a non-overlapping spectrum. These subcarriers then modulate a final carrier for transmission. The problem with the conventional frequency-division multiplexing is the ISI at the receiver due to the dispersive fading channel effect and requires complex equalization. On the other hand, OFDM uses subcarriers that are orthogonal, which does not cause ISI. It also supports high data rates. Therefore, OFDM has gained popularity and is used in standards such as digital audio broadcasting and digital TV. It is also used in high data rate transmission in mobile wireless channels. Remember that our objective here is to show that DSP is used in OFDM.

OFDM Basics OFDM is a digital multicarrier modulation. The symbols from different users to be transmitted simultaneously are denoted by $\{X(k), 0 \leq k \leq N-1\}$. The OFDM signal in the time domain can be described by

$$x(t) = \sum_{k=0}^{N-1} X(k) e^{j2\pi f_k t}, 0 \leq t \leq T_s \tag{10.110}$$