

Chapter 2

Discrete and Digital Signals and Systems

2.1 Introduction

Before an analog signal can be processed with a digital computer chip, the signal need to be converted into digital form.

The *first* phase of the digitization process is to sample the analog signal at discrete time instants $t = nT_s$, where n is an integer (1, 2, 3,...) and where T_s is the sampling period (i.e., the time interval between successive samples). Note that this kind of sampling is uniform in the sense that T_s is constant, and the sampling frequency is also constant at $f_s = 1/T_s$. The result of sampling an analog signal in this fashion is a discrete-time signal, also known as a pulse amplitude modulation (PAM) signal. It should be noted that while uniform sampling is simple and intuitively appealing, it is not always the most suitable way to perform sampling. Non-uniform sampling can in some cases actually increase the range of frequencies which can be processed. Nonetheless, for the sake of simplicity, this book will focus only on uniformly sampled signals.

The *second* phase of the digitization process is to quantize the discrete-time signal. The term quantization refers to the approximation of the sampled analog values using a finite set of values. For example, in M -bit quantization there are 2^M allowable levels, and the signal amplitude must be approximated to one of these levels. If, for example, the expected analog voltage limits are ± 3 volts, the input signal can be quantized in steps of $\Delta = 6/2^M$ volts. Typically the voltage quantization levels are chosen to be symmetric around 0 volts.

In the quantization process approximation via rounding or truncation normally occurs and some information is lost. The quantized signal therefore consists of a true signal plus an error signal which is referred to as quantization noise. The power of this noise is dependent on the quantization step, Δ ; with this power being given by $\Delta^2/12$ (see Sect. 3.6.1.1).

The *third* phase of the digitization process involves an encoding of the multibit binary sequence into a practical and convenient form before it is used or transmitted. This process is known as pulse code modulation (PCM), and the resulting

signal is known as a PCM signal. In PCM, each sample is represented by one of 2^M codewords, each of length M bits. If f_s is the sampling rate, the *data rate* would be Mf_s bps. The binary encoding may correspond to the natural binary code (NBC), which is a straightforward decimal-to-binary mapping of the index of the voltage level (between 0 and $2^M - 1$). Alternatively the encoding may use one of the *Gray Codes*, which, because it reduces bit transmission errors, is particularly relevant for long range transmission.

Practical multibit analog-to-digital converter (ADC) tend to perform all three phases mentioned above within the one device: sampling, quantization, and binary encoding. The accuracy (resolution) of multibit ADCs is strongly dependent on the number of bits M , and increasing the number of bits typically improves accuracy.

2.1.1 Digital Systems

Digital systems are hardware or software systems that process digital signals. Most digital systems are built up using binary or “on–off” logic, and so the operation of digital processors can be described using binary arithmetic.

In contrast to the resistors, capacitors and inductors which make up analog systems, the common building blocks for DSP systems are shift registers, adders and multipliers. These building blocks may be realized in either hardware or software, and if implemented in hardware, usually incorporate flip-flops, logic gates and synchronously controlled clocks. Like analog systems, discrete-time and digital systems can be analyzed in either the time or frequency domains. Both forms of analysis are considered later in this book.

2.2 Ideal Sampling and Reconstruction

Sampling is the process of selecting (and possibly storing) the values of a continuous-time signal $x(t)$ at specific time instants which can be indexed by the ring of integers $Z = \{\dots, -2, -1, 0, 1, 2, 3, \dots\}$. As long as the sampling arrangement has been well designed, the discrete-time signal, denoted by $x(n)$, reliably represents the original signal, at least under certain assumptions. In other words, with appropriate design of the sampling strategy, it is possible to process analog signals using digital systems without any loss of information.

2.2.1 Ideal Uniform Sampling

Suppose that an analog signal $x(t)$ is sampled uniformly at a rate of $f_s = 1/T_s$. One can represent this kind of sampling mathematically as multiplication of $x(t)$ by the

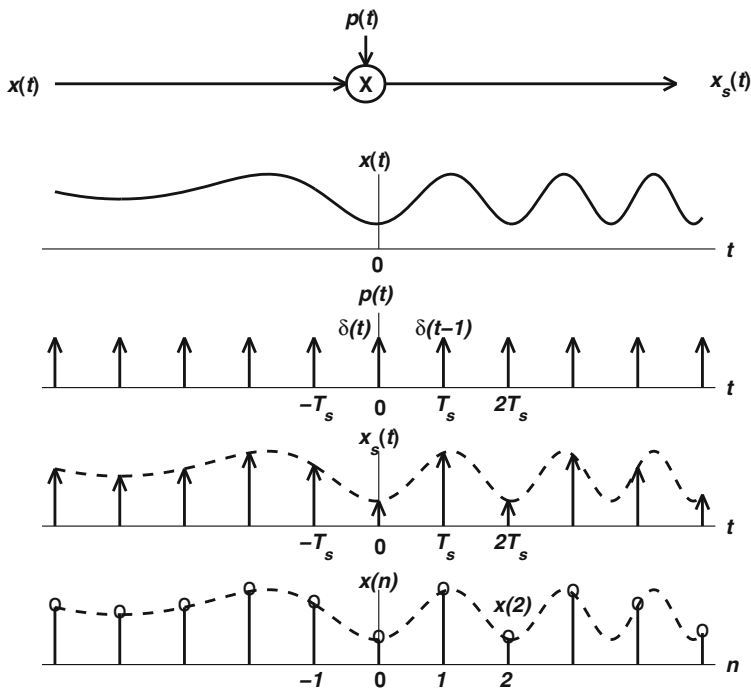


Fig. 2.1 The sampling process in the time domain

impulse train $p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$ (See Fig. (2.1)). The resulting sampled signal in the continuous-time domain is $x_s(t)$, which corresponds to a train of delta functions weighted by the values of $x(t)$ at integer multiples of T_s , i.e., weighted by $x(nT_s)$. The *discrete-time representation* of the same sampled signal is $x(n)$, which is the sequence of actual values of $x(t)$ at the discrete-time instants $t = nT_s$. (Note that $x(n)$ is actually $x(nT_s)$, although, T_s is normally omitted from the notation for simplicity). In practical DSP, the discrete-time representation ($x(n)$) is more commonly used than the continuous-time representation of sampled signals ($x_s(t)$).

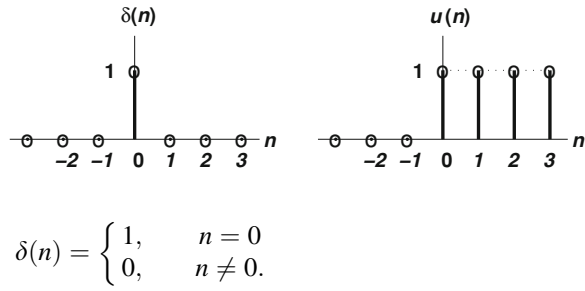
2.2.1.1 Definitions for Some Important Discrete-Time Signals

Formal definitions for two important discrete-time signals are presented below.

The Discrete-Time Unit Pulse

The discrete-time counterpart for the continuous-time unit impulse (delta function) is the discrete-time unit pulse. It is defined as:

Fig. 2.2 The discrete-time delta and unit-step functions



The Discrete-Time Unit-Step Function

The discrete-time counterpart for the continuous-time unit step function is the discrete-time unit step function. Its definition is:

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0. \end{cases}$$

Figure (2.2) shows graphical representations of the above two functions.

Time- and Frequency-Domain Interpretation of Sampling

The sampling process can be illuminated by using various properties from Table 2.1. One of the key results from these *Tables* is that multiplication “ \cdot ” in the time domain is transformed into convolution “ $*$ ” in the frequency domain and vice-versa. With these properties the following time domain and frequency domain mathematical descriptions can be obtained:

Time Domain

$$\begin{aligned} x_s(t) &= x(t) \cdot p(t) \\ &= x(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \\ &= \sum_{n=-\infty}^{\infty} x(nT_s) \delta(t - nT_s) \end{aligned}$$

Table 2.1 Some common time windows

	<i>a</i>	<i>b</i>	<i>c</i>	Comment
Rectangular	1	0	0	Harsh Gibbs effect (strong oscillations)
Hanning	0.5	−0.5	0	Mild oscillations
Hamming	0.5	−0.46	0	Mild oscillations
Blackman	0.42	−0.5	0.08	Mild oscillations

Frequency Domain

$$\begin{aligned}
X(f) &= X(f) * P(f) \\
&= X(f) * f_s \sum_{kn=-\infty}^{\infty} \delta(f - kf_s) \\
&= f_s \sum_{n=-\infty}^{\infty} X(f - kf_s) \delta(f - nT_s)
\end{aligned}$$

where we have been used *Tables* to find that:

$$\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \xrightarrow{\mathcal{F}} f_s \sum_{k=-\infty}^{\infty} \delta(f - kf_s), \quad (2.1)$$

$$X(f) * \delta(f - kf_s) = X(f - kf_s), \quad (2.2)$$

$$x(t) \cdot \delta(t - nT_s) = x(nT_s) \delta(t - nT_s). \quad (2.3)$$

The sampling process in the frequency domain is illustrated in Fig. (2.3). From the above equations and Fig. (2.3) one can deduce that $X_s(f)$ is essentially a collection of repeated versions of $X(f)$ scaled by f_s . These repeated versions are often referred to as “images”. Hence, when a signal $x(t)$ is sampled, its spectrum $X(f)$ is:

1. scaled by f_s and
2. repeated every f_s .

Continuous-time Spectrum

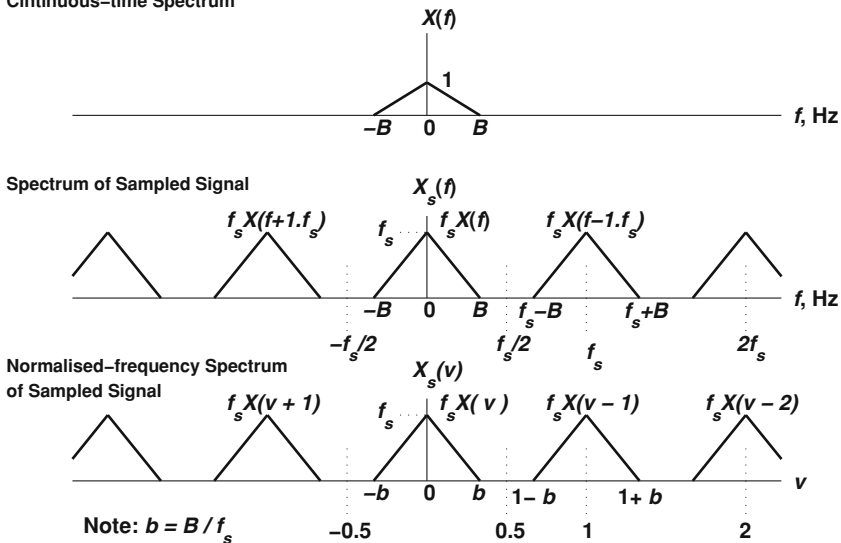


Fig. 2.3 The discrete-time Fourier transform (DTFT) of a sampled signal

Due to the periodicity of $X_s(f)$, there is redundant frequency information in the spectrum or “discrete-time Fourier transform” (DTFT) of $x_s(t)$. All the non-redundant information in $X_s(f)$ is fully contained in the interval $f \in \{-f_s/2, f_s/2\}$ Hz, or equivalently $f \in \{0, f_s\}$. In the literature $X_s(f)$ is normally plotted as a function of the *normalized frequency* $\nu = f/f_s$, or the *normalized radian frequency*, $\Omega = 2\pi\nu = 2\pi f/f_s = \omega T_s$ [see Fig. (2.3)]. Note that ν and Ω have no units, and the period of $X_s(f)$ is 1.

For real signals the frequency content (information) in the interval $[-f_s/2, 0)$ is a reflected copy of the frequency content in $[0, +f_s/2)$, and therefore many authors display only the frequency interval $[0, +f_s/2)$ or the normalized frequency interval $[0, 1/2]$ of discrete-time signal spectra.

Note also that $X_s(f)$ is still continuous in the frequency variable. From a practical perspective one cannot compute $X_s(f)$ for a continuous range of frequency values on a digital computer—one can only compute it for a finite number of frequency positions. For this reason, $X_s(f)$ is usually only evaluated in practice at a finite set of discrete frequencies. This discretization of the frequency domain of the DTFT gives rise to the so-called discrete Fourier transform (DFT), which will be studied later in Sect. 2.4.1.

2.2.2 Ideal Reconstruction

Consider the discrete-time Fourier transform (DTFT) of the sampled signal $x_s(t)$ shown graphically in Fig. (2.3). It is not hard to see that one can reconstruct the original spectrum $X(f)$ from $X_s(f)$ [and hence, the original signal $x(t)$ from $x_s(t)$] by filtering the sampled signal with an ideal low-pass filter whose cutoff frequency is B Hz.

Often in practice reconstruction occurs in a two stage process—the first involves using a digital to analog converter (DAC) which applies a sample and hold function, and the second stage involves applying a low-pass reconstruction filter. Both stages are considered in more detail below.

2.2.2.1 Stage 1

When a digital signal is ready to be converted back to the analog domain, the sequence of digitally stored values is usually fed synchronously into a DAC. Almost all DACs apply a *zero-order hold* (sample-and-hold) function to the sequence of input values [see Fig. (2.4)]. The sample and hold function is effectively a filter with a rectangular impulse response $h(t) = \Pi_{T_s}(t - T_s/2)$. This circuit simply holds the sample $x(nT_s)$ for T_s seconds. The corresponding filter transfer function is a sinc function, as shown in Fig. (2.4).

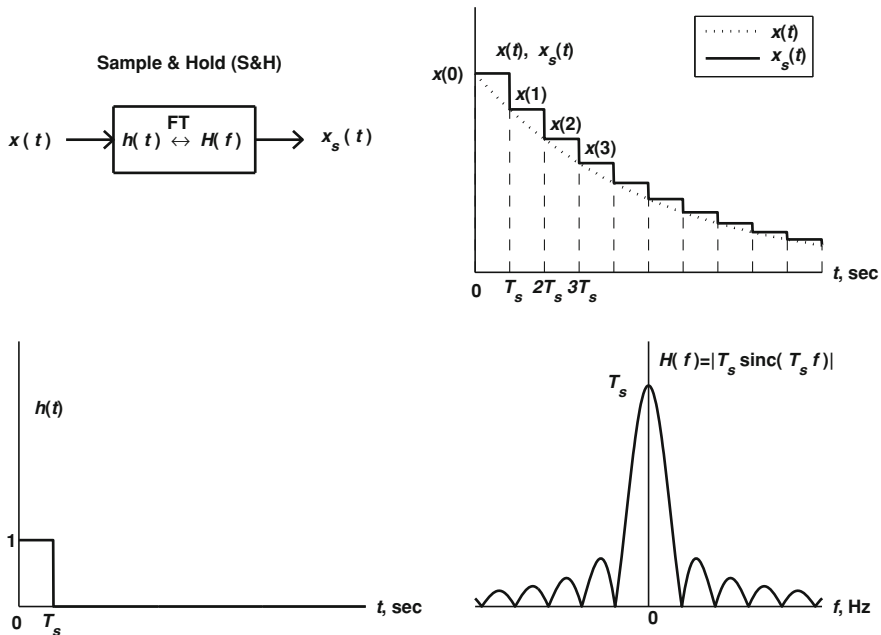


Fig. 2.4 Sample-and-Hold: operation, impulse response, and transfer function

2.2.2.2 Stage 2

Note that the DAC converts the digital signal into an analog stair-step waveform, which contains many high-frequency artefacts. This is because the sample and hold is effectively a filter with a very slow roll off, and it does not fully eliminate the energy in all the unwanted spectral images. To reconstruct a smooth replica of the original signal, the stair-step function must be passed through a low-pass filter, often called an anti-imaging filter. This filter ideally removes all frequency content above the upper frequency limit, B .

For perfect reconstruction of the analog signal to occur within this two stage process, three criteria must be satisfied. The first criteria is that there is no quantization noise in the digital signal. The second criterion is that the sampling frequency should be greater than twice the signal bandwidth B , i.e., greater than twice the highest frequency present in the signal:

$$B \leq \frac{f_s}{2}. \quad (2.4)$$

This requirement is necessary so that the repeating spectra shown in Fig. (2.3) do not “run into other”. The minimum sampling rate needed to avoid reconstruction errors ($f_s = 2B$) is called the Nyquist rate. The third criteria is that the filtering provided by the combination of the sample and hold function and the subsequent

low-pass filter is ideal. That is, those two filters combine to form a filter with a perfectly flat pass-band and a perfectly sharp cutoff at $f = B$ Hz.

In practice none of the three criteria above are met perfectly. There is usually a very small amount of quantization noise, there is typically a small amount of spectral energy in the original analog signal above $f_s/2$ and the reconstruction filter is usually not ideal. The errors due to these imperfections, however, are often small.

2.2.2.3 Frequency Aliasing

If $B > f_s/2$, replicas of the signal spectrum $X(f)$ within the DTFT overlap, and part of the information content from the input signal is lost [See Fig. (2.5) and compare it with Fig. (2.3)]. This overlap is called frequency aliasing. To avoid aliasing one normally band-limits the signal before sampling to remove the high frequency content that may cause aliasing. This can be achieved using a LPF, which is often called an anti-aliasing filter in this scenario.

One technique which is often used in practice to help reduce errors in reconstruction is to increase the sampling rate of the digital system well above the Nyquist rate. When this is done the spectral images in the DTFT are well separated from each other, and when it is necessary reconstruct the analog signal, it is not as difficult to filter out the unwanted images. In particular, the requirements on the sharpness of the anti-imaging filter are relaxed when one increases the sampling rate. Increasing the sampling rate not only relaxes the requirements on the anti-imaging filter, but also on the front-end anti-aliasing filter—i.e., the increase in the $f_s/2$ value effectively allows the transition band of the anti-aliasing filter to be wider.

Figure (2.6) shows a practical signal processing system which incorporates the anti-aliasing filter, the sampling and ADC, the digital processing, the DAC and the anti-imaging filter.

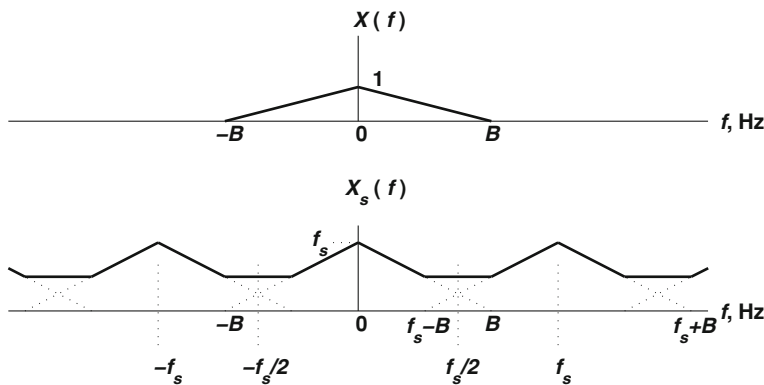
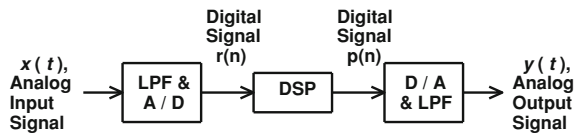


Fig. 2.5 Frequency aliasing $B > f_s/2$

Fig. 2.6 Block diagram of a practical signal processing system



2.3 Time-Domain / Frequency-Domain Representations

Like analog signals and systems, digital signals and systems can be analyzed either in the time-domain or in the frequency domain. The two domains are equivalent, provided that suitable transformations are used. The most common time-to-frequency transformations for continuous-time signals are the Fourier transform and the Laplace transform. The counterparts of these transforms for sampled signals are the discrete-time Fourier transform (DTFT) and the z-transform. The DTFT is useful for analyzing the frequency content of signals, while the z-transform is useful for both analyzing the frequency content of signals and analyzing the stability of systems. The DTFT and z-transforms will be defined and studied in more detail in [Sects. 2.3.2.2](#) and [2.3.3](#).

2.3.1 Time-Domain Representation of Digital Signals and Systems

This section considers the representation and analysis of digital signals and systems. Fundamental to time domain analysis of discrete-time signals is discrete-time convolution, which is defined in what follows.

2.3.1.1 Discrete Linear Convolution

If $x(n)$ and $y(n)$ are two discrete signals, their discrete linear convolution $w(n)$ is given by:

$$w(n) = x(n) * y(n) = \sum_{k=-\infty}^{\infty} x(k)y(n-k) .$$

Note that k is a dummy variable of summation. The above formula is similar to that of continuous-time linear convolution, except that summation is used instead of integration.

If both $x(n)$ and $y(n)$ are finite signals of lengths N_1 and N_2 samples, respectively, the length of $w(n)$ is finite and is given by $L = N_1 + N_2 - 1$. Hence, if $x(n)$ and $y(n)$ are of equal length N , then the result of the convolution is $L = 2N - 1$ samples long.

Example (1) If $x(n) = (0.8)^n u(n)$ and $y(n) = (0.4)^n u(n)$, then their convolution is:

$$w(n) = x(n) * y(n) = \sum_{k=-\infty}^{\infty} x(k)y(n-k) \quad (2.5)$$

Since the variable of summation is k , x and y should be re-expressed as functions of k :

$$\begin{aligned} y(k) &= \begin{cases} (0.4)^k, & k \geq 0 \\ 0, & k < 0; \end{cases} \\ x(k) &= \begin{cases} (0.8)^k, & k \geq 0 \\ 0, & k < 0 \end{cases} \\ \therefore y(n-k) &= \begin{cases} (0.4)^{n-k}, & n-k \geq 0 \Rightarrow \therefore k \leq n \\ 0, & n-k < 0 \Rightarrow \therefore k > n \end{cases} \end{aligned}$$

Examination of Fig. (2.7) reveals that the product $x(k)y(n-k)$, is non-zero only when $n \geq 0$. Hence it follows that:

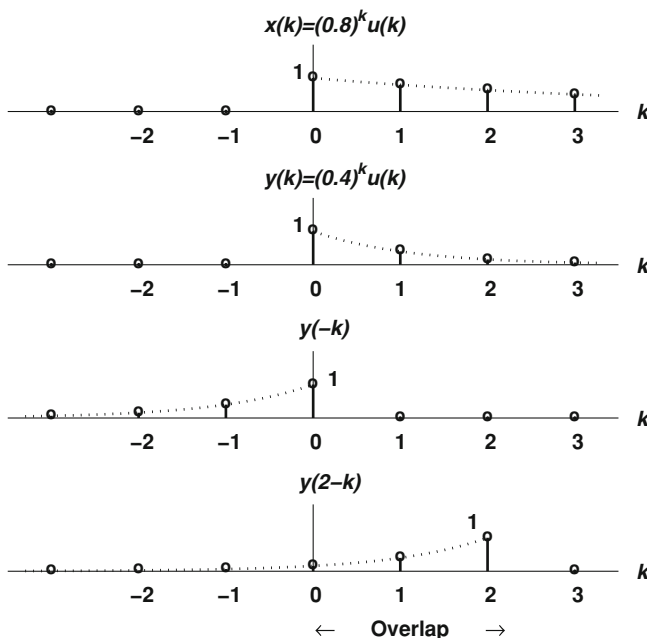


Fig. 2.7 Convolution of $x(n) = (0.8)^n u(n)$ with $y(n) = (0.4)^n u(n)$

$$\begin{aligned}
w(n) &= \sum_{k=0}^n (0.8)^k (0.4)^{n-k}; \quad n \geq 0 \\
&= (0.4)^n \sum_{k=0}^n 2^k = (0.4)^n \frac{1 - 2^{n+1}}{1 - 2} \quad [\text{Tables, Formula 10}] \\
&= (0.4)^n (2^{n+1} - 1)u(n).
\end{aligned}$$

Example (2) If there are two finite duration signals $x(n) = \{\hat{2}, 3, 4\}$ and $y(n) = \{-\hat{1}, 5\}$, where “ $\hat{\cdot}$ ” indicates that the sample is taken at $n = 0$, then their convolution can be found directly from the general formula as follows:

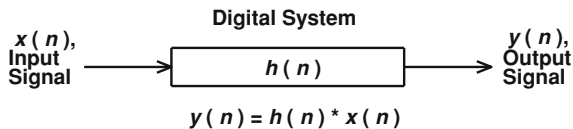
$$\begin{aligned}
w(n) &= \sum_{k=-\infty}^{\infty} x(k)y(n-k) \quad (\text{General formula}) \\
w(0) &= \sum x(k)y(0-k) = x(0)y(0) = (2)(-1) = -2, \\
w(1) &= \sum x(k)y(1-k) = x(0)y(1-0) + x(1)y(1-1) = (2)(5) + (3)(-1) = 7, \\
w(2) &= \sum x(k)y(2-k) = x(0)y(2-0) + x(1)y(2-1) + x(2)y(2-2) \\
&= (2)(0) + (3)(5) + (4)(-1) = 11, \\
w(3) &= \sum x(k)y(3-k) \\
&= x(0)y(3-0) + x(1)y(3-1) + x(2)y(3-2) + x(3)y(3-3) \\
&= (4)(5) = 20, \\
w(4) &= w(5) = \dots = 0. \quad (\text{Note that } L = N_1 + N_2 - 1 = 3 + 2 - 1 = 4)
\end{aligned}$$

2.3.1.2 Mathematical Representation of Digital Signals and Systems in the Time Domain

While analog signals are normally specified as a function of continuous-time, uniformly sampled digital signals are normally represented in the time domain as a function of the sample number (or the sample count integer). Digital systems are typically characterized in ways that are similar to those used in the continuous-time domain. Rather than being represented by a continuous-time impulse response, a digital system is fully specified by a discrete-time impulse response. This impulse response is the output of the digital system when the input is the discrete delta function, $\delta(n)$. Within this book, the discrete-time impulse response is denoted by $h(n)$.

The system input/output (I/O) relationship is specified by the discrete linear convolution of the impulse response $h(n)$ and the input signal $x(n)$:

Fig. 2.8 Discrete-time domain representation of a digital system



$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (2.6)$$

Note For *causal* digital systems $h(n) = 0$ for $n < 0$.

A graphical illustration of a digital system is shown in Fig. (2.8).

Eigenfunctions of LTI Digital Systems

If the input signal to an LTI digital system is $x(n) = e^{j\omega n T_s} = e^{jn\Omega}$, then the output is given by:

$$y(n) = x(n) * h(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)e^{j(n-k)\Omega} = e^{jn\Omega} \sum_{k=-\infty}^{\infty} h(k)e^{-jk\Omega}.$$

If one defines $H(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} h(k)e^{-jk\Omega}$, then for an input of $e^{jn\Omega}$ one obtains the output $y(n) = e^{jn\Omega}H(e^{j\Omega})$. Hence, $e^{jn\Omega}$ is an eigenfunction, and the associated eigenvalue is $H(e^{j\Omega})$ (Compare with the analogous result obtained for continuous-time systems in Sect. 1.2.3.4).

Analyzing the Stability of Digital Systems in the Time Domain

For a digital system to be BIBO stable, the output $y(n)$ should be bounded when the input $x(n)$ is bounded, i.e.,

$$|y(n)| < \infty \text{ when } |x(n)| < A (\text{which is a finite constant}), \forall n.$$

Using the inequality $|a + b| \leq |a| + |b|$, it follows that

$$\left| \sum_{k=-\infty}^{\infty} x(k)h(n-k) \right| \leq \sum_{k=-\infty}^{\infty} |x(k)||h(n-k)|.$$

If $|x(k)| < A \forall k$, then

$$|y(n)| = \left| \sum_{k=-\infty}^{\infty} x(k)h(n-k) \right| \leq A \sum_{k=-\infty}^{\infty} |h(n-k)| \leq A \sum_{m=-\infty}^{\infty} |h(m)|,$$

where $m = n - k$. Hence, a digital system is BIBO-stable if $\sum_{k=-\infty}^{\infty} |h(k)| < \infty$. (Compare with the condition for stability of analog systems found in Sect. 1.2.3.5).

2.3.2 Frequency-Domain Representation of Digital Signals and Systems

2.3.2.1 Discrete-Time Fourier Series for Periodic Digital Signals

In the analog domain the continuous-time signal and the Fourier Series are related according to:

$$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{+j2\pi k f_o t} \Leftrightarrow X_k = \frac{1}{T_o} \int_0^{T_o} x(t) e^{-j2\pi k f_o t} dt \quad (2.7)$$

If the analog signal $x(t)$ is sampled with a rate of f_s (sampling interval $= T_s = 1/f_s$), a discrete-time signal $x(n)$ is obtained:

$$x_s(t) = x(n) = \sum_{k=-\infty}^{\infty} x(t = nT_s) \delta(t). \quad (2.8)$$

It will be assumed that the period of the above discrete-time signal is N samples, where $N = T_o/T_s = f_s/f_o$. If $x_s(t)$ is fed into the formula for the Fourier Series in [Sect. 1.2.3.1](#) one obtains:

$$X_k = X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \quad [\text{DFS pair}]$$

The above expression is obtained with the result that $\int_0^{T_o} \delta(t) dt = 1$. The equation linking time domain signals with Fourier series for discrete-time signals is therefore given by:

$$x(n) = \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N} \Leftrightarrow X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} [\text{DFS pair}].$$

Note that, unlike the continuous-time FS, the summation for $x(n)$ in the DFS does not need to go from $-\infty$ to ∞ due to the periodicity of both $X(k)$ and the discrete exponential $e^{j2\pi kn/N}$ in k . That is, the DFS coefficients are *periodic* (with period N).

2.3.2.2 The Discrete-Time Fourier Transform for Non-Periodic Digital Signals

Recall from [Sect. 2.2](#) that the formula for the discrete-time Fourier transform (DTFT) for non-periodic signals is:

$$\begin{aligned}
X_s(f) &= X(f) * P(f) \\
&= X(f) \cdot f_s \sum_{kn=-\infty}^{\infty} \delta(f - kf_s) = \sum_{k=-\infty}^{\infty} X(f - kf_s) .
\end{aligned}$$

The DTFT consists of a sum of scaled and infinitely repeated versions of the spectrum of the original analog signal, $x(t)$. This is depicted in Fig. (2.3).

2.3.3 The z-Transform

The Laplace transform (LT) was previously seen to be very useful for stability analysis of continuous-time signals. One can obtain similar stability analysis capabilities for discrete-time signals by using a discrete-time counterpart to the LT. If one substitutes the expression for a discrete-time signal, $x_s(t)$ into the expression for the LT in Sect. 1.2.3.3 and simplifies, one obtains the *discrete-time Laplace transform (DTLT)*:

$$X(s) = X(e^{sT_s}) = \sum_{n=-\infty}^{\infty} x(n) e^{-nsT_s}. \quad (2.9)$$

The DTLT is, like the DTFT, periodic, and can be shown to have the equivalent alternative expression:

$$X_s(s) = f_s \sum_{k=-\infty}^{\infty} X(s - jk\omega_s).$$

where $X(s)$ is the LT of the original analog signal. It is convenient to define a new variable $z = e^{sT_s}$, which can be substituted into (2.9) to obtain the following transform:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}. \quad (2.10)$$

The above relation is called the z-transform (ZT). It is evident that the z-transform is reduced to the Fourier transform if one substitutes $z = e^{j\Omega}$.

Unlike the DTFT the ZT is *not necessarily periodic* in the frequency variable. This non-periodicity is due to the additional factor α in the exponent (e^{sT_s}) which results in a *decaying or expanding factor* (e^{α}) in the overall transform.

Under certain conditions (that are normally satisfied in practical signals and systems), the ZT exists as a pair, i.e., there is a z-transform and an inverse z-transform (IZT). Like the LT, the ZT has a region of convergence (ROC).

Computation of the IZT can be quite involved as it requires integration in the complex z -plane. For this reason, it is advisable to use *Tables* to determine the IZT.

2.3.3.1 The Single-Sided ZT

As with the LT, one can define double-sided and single-sided transforms. Mimicking the approach used for the LT, this book will focus only on the single-sided ZT, which is defined as:

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n}. \quad (2.11)$$

Example (1) If $x(n) = a^n u(n)$, then its ZT is given by:

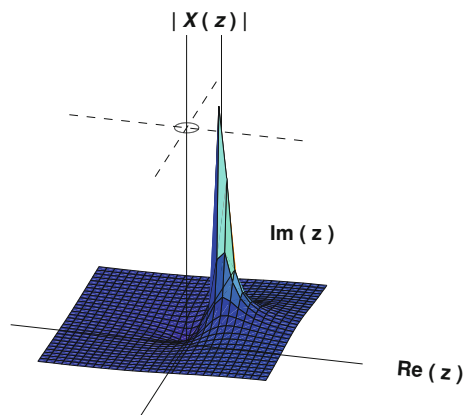
$$\begin{aligned} X(z) &= \sum_{n=0}^{\infty} x(n)z^{-n} = \sum_{n=0}^{\infty} a^n z^{-n} = \sum_{n=0}^{\infty} (az^{-1})^n \\ &= \frac{1 - (az^{-1})^{\infty}}{1 - az^{-1}} \text{ [from Tables, Formula 10]} = \frac{1}{1 - az^{-1}} = \frac{z}{z - a} \text{ (if } |z| > a). \end{aligned}$$

Figure (2.9) shows this function for $a = 3$.

Example (2) If $x(n) = u(n)$, then from *Example (1)*, its ZT is given by $z/(z - 1)$ for $|z| > 1$, i.e., the ROC is $|z| > 1$.

Example (3) If $x(n) = \delta(n)$, then its ZT is given by $X(z) = \sum_{n=0}^{\infty} \delta(n)z^{-n} = 1 + 0 + 0 + \dots = 1$.

Fig. 2.9 A plot of the magnitude of the z -transform $X(z) = z/(z - a)$ for $a = 3$



2.3.3.2 The Time-Shift Property of the ZT

Assume that $y(n) = x(n - M)$, i.e., assume that $y(n)$ is a time delayed version of $x(n)$ with the delay being M samples. Then:

$$Y(z) = \sum_{n=0}^{\infty} y(n)z^{-n} = \sum_{n=0}^{\infty} x(n - M)z^{-n}.$$

Letting $k = n - M$ and assuming that $x(k) = 0$ for $k < 0$ gives

$$Y(z) = \sum_{k=-M}^{\infty} x(k)z^{-k-M} = z^{-M} \sum_{n=0}^{\infty} x(k)z^{-k} = z^{-M}X(z).$$

That is, a delay of M samples in the time-domain corresponds to a multiplication by z^{-M} in the z -domain.

2.3.3.3 Relationship Between the FT and ZT of a Discrete-Time Signal

It has been seen previously that to retrieve the FT from the LT, one substitutes $s = j\omega$. In similar fashion one can recover the DTFT from the ZT by putting $z = e^{sT_s} = e^{j\omega T_s}$. Since $|e^{j\omega T_s}| = 1$, the DTFT is effectively the ZT evaluated along the unit circle of the complex z -plane, i.e., along $|z| = 1$.

It should be noted that FT plots are in general 2-D, while LT and ZT plots are 3-D. The extra dimension is due to the additional variable, α in the LT and ZT formulations.

2.3.3.4 Relationship Between the LT and the ZT for Discrete-Time Signals

Recall that the ZT is obtained from the LT by making the assignment $z = e^{sT_s} = e^{(\alpha + j\omega)T_s}$. Since $z = e^{sT_s} = e^{(\alpha + j\omega)T_s} = e^{\alpha T_s} e^{j\omega T_s}$, the magnitude of z is $r = |z| = e^{\alpha T_s}$ and the phase of z is:

$$\theta = \tan^{-1} \left[\frac{\text{Im}(z)}{\text{Re}(z)} \right] = \tan^{-1} \left[\frac{\sin(\omega T_s)}{\cos(\omega T_s)} \right] = \omega T_s$$

(using Euler's formula to expand $e^{j\omega T_s}$).

Hence, the $z = e^{sT_s} = e^{(\alpha + j\omega)T_s}$ assignment inherent in the Laplace to z -transform mapping causes the imaginary axis in the s -plane (i.e., $s = 0 + j\omega$) to map to the *circumference of the unit circle* in the z -plane (i.e., $|z| = 1$). This is illustrated graphically in Fig. (2.10). The left half of the s -plane (with $\alpha < 0$) is transformed into the interior of the unit circle in the z -plane (i.e., $|z| < 1$), while the right half of the s -plane is transformed into the exterior of the unit circle (i.e., $|z| > 1$).

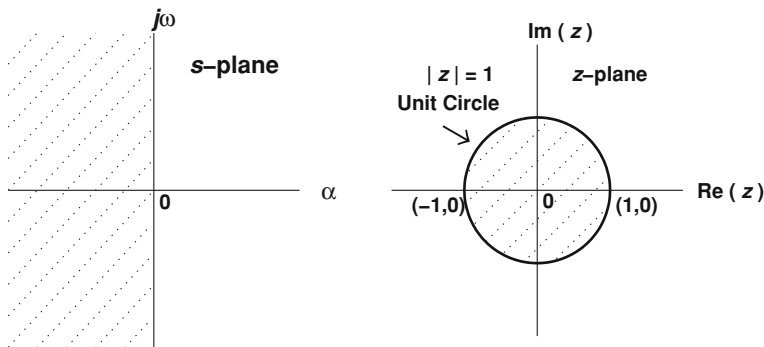


Fig. 2.10 Mapping the s -plane onto the z -plane using $z = e^{sT_s}$

Note that this relation is not a general relation between analog and digital representations. It is only applicable to s -plane and z -plane representations of discrete-time signals, since it is based on the relation between the DTLT and ZT.

2.3.4 Mathematical Representation of Signals and Systems in the Frequency Domain

A discrete-time signal $x(n)$ can be represented by its DTFT in the frequency domain. Similarly, a digital system with impulse response $h(n)$ can be represented in the frequency domain by its transfer function, $H_s(f) = H(e^{j\omega T_s}) = H(e^{j\Omega})$ which is the DTFT of its impulse response $h(n)$ [See Fig. (2.11)].

Recall that characterization of the system function is achieved in the time domain via the impulse response. The output of a system can be obtained for a given input via discrete linear convolution of the impulse response with the input:

$$y(n) = h(n) * x(n)$$

In the frequency domain, the *system function* information is contained in the transfer function. The system output for a given input is found by multiplying the transfer $H_s(f)$ by the DTFT of the input signal $X_s(f)$:

$$Y_s(f) = H_s(f) \cdot X_s(f),$$

where $Y_s(f) = \text{DTFT}[y(t)]$ and $X_s(f) = \text{DTFT}[x(t)]$. This relation can also be written with the alternative notation $Y(e^{j\Omega}) = H(e^{j\Omega}) \cdot X(e^{j\Omega})$, where $\Omega = \omega T_s$. A similar relation is obtained in the z -domain:

$$Y(z) = H(z) \cdot X(z).$$

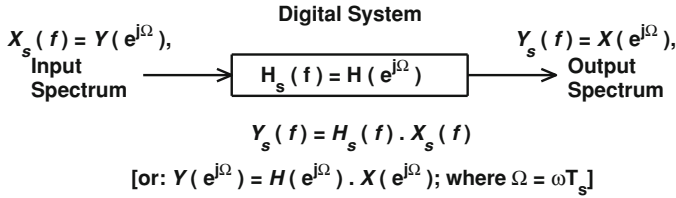


Fig. 2.11 Frequency-domain representation of a discrete-time system

Again, analogously to the continuous-time case, convolution in the discrete-time domain is transformed into multiplication in both the frequency domain and the z -domain:

$$x(n) * y(n) \xrightarrow{Z} X(z) \cdot Y(z) \quad (2.12)$$

$$x(n) \cdot y(n) \xrightarrow{Z} X(z) * Y(z) \quad (2.13)$$

(See also *Tables*, z -Transform Pairs and Theorems).

2.3.4.1 Relationship Between the ZT Transfer Function and the Frequency Response

The (periodic) frequency response $H_s(\omega) = H(e^{j\omega T_s})$ of a digital system can be obtained from its ZT-transfer function $H(z)$ with the substitution $z = e^{j\omega T_s}$ as follows:

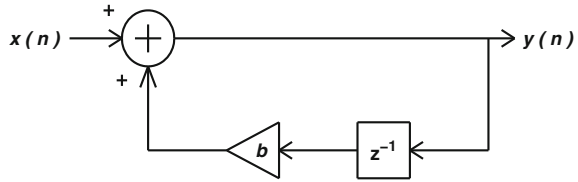
$$H(e^{j\omega T_s}) = H(z)|_{z=e^{j\omega T_s}}.$$

2.3.4.2 Stability of Digital Systems in the z -Domain

It has been shown previously that a digital system is BIBO-stable if its impulse response is absolutely summable, i.e., if $\sum_{k=-\infty}^{\infty} |h(k)| < \infty$. Practically it is often difficult to analyze the system stability in this way. Equivalently, a *causal* digital system is BIBO-stable *if and only if all the poles of its z -transfer function $H(z)$ are inside the unit circle* (i.e., $|z_p| < 1$, where z_p is the location of the p th pole in the z -plane). [Note that this condition is for causal digital systems only, otherwise the condition would be that the ROC of $H(z)$ should contain the unit circle].

Example (1) If the system impulse response is $h(n) = a^n u(n)$, then its z -transfer function is $H(z) = z/(z - a)$ [from *Tables*. This system has a zero at $z = 0$, and a pole at $z = a$. It is BIBO-stable if $|a| < 1$, i.e., if the pole is within the unit circle

Fig. 2.12 An example of a recursive digital system (leaky integrator)



Example (2)) The difference equation that describes the recursive system in Fig. (2.12) is given by:

$$y(n) = x(n) + by(n-1), \quad (2.14)$$

where, b is a multiplicative constant (gain). Taking the ZT of both sides of (2.14) yields:

$$Y(z) = X(z) + bz^{-1}Y(z).$$

Re-arranging terms leads to

$$\begin{aligned} [1 - bz^{-1}]Y(z) &= X(z). \\ \therefore H(z) = \frac{Y(z)}{X(z)} &= \frac{1}{1 - bz^{-1}} = \frac{z}{z - b}. \end{aligned}$$

From *Tables* (z-Transform Pairs), the IZT is $h(n) = b^n u(n)$.

The system frequency response is:

$$\begin{aligned} H(\omega) &= H(z)|_{z=\exp(j\omega T_s)} \\ &= \frac{e^{j\omega T_s}}{e^{j\omega T_s} - b} = \frac{e^{j2\pi f T_s}}{e^{j2\pi f T_s} - b} = \frac{e^{j2\pi f / f_s}}{e^{j2\pi f / f_s} - b} \\ &= \frac{e^{j2\pi v}}{e^{j2\pi v} - b} = \frac{e^{j\Omega}}{e^{j\Omega} - b} \end{aligned}$$

where $v = ff_s$ and $\Omega = 2\pi v$ are the normalized cyclic frequency and normalized angular frequency, respectively. The magnitude and phase responses are:

$$\begin{aligned} |H(\omega)| &= |H(e^{j\omega T_s})| = \frac{|e^{j\omega T_s}|}{|e^{j\omega T_s} - b|} = \frac{1}{|[\cos(\omega T_s) - b] - j \sin(\omega T_s)|} \\ &= \frac{1}{\sqrt{[\cos(\omega T_s) - b]^2 + [\sin(\omega T_s)]^2}} \angle H(\omega) \\ &= \angle H(e^{j\omega T_s}) = -\tan^{-1} \left[\frac{b \sin(\omega \cdot T_s)}{1 - b \cos(\omega \cdot T_s)} \right] \end{aligned}$$

As a special case, let $b = 0.5$ and $f_s = 1$ Hz (i.e., $T_s = 1$ s). At $f = 0.1$ Hz (i.e., $\omega = 2\pi(0.1) = 0.6283$ rad/s), the magnitude response is $|H(e^{j\omega T_s})| = 1.506$ and

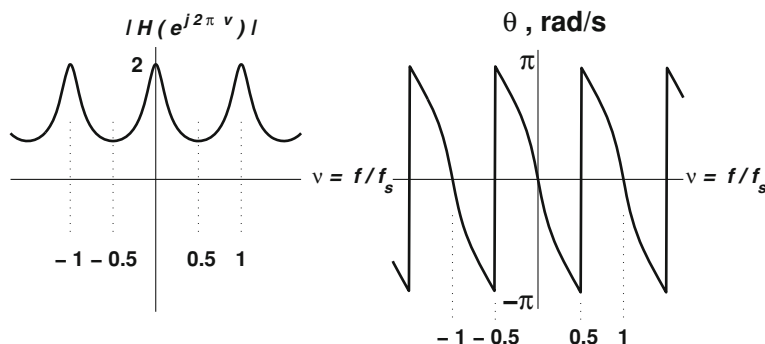


Fig. 2.13 Magnitude and phase response of a leaky integrator

the phase response is $\angle H(e^{j\omega T_s}) = -0.458$ rad. Figure (2.13) shows the magnitude and phase response of this system as a function of the normalized frequency. If one needs to recover the actual response versus frequency from the normalized frequency response one simply scales the frequency axis by f_s .

From Fig. (2.13) one can see that the shape of the transfer function is that of a LPF; the circuit is in fact what is sometimes referred to as a leaky integrator. If $b = 1$, it would be an integrator, as will be seen later.

2.4 A Discrete-Time and Discrete-Frequency Representation

The DTFT of a discrete-time signal $x(n)$ is still continuous in the frequency variable, and hence it is not possible to implement with practical digital technologies such as computers. It is necessary, then, to find a discrete-time and discrete-frequency representation for practical analysis. This doubly discrete representation will be referred to as the *Discrete Fourier Transform (DFT)*.

2.4.1 The Discrete Fourier Transform

In practice one has to restrict oneself to computing the Fourier transform at a limited number of representative sample frequencies. It was seen previously that with appropriate precautions no information is lost provided that certain requirements on the sampling rate are met. In particular, it is necessary that the sample rate in the time domain is at least twice as high as the highest frequency component present in the analog signal. It will be seen here that there is an analogous result for frequency domain sampling—one does not lose any information by sampling in the frequency domain, provided that certain conditions are met on the frequency sampling rate.

Consider now the problem of sampling the DTFT in the frequency domain. It is assumed that the frequency sampling is uniform, and that the spacing between samples is F_s . Now it is critical that this frequency domain sampling does not lead to information loss. That is, it is necessary for the time domain signal to still be perfectly recoverable, despite this sampling.

It is now required to determine the maximum frequency sampling interval which ensures no loss of information. Assume initially that this rate is $F_s = f_s/N$. This choice implies that the number of samples in the frequency domain is the same as the number of samples in the time domain. Then the resulting DFT is defined by:

$$X_s(k) = \sum_{n=0}^{N-1} x_s(n) e^{-j2\pi kn/N}.$$

Now consider the inverse Fourier Transform of $X_s(k)$. By using an analysis very similar to that done in [Sect. 2.3.2.1](#) it is possible to show that the inverse is given by:

$$x_p(n) = \frac{1}{N} \sum_{k=-\infty}^{\infty} X_s(k) e^{j2\pi kn/N} \iff X_s(k) = \sum_{n=0}^{N-1} x_p(n) e^{-j2\pi kn/N}$$

Now it is important to realize that both $X_s(k)$ and $e^{-j2\pi kn/N}$ are periodic, and because of this periodicity, $x_p(t)$ is also periodic, with period N . This indicates that just as sampling in the time domain causes periodicity in the frequency domain, so sampling in the frequency domain causes periodicity in the time domain. Furthermore, with a frequency sampling interval of $F_s = f_s/N$ the periodicity is seen to be N , which is just adequate to prevent the time domain images from “running into each other”. That is, the frequency sampling interval of $F_s = f_s/N$ is just enough to prevent time-domain aliasing. If the sampling interval were any greater aliasing in the time domain would occur.

In summary then, sampling in the time and frequency domains causes repetition in both domains. The DFT is normally obtained from the DTFT by sampling at a rate of $F_s = f_s/N$, because this is just enough to avoid time-domain aliasing. With this sample rate the number of samples in both the time and frequency domains is N . Although the time and frequency domains both repeat doubly discredited systems, it is common to only display one image. With this in mind the usual definitions for the DFT and inverse DFT (IDFT) are:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N} \iff X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}. \quad (2.15)$$

Note that

$$IDFT\{DFT[x(n)]\} = 1; \quad (2.16)$$

that is, the DFT and IDFT are reversible transforms.

2.4.1.1 Approximation of the FT Using DFT

The DFT can be used to approximate the original (continuous-time) FT. The quality of the approximation depends largely on the sampling rate used in the time domain. As long as the time domain sampling rate is greater than the highest frequency present in the original analog signal, and the frequency domain sampling rate is $F_s = f_s/N$, the approximation is perfect.

The effectiveness of the DFT for approximating the Fourier Transform derives from the fact that the DFT is simply a sampled version of the DTFT, which is in turn a scaled and periodic version of the FT (see [Sect. 2.3.2.2](#)).

2.4.1.2 Relationship Between the DFT and the DFS Coefficients

A simple comparison between the DFS of a periodic sequence (with period N) and the DFT of one period of this sequence reveals that the two are related by a simple multiplicative constant:

$$X_{k,DFS} = X_{k,DFT}/N, \quad (2.17)$$

where $X_{k,DFS}$ is the pertinent DFS coefficient and $X_{k,DFT}$ is the corresponding DFT sample. This is reminiscent of a similar relation in the analog domain, where the FS coefficients of a periodic signal (of period T_o) are related to the FT of a single period of the same signal according to:

$$X_k = X_{1p}(f)/T_o \mid_{f=kf_o}, \quad (2.18)$$

where X_k is the pertinent FS coefficient DFS and $X_{1p}(f) \mid_{f=kf_o}$ is the corresponding FT value.

2.4.1.3 The Fast Fourier Transform

Calculation of the Discrete-Fourier Transform directly according to the definition in (2.15) requires of the order of N^2 computations, i.e., it requires $O(N^2)$ arithmetic operations. These calculations cannot normally be done in real-time, as is required for many practical applications. For this reason many scientists and engineers have sought alternative techniques for rapidly calculating the DFT. The first to devise an efficient algorithm was probably Gauss, but his work went largely un-noticed until about 1990. In 1965, however, the so-called Fast Fourier Transform (FFT) was re-invented and popularized. [see J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965]. In contrast to calculation via the direct definition, the FFT algorithm can be implemented in $O(N \log(N))$ operations. This algorithm provided not only economy of time in run-time operation, but also economy of computational and storage elements in hardware for DFT implementation.

The FFT algorithm can generally support real-time processing (at least on modern computers and DSP chips).

MATLAB the Fast Fourier Transform algorithm is available on MATLAB as `fft`.

2.4.1.4 Circular Convolution and Its Relation to the Linear Convolution

In time and frequency discretized systems the time domain signal and the frequency domain signal are both periodic. Within these doubly discretized systems, it is not only the input signal which is periodic, but also the impulse response and frequency transfer function. To appreciate the implications of this last fact, consider two time-domain signals $x(n)$ and $h(n)$ which are both periodic. Their *linear* convolution

$$x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (2.19)$$

diverges in general (i.e., does not exist). This is because both signals are infinitely long and so the sum within (2.19) can easily be infinite. With periodic signals, therefore, it is more natural to define a modified form of convolution known as circular convolution. As will be seen subsequently, this is the kind of convolution which is implemented implicitly in doubly discretized systems.

Assuming that two signals have *identical periods* of length N , then one considers only one period of each signal and defines circular convolution as:

$$x(n) \otimes h(n) = \sum_{k=0}^{N-1} x(k)h[(n-k)_N],$$

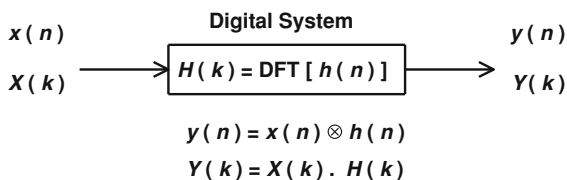
where $p_N = p$ modulo N for any integer p . The circular convolution $x(n) \otimes h(n)$ is also periodic with the same period N .

Recall that the linear convolution of two non-periodic finite-length signals $x(n)*h(n)$ has length $L = N_x + N_y - 1$. If one desires therefore to have $x(n) \otimes y(n) = x(n) * h(n)$ over one period of the periodic signals $x(n)$ and $h(n)$, then one has to artificially extend the length of both signals by zero-padding both $x(n)$ and $h(n)$ to be of length $L = N_x + N_y - 1$. That is, one adds extra samples to $x(n)$ and $h(n)$, with these samples having the value 0.

2.4.1.5 I/O Relations Using Circular Convolution and the DFT

One of the key areas of application for the DFT is digital filtering. In DFT based digital filtering one has an input signal $x(n)$ and an impulse response $h(n)$, and the filtered output $y(n)$ is given by the convolution of $x(n)$ and $h(n)$. Implementation of the filtering in the time domain, however, requires $O(N^2)$ operations, assuming both $x(n)$ and $h(n)$ have N samples. One can take advantage of the efficiency of the

Fig. 2.14 I/O relations in a digital system



FFT algorithm and implement the filtering in the frequency domain instead. This alternative is discussed in the following paragraphs (Fig. 2.14).

Since convolution in the time domain is equivalent to multiplication in the frequency domain, it is tempting to

1. take the FFTs of $x(n)$, and $h(n)$ to obtain $X(k)$ and $H(k)$, respectively,
2. multiply $X(k)$ and $H(k)$ together to obtain $Y(k)$, and
3. then take the inverse FFT to obtain $y_c(n)$.

This three-step procedure, however, would yield the wrong result in general. To see why the procedure is flawed, imagine that $x(n)$ and $h(n)$ are both N samples long. Then $X(k)$, $H(k)$, $Y(k)$ and $y_c(n)$ would all be N samples long as well. Since $y_c(n)$ is supposed to be the convolution of $x(n)$ and $h(n)$, however, it should be $N + N - 1$ samples long rather than N samples.

The reason for the flaw in the three step procedure above is that multiplication in the DFT (or FFT) domain implicitly corresponds to *circular* convolution in the time domain. For true filtering to occur, one must have *linear* convolution rather than circular convolution, and so one must *zero-pad* both $x(n)$ and $h(n)$ to be of at least length $L = N_x + N_h - 1$, where N_x is the length of $x(n)$ and N_h is the length of $h(n)$. If one performs this zero-padding, the circular convolution inherently implemented in the FFT domain becomes equivalent to linear convolution in the time domain.

2.5 Signal Correlation, Power, and Energy

2.5.1 Definitions

This subsection presents formulae for correlation, power, and energy of discrete-time signals. These formulae are analogous to those for analog signals, but with integrations changed to summations. The formulae are presented below.

2.5.1.1 Autocorrelation of Non-Periodic Discrete-Time Energy Signals

$$R_x(k) = \sum_{n=-\infty}^{\infty} x(n)x^*(n+k).$$

2.5.1.2 Autocorrelation for Periodic Discrete-Time Power Signals

$$R_x(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)x^*(n+k),$$

where N is the signal period.

2.5.1.3 Energy in Non-Periodic Discrete-Time Energy Signals

$$R_x(k) = \sum_{n=-\infty}^{\infty} T_s |x(n)|^2$$

Compare the above formula with the corresponding formula for analog signals:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt.$$

2.5.1.4 Power in Periodic Discrete-Time Power Signals

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

2.5.1.5 Parseval's Theorem

$$\text{For periodic signals: } P = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 = \sum_{k=0}^{N-1} \underbrace{|X(k)|^2}_{\text{PSD}}.$$

$$\text{For non-periodic signals: } E = \sum_{n=-\infty}^{\infty} T_s |x(n)|^2 = \int_0^{f_s} \underbrace{T_s^2 |X_s(f)|^2}_{\text{ESD}} df,$$

where $X_s(f) = X(e^{j2\pi f} T_s)$ is the DTFT.

2.5.1.6 The Wiener-Kinchin Theorem

For periodic signals: $R_x(k) \longleftrightarrow \underbrace{|X_k|^2}_{\text{PSD}}$.

For non-periodic signals: $R_x(k) \longleftrightarrow \underbrace{T_s^2 |X_s(f)|^2}_{\text{ESD}}$.

2.6 Digital Filters and Their Applications

The advancement of digital computers during the 1960s paved the way for many analog electronic circuits to be emulated in digital computers. The advantages of this kind of digital emulation are many—greater accuracy, greater flexibility, lower cost, lower power requirements, greater reproducibility, etc. This section of the book focuses on a particular type of digital emulation—namely the emulation of conventional filters with digital computer hardware. This type of emulation is commonly referred to as digital filtering.

2.6.1 Ideal Digital Filters

Ideal digital LP, HP, BP, and BS filters can be defined by simple analogy with analog filters. In exploiting the similarities between digital and analog filters, however, it is also important to keep in mind that there are some key differences. One of the key differences is that there is a periodicity in the frequency response of digital filters.

2.6.1.1 Mathematical Formulation

The Digital LPF

The transfer function of a digital LPF over the principal frequency domain $(-f_s/2, f_s/2)$ is given by

$$H_L(f) = H_L(e^{j2\pi f T_s}) = \begin{cases} 1, & 0 \leq |f| \leq f_c \\ 0, & f_c \leq |f| \leq f_s/2. \end{cases}$$

[see Fig. (2.15)].

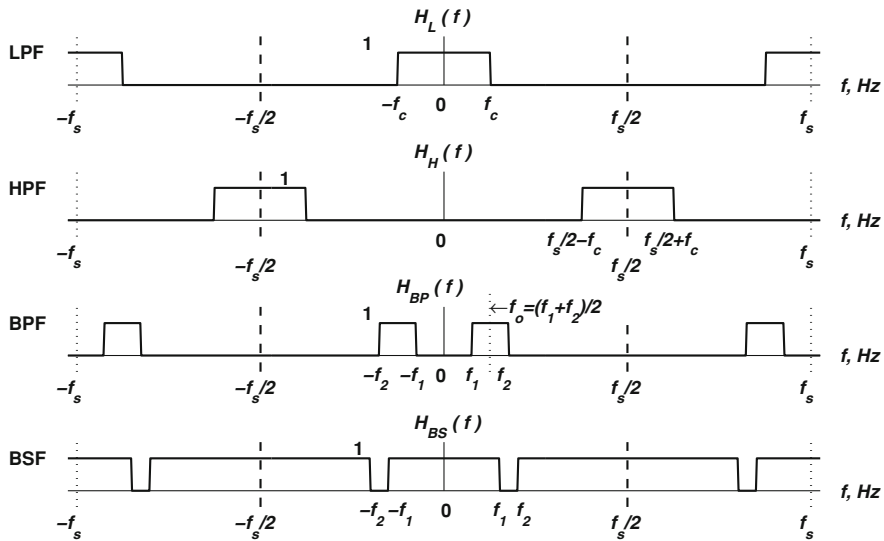


Fig. 2.15 Transfer functions of ideal digital filters

Digital Filter Transformations in the Time-Domain

Similar definitions to that of the LPF can be created for HPF, BPF, and BSF filters. These definitions are provided below, and illustrative plots for the corresponding transfer functions are provided in Fig. (2.15).

1. LP \rightarrow HP : from Fig. (2.15) it is seen that the transfer function of a HPF with cutoff frequency f_c is a spectrally-shifted version of a LPF with the same cutoff frequency, i.e.,

$$\begin{aligned}
 H_H(f) &= H_L(f \pm f_s/2), \\
 \sum_{n=-\infty}^{\infty} h_H(n) e^{-j2\pi n f T_s} &= \sum_{n=-\infty}^{\infty} h_L(n) e^{-j2\pi n (f \pm f_s/2) T_s} \\
 &= \sum_{n=-\infty}^{\infty} h_L(n) (-1)^n e^{-j2\pi n f T_s}
 \end{aligned} \tag{2.20}$$

noting that $e^{-j2\pi n (f_s/2) T_s} = e^{-jn\pi} = (-1)^n$. A simple examination of the above equation indicates that:

$$h_H(n) = (-1)^n h_L(n).$$

i.e., the impulse response of a highpass filter can be obtained from a lowpass one by simply inverting every second sample.

2. LP \rightarrow BP: from Fig. (2.15) it is apparent that:

$$H_{BP}(f) = H_L(f + f_o) + H_L(f - f_o).$$

where $f_o = (f_1 + f_2)/2$ and the cutoff frequency of the LPF $H_L(f)$ is $f_c = (f_2 - f_1)/2$.

$$\therefore h_{BP}(n) = 2 \cos(2\pi n f_o / f_s) h_L(n) [\text{from Tables}].$$

3. BP \rightarrow BS: From Fig. (2.15) one can see that $H_{BS}(f) = 1 - H_{BP}(f)$, hence,

$$h_{BS}(n) = \delta(n) - h_{BP}(n) = \begin{cases} 1 - h_{BP}(0), & n = 0 \\ -h_{BP}(n), & n \neq 0. \end{cases}$$

2.6.2 Linear-Phase Systems

Ideally, it would be convenient if a filter had

1. a magnitude response which did not vary with frequency, and so introduced no amplitude distortion,
2. a zero phase response and hence had no phase distortion.

However, these ideals are not achievable in practical filters. A zero phase response implies that the filter's impulse response is symmetric about $n = 0$ and this in turn implies that the filter is non-causal (see Sect. 1.5.1). Practical filters cannot be non-causal and cannot therefore have zero phase response.

A practical impulse response can be obtained from the ideal (non-causal) impulse response by inserting a delay which positions all (or nearly all) of the impulse response after $n = 0$. This delay introduces a linear phase shift to the transfer function according to Fourier and z-transform *Tables*:

$$h(n - P) \xleftrightarrow{\mathcal{F}} H(z) z^{-P} \Rightarrow H(e^{j\omega T_s}) e^{j\omega P T_s} = H(e^{j2\pi f T_s}) e^{j2\pi f P T_s}.$$

The introduced phase shift is linear with negative slope.

Since a linear phase change in the frequency domain corresponds to a simple time-delay in the time domain, it does not cause any change to the overall shape of the filtered signal. This essentially means that no phase distortion occurs. On the other hand, a non-linear phase response can damage the information content of a signal. Therefore, it is desirable in the practical design of a digital filter to aim for a linear phase. Generally, the transfer function of a linear phase digital system is given by:

$$H(e^{j\omega T_s}) = |H(e^{j\omega T_s})| e^{j\omega\tau} = A(e^{j\omega T_s}) e^{j\omega\tau}$$

where $A(e^{j\omega T_s})$ is real (i.e., is a zero-phase system), and the phase shift is linear and is given by $\phi = \omega\tau$, where τ is a constant).

2.6.3 Classification of Digital Filters

Digital filters can be classified into two major categories according to their impulse response length:

1. *A finite impulse response (FIR) digital filter*: has an impulse response with a finite number of non-zero samples.
2. *An infinite impulse response (IIR) digital filter*: has an impulse response with an infinite number of non-zero samples.

2.6.4 FIR Digital Filters

2.6.4.1 Structure and Implementation of FIR Filters

A causal FIR filter can be specified mathematically by the following difference equation [see Fig. (2.16)]:

$$\begin{aligned} y(n) &= h(n) * x(n), \\ &= \sum_{k=0}^{N-1} h(k)x(n-k) \\ &= h_0x(n) + h_1x(n-1) + \cdots + h_{N-1}x[n-(N-1)] \end{aligned}$$

where h_k is used in place of $h(k)$ for notational simplicity. Taking the z-transform of both sides yields:

$$Y(z) = h_0X(z) + h_1z^{-1}X(z) + \cdots + h_{N-1}z^{-(N-1)}X(z) \text{ [Using Tables]}.$$

Hence, the transfer function is given by:

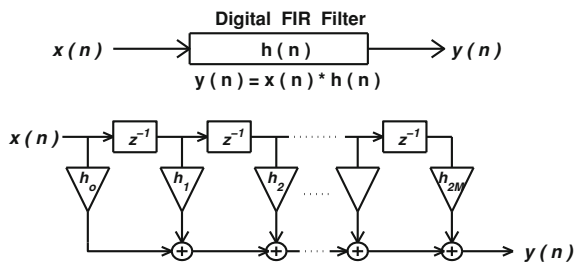
$$H(z) = Y(z)/X(z) = h_0 + h_1z^{-1} + \cdots + h_{N-1}z^{-(N-1)} \quad (2.21)$$

Remembering that a z^{-1} corresponds to a single sample delay, one can implement the causal FIR filter using delay elements and digital multipliers as shown in Fig. (2.16).

2.6.4.2 Software Implementation of FIR Filters

The I/O relation in (2.6) could easily be implemented using a software program on a DSP chip or a digital computer. However, implementation according to the definition in (2.6) would require $O(N_h N_x)$ operations. As suggested in Sect. 2.4.1.5, a more efficient implementation is possible in the frequency domain with the use of FFTs. One uses the relations:

Fig. 2.16 A finite impulse response (FIR) digital filter and its implementation (with $N = 2M + 1$ being odd)



$$Y(k) = X(k) \cdot H(k) \xrightarrow{\text{ifft}} y(n) = Y^{-1}(k).$$

remembering that it is necessary to zero-pad both the input signal and impulse response up to $N_x + N_h - 1$ samples so as to avoid undesirable circular convolution effects.

2.6.4.3 FIR Filtering of Long Data Sequences

In many applications (e.g., speech processing), the signal $x(n)$ can be very long. In this case, one cannot perform FIR filtering using the above technique efficiently since the fft computation time would be very large. Also, saving all of the input data would put a significant strain on the computer memory. Furthermore, real-time processing would be impossible since it would be necessary to wait until the end of the signal to start processing. Instead one can exploit the fact that an FIR filter is a linear system, and obeys the superposition property. Hence, one can partition the data record \mathbf{x} into a sequence of *blocks*, each of relatively small length K , noting that $K \gg M$, M being the filter length. Each of these blocks can be filtered separately and the resulting outputs combined to yield an overall output. The process is called the overlap-add method and is illustrated in Fig. (2.17).

MATLAB the above overlap-add method could be implemented on MATLAB using the instruction `fftfilt(x,h)`, where \mathbf{x} is the signal and \mathbf{h} is the FIR filter impulse response.

2.6.4.4 Pole-Zero Diagram and Stability of FIR Filters

The FIR transfer function in (2.21) can be written as:

$$H(z) = Y(z)/X(z) = \left(\frac{1}{z^{N-1}}\right)[h_0 z^{N-1} + h_1 z^{N-2} + \cdots + h_{N-1}],$$

which shows that an FIR filter with impulse response of length N has $N - 1$ zeros and a multiple pole of order $N - 1$ at the origin ($z = 0$) (Since this multiple pole is inside the unit circle, it poses *no stability problems*).

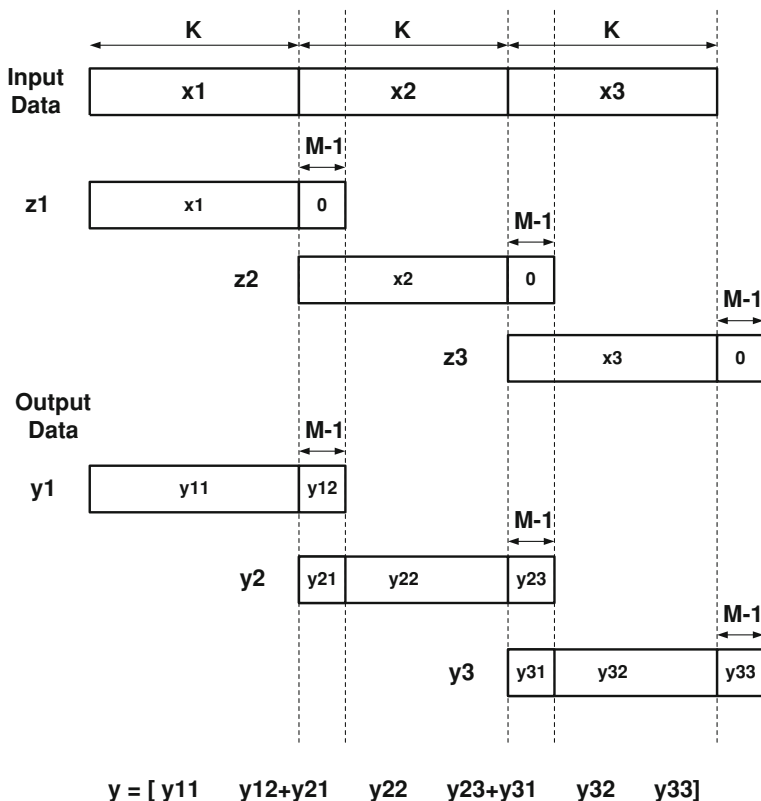


Fig. 2.17 Overlap-add method for processing long data sequence

MATLAB the pole-zero diagram can be plotted on MATLAB using `zplane(A,B)`, where A and B are the numerator and the denominator polynomials coefficients (in descending powers of z). See also the example pole-zero plot for the Digital Differentiator in [Sect. 2.6.6.3](#).

2.6.4.5 Linear-Phase FIR Filters

A sufficient (but not necessary) condition that a FIR filter of impulse response $h(n)$ with length N is a linear-phase system is that $h(n)$ is either:

1. that the impulse response is symmetric around the midpoint of the filter, i.e., $h(n) = h(N - n - 1)$, or:
2. that the impulse response is anti-symmetric around the mid-point of the filter, i.e., $h(n) = -h(N - n - 1)$.

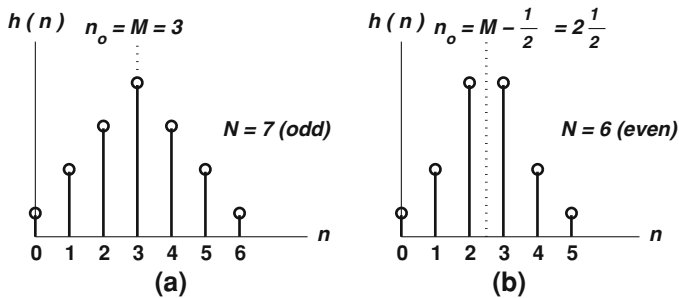


Fig. 2.18 Symmetric impulse response for a FIR filter. **a** odd length. **b** even length

If N is an odd integer, i.e., $N = 2M + 1$, the midpoint of symmetry is $n_o = M$. If N is an even integer, i.e., $N = 2M$, the midpoint of symmetry is $n_o = M - \frac{1}{2}$ [see Fig. (2.18)].

The frequency response of a linear phase FIR filter (symmetric or anti-symmetric) is given by:

$$\begin{aligned}
 H(e^{j\omega T_s}) &= H(z) \big|_{z=\exp(j\omega T_s)} \\
 &= \begin{cases} A_o(\omega)e^{-j\omega T_s M + j\alpha}, & N = 2M + 1(\text{odd}), \alpha = 0 \text{ or } \pi \\ A_e(\omega)e^{-j\omega T_s (M-1/2) + j\beta}, & N = 2M(\text{even}), \beta = 0 \text{ or } \pi \end{cases} \quad (2.22)
 \end{aligned}$$

where $A_o(\omega)$ and $A_e(\omega)$ are arbitrary *real*-valued functions.

Example If $h(n) = \delta(n) + \delta(n-1)$, then:

$$\begin{aligned}
 N &= 2, M = 1, n_o = (N-1)/2 = M/2 = \frac{1}{2}, \text{ and} \\
 H(e^{j\omega T_s}) &= 1 + e^{-j\omega T_s} = 2e^{-j\omega T_s/2} [(e^{j\omega T_s/2} + e^{-j\omega T_s/2})/2] \\
 &= 2e^{-j\omega T_s/2} \cos(\omega T_s/2).
 \end{aligned}$$

Hence, the magnitude response is given by $A_e(e^{j\omega T_s}) = 2|\cos(\omega T_s/2)|$, and the phase response is $\phi = -\omega T_s/2$.

In the z -domain, the system function is $H(z) = 1 + z^{-1} = (z+1)/z$, so that there is a pole at $z = 0$ and a zero at $z = -1$ [see Fig. (2.19)]. It is essentially a LPF. [As an exercise plot the frequency response vs. true frequency f (Hz) or ω (rad/s)!]. The pole-zero diagram can be found on MATLAB using `zplane(A,B)`, where $A = [1 \ 1]$ and $B = [1 \ 0]$.

2.6.4.6 Efficient Hardware Implementation of Linear Phase FIR Filters

Recall that the transfer function of a general FIR filter of length N is:

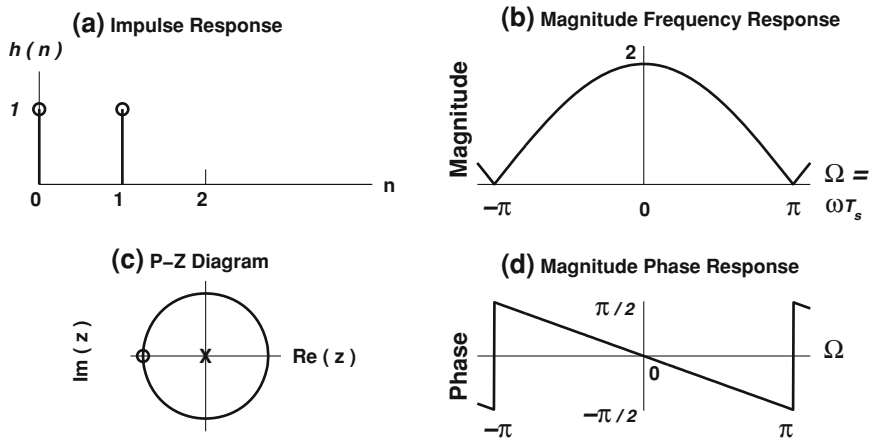


Fig. 2.19 Impulse and frequency response of the filter $h(n) = \delta(n) + \delta(n-1)$

$$H(z) = Y(z)/X(z) = h_0 + h_1 z^{-1} + \dots + h_{N-1} z^{-(N-1)}.$$

A straightforward hardware implementation which follows directly from the expression in (2.21) is shown in Fig. (2.16). If the filter is designed to have linear phase (as many FIR filters are), then its impulse response is typically symmetric or anti-symmetric and one can exploit this symmetry to halve the number of coefficient multipliers. These elements are generally costly and consume a lot of hardware resources. An efficient implementation which takes advantage of the coefficient symmetry is shown in Fig. (2.20) for odd and even length impulse responses.

2.6.5 Design of FIR Digital Filters

2.6.5.1 Time-Domain Design

The frequency response of an ideal digital LPF over the principal frequency domain $(-f_s/2, f_s/2)$ is given by:

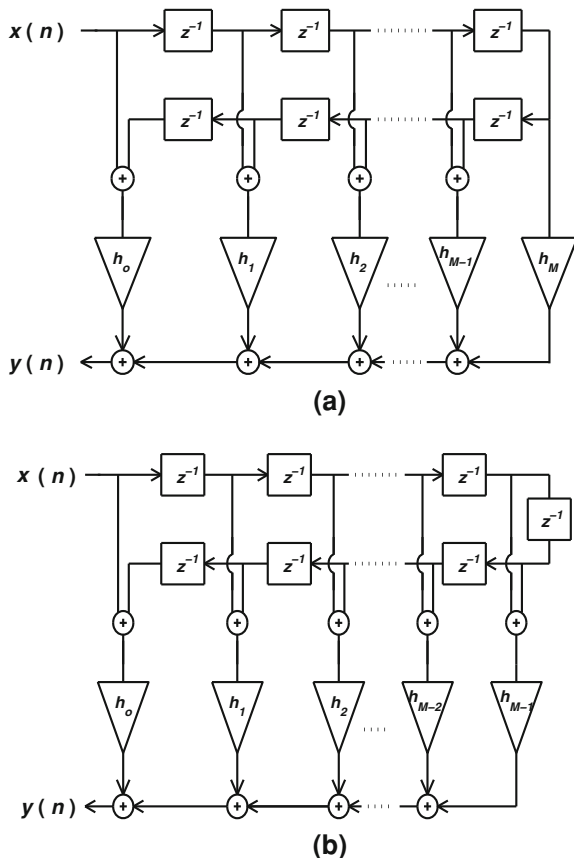
$$H_L(f) = H_L(e^{j2\pi f T_s}) = \begin{cases} 1, & 0 \leq |f| \leq f_c \\ 0, & f_c \leq |f| \leq f_s/2. \end{cases}$$

The impulse response of this filter can be obtained from (see *Tables* to be:

$$h_L(n) = \frac{2f_c}{f_s} \text{sinc}\left(\frac{2f_c}{f_s} n\right).$$

This impulse response and its Fourier transform are depicted in Fig. (2.21).

Fig. 2.20 Efficient implementation of a linear-phase FIR digital filter with symmetric impulse response $h(n)$ **a** with odd length $N = 2M + 1$, **b** with even length $N = 2M$



Since there are non-zero values of $h(n)$ for $n < 0$, the ideal LPF is non-causal, and is hence physically unrealizable. It cannot therefore be used to process real-time signals. In addition, $\sum |h_L(n)| \rightarrow \infty$, and so the filter is also unstable. Since $-\infty < n < \infty$, the ideal LPF is an IIR filter. A practical $2M + 1$ sample FIR approximation to the ideal LPF can be obtained by first shifting $h_L(n)$ right by M samples to get $H_1(n) = h_L(n - M)$. Note that this time-shift causes *only a phase shift* in the frequency domain (see *Tables, Fourier Transform Properties*), and hence $|H_1(f)| = |H_L(f)|$. Second, $h_1(n)$ needs to be truncated (symmetrically around $n = M$) by putting $h_1(n) = 0$ for $n < 0$ or $n > 2M$ (total length is $N = 2M + 1$). This process yields a finite impulse response $h_L(n)$, which is symmetric about $n = M$, as shown in Fig. (2.22) for the scenario where $M = 5$, $N = 11$, $f_c = 1.25$, and $f_s = 5$ Hz.

As would be expected intuitively, larger values of M tend to give rise to better approximations of the ideal impulse response. It is important to note that when the original infinite length time domain impulse response is truncated, the

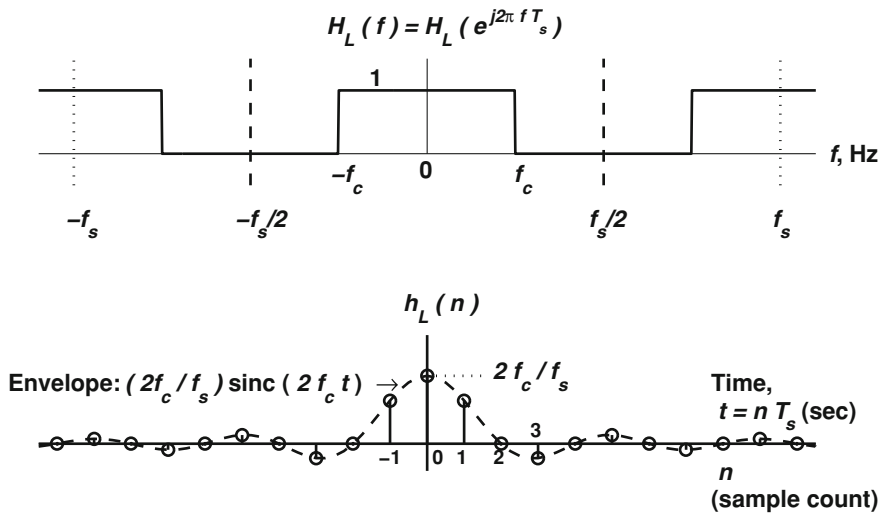


Fig. 2.21 Transfer function of the ideal digital LPF and its impulse response

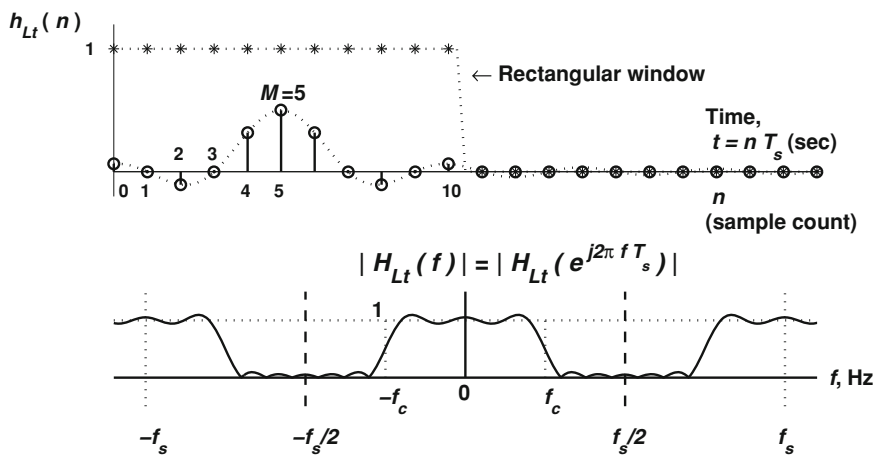


Fig. 2.22 Magnitude and impulse responses of a practical digital LPF, where the impulse response is obtained by truncating the ideal response using a rectangular time window

effect on the frequency domain is a reduction in the sharpness of the transition band. In particular there is an introduction of ripples into the transfer function $H_{Lt}(f)$, with the appearance of these ripples often being referred to as the *Gibbs Phenomenon* [see (Fig. 2.23)]. This phenomenon is discussed further in the next subsection.

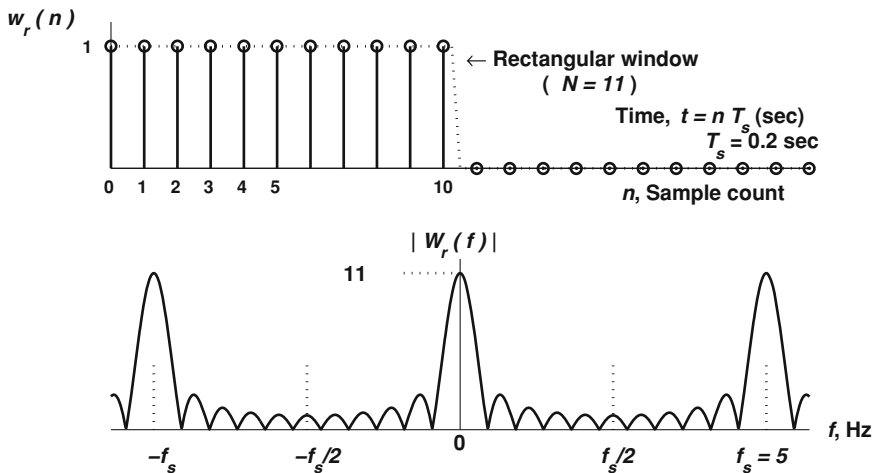


Fig. 2.23 The rectangular time window and its magnitude spectrum (with total length $N = 11$ points, mid-point $M = (N - 1)/2 = 5$ points, and sampling frequency $f_s = 5$ Hz)

Gibbs Phenomenon

The Gibbs phenomenon pertains to the oscillatory behavior in the frequency response corresponding to a truncated digital impulse response sequence. In the previous subsection the approximated impulse response was obtained by truncating a shifted version of the ideal low-pass impulse response ($h_1(n) = h_L(n - M)$) to get the truncated impulse response $h_{Lr}(n)$. This truncation process is equivalent to multiplying $h_1(n)$ in the time domain by a rectangular time window $w_r(n) = \Pi_N(n - M)$. This time window has a Fourier transform of the form

$$W_r(f) = N \frac{\text{sinc}(Nf/f_s)}{\text{sinc}(f/f_s)} e^{-jM2\pi f/f_s},$$

which is a *sinc-like* frequency function with a major lobe and many side lobes as seen in Fig. (2.23). This multiplication in the time domain is equivalent to convolving $H_1(f)$ with $W_r(f)$ in the frequency domain. This convolution with a function having substantial side-lobes is the reason for the oscillations which appear in the magnitude response.

To reduce the effect of the oscillations in $H_{Lr}(f)$, one can multiply the ideal impulse response with a non-rectangular window which has lower amplitude side-lobes. Some suitable windows include the Hamming window, the Hanning window, the Blackman window, the Gaussian window and the Kaiser window [1]. If one does use this kind of *smooth* windowing one typically needs a longer impulse response to have an equivalent quality of approximation.

General formula

A general formula may be used to describe many smooth windows used in practice. This formula is: $w(n) = a + b \cdot \cos(n\pi/M) + c \cdot \cos(2n\pi/M)$, $0 \leq n \leq 2M$ length $2M + 1$

Some common smooth windows conforming to this formula are listed in Table 2.1 and Fig. (2.24).

2.6.5.2 Frequency-Domain Design

So far the design of FIR filters has focused on approximating the desired impulse response (i.e., the time-domain characterization of the filter). An alternative approach is to design the filter by approximating the desired filter transfer function (or frequency representation). This is the approach that is used in the so-called *frequency sampling method* described next.

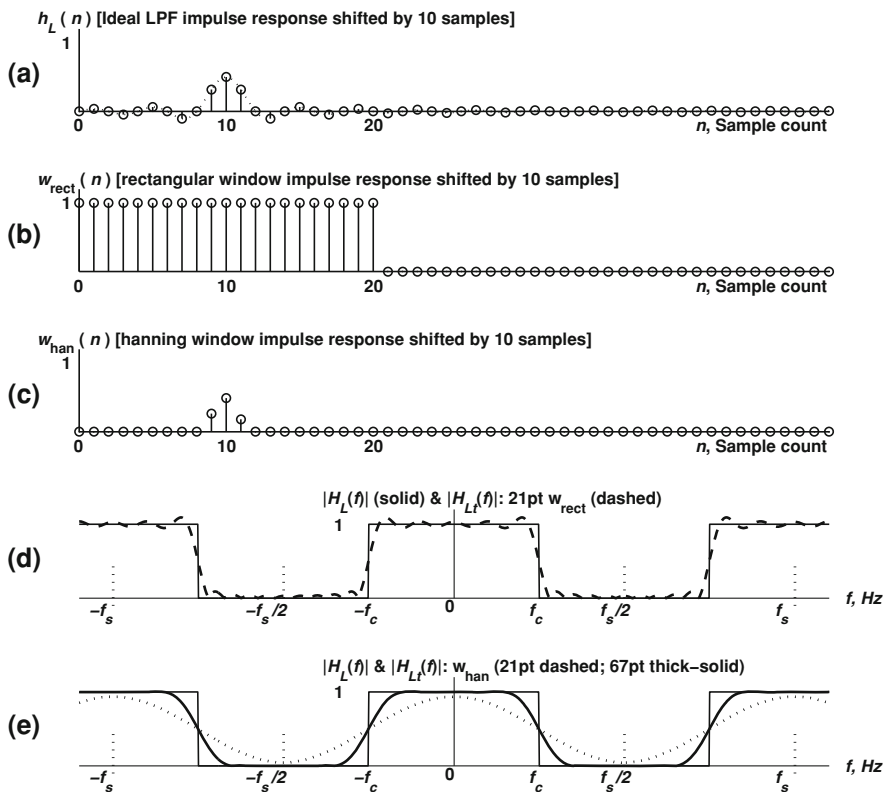


Fig. 2.24 Rectangular and Hanning windowing of the impulse response of an ideal digital LPF

A. Design of FIR Digital Filters by Frequency-Sampling

A intuitively appealing approach to designing a FIR filter is to simply take the desired frequency transfer function and then inverse Fourier transform to find the filter's impulse response. Within a digital computer, however, one cannot specify the Fourier transform, only the *samples* of the desired Fourier transform. In other words one can define the samples of a desired frequency response $H_d(e^{j\omega T_s}) = H_d(e^{j\Omega})$, then use the IDFT to find the impulse response $h(n)$. Often in digital filters it is the transition band which is particularly critical and so it is necessary to specify this portion of the transfer function most accurately. This implies that one needs to have one or more samples within the transition band if one is to achieve good filter designs. It is also important to note that if one seeks to specify too sharp a transition band one will be confronted by the Gibbs Phenomenon—substantial ripple will then manifest in both the pass-band and the stop-band.

Example The goal is to design a 33rd order LPF with cutoff frequency $f_c = f_s/4$ (or, $\Omega_c = 2\pi/4 = \pi/2$). With the frequency sampling method it is necessary to use a 33-point DFT to define the samples of the desired frequency response $H_d(e^{j\omega T_s})$. This desired response is the ideal rectangular response over the principal domain $0 < f < f_s$ which is sampled to yield the set of samples $\{H_d(k) \mid k = 0, 1, \dots, 32\}$. The frequency sampling method is illustrated in Fig. (2.25) for two scenarios, The first scenario is where no samples are specified in the transition band, and the second is where one sample is specified in the transition band. Once the samples have been specified the IDFT can be taken to recover the filter's impulse response. Figure (2.25) shows the DTFT magnitudes of the impulse response so obtained. It is seen in Fig. (2.25) that when no transition band sample is specified the ripple in both the pass and stop bands is significantly greater [Note that $H_d(9)$ and $H_d(24)$ are the transition samples, chosen to be 0.5].

B. Optimal Frequency-Sampling FIR Filter Design

The frequency-sampling method described above is straightforward but it is an *ad hoc* method. It is possible to design filters which are optimal in the sense of minimizing the maximum error between the desired frequency response and the actual frequency response. These types of filters are referred to as *equiripple*, and can be designed using the Parks-McClellan algorithm. The latter utilizes Remez and Chebychev approximation theories to design optimal equiripple linear-phase FIR filters.

MATLAB the Parks-McClellan algorithm is available on MATLAB as:

$$\mathbf{h} = \text{remez}(N - 1, Fd, Ad).$$

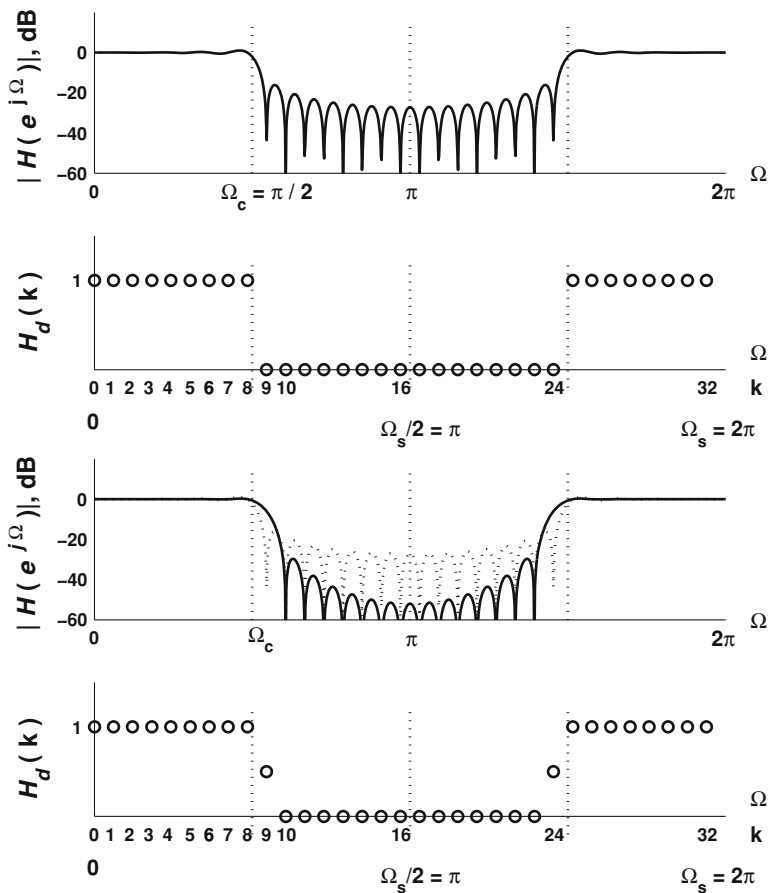


Fig. 2.25 FIR filter design using frequency-sampling method. *Above* without a transition sample. *Below* with two transition samples (*dotted curve* for the first case without transition samples)

This command yields the impulse response of a length- N (i.e., order $N - 1$) linear phase FIR filter with desired magnitude response, given by the vectors F_d and A_d . F_d is a vector of normalized frequency points, ranging from 0 to 1 (corresponding to the range $0 < f < f_s/2$ Hz), and A_d contains the desired magnitudes for points in F_d . Example: To design an optimal FIR filter in MATLAB with the specifications in the previous example, one can use:

$$h = \text{remez}(N - 1, [0 \ 0.5 \ 0.6 \ 1], [1 \ 1 \ 0 \ 0])$$

Figure (2.26) shows the actual frequency response as compared to the ad-hoc frequency-sampling approach described in Sect. 2.6.5.2-A.

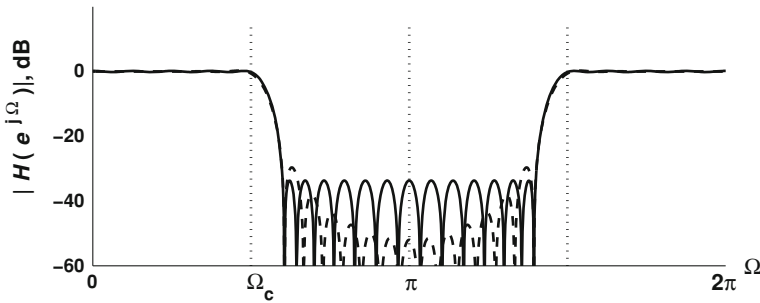


Fig. 2.26 Frequency response of a LPF designed by remez (dotted curve for ad-hoc frequency-sampling design)

2.6.6 Applications of FIR Digital Filters

2.6.6.1 Communication Channel Equalization

Telephone and wireless channels often behave like band-limited filters with frequency response $H(f)$. This filtering is in general an undesirable frequency dependent distortion which needs to be eliminated - it can be removed via the use of an *equalizer* as explained below.

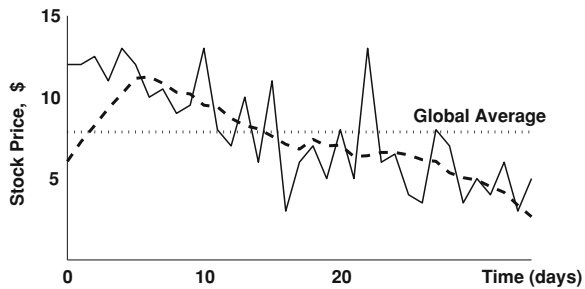
The transfer function of the channel can be estimated by sending a training signal [e.g., a unit pulse function $\delta(n)$] along the channel and then measuring the impulse response $h(n)$ and the corresponding transfer function $H(f) = \mathcal{F}\{h(n)\}$. An equalizer (or inverse filter) with a frequency response $H_e(f) = 1/H(f)$ is then applied to counteract the channel distortion. Normally one chooses an FIR filter (also called a transversal filter), and the frequency sampling approach is often used to design $H_e(f)$.

2.6.6.2 The Moving Average Filter

Sometimes it is necessary to smooth data before making a decision or interpretation about that data. Averaging is a commonly used smoothing technique and this averaging can be implemented with simple FIR filters. For example, the stock market prices fluctuate from day to day, and even sometimes hour to hour. To observe trends some form of smoothing or averaging is necessary. Typically, one takes an average of the stock price over several days before deciding the actual trend of prices. *Note that the global average is misleading in these cases and cannot be used—rather, local or moving average should be used.* (See Fig. (2.27)).

Since data $x(n)$ is assumed to be *continuously flowing in* for the stock market scenario, local averaging is done at each instant n . If, for example, averaging is performed over three consecutive samples, the output becomes:

Fig. 2.27 Moving average of a stock price over 35 days using a 10-tap FIR filter. Note that the first $M - 1 = 9$ samples are inaccurate due to insufficient information at start



$$y(n) = \frac{1}{3}[x(n) + x(n-1) + x(n-2)] \quad (2.23)$$

For example, at $n = 2, y(2) = \frac{1}{3}[x(2) + x(1) + x(0)]$, at $n = 3, y(3) = \frac{1}{3}[x(3) + x(2) + x(1)]$. Comparing the above Eq. (2.23) with the general FIR equation:

$$\begin{aligned} y(n) &= h(n) * x(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \\ &= h_0(n) + h_1x(n-1) + \cdots + h_{N-1}x[n-(N-1)] \end{aligned}$$

it is seen that (2.23) represents an FIR filter with $N = 3$ taps: $h(0) = h(1) = h(2) = \frac{1}{3}$. Although these filter coefficients are all equal in this case they can be different in general. For example, in the stock market studies one may give more weight (importance) to the current sample [i.e., $x(n)$] than to others, e.g.,

$$\begin{aligned} h(0) &= 0.5, & 50\% \text{ weight to the current price (today)} \\ h(1) &= 0.3, & 30\% \text{ weight to the previous price (yesterday)} \\ h(2) &= 0.2, & 20\% \text{ weight to the first price (2 days ago)} \end{aligned}$$

Figure (2.27) shows an example of a changing price over 35 days along with the overall average and the moving average with $h(n) = [.1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1]$ ($M = 10$ taps with equal weight).

2.6.6.3 The Digital Differentiator

Time Domain Approach

The derivative of a signal $x(t)$, $dx(t)/dt$, can be approximated in the digital domain (when the sampling period T_s is small and $x(t)$ is slowly varying) by the relation:

$$y(n) = [x(n) - x(n-1)]/T_s \quad (2.24)$$

as shown in Fig. (2.28). Comparing with the general form of an FIR filter:

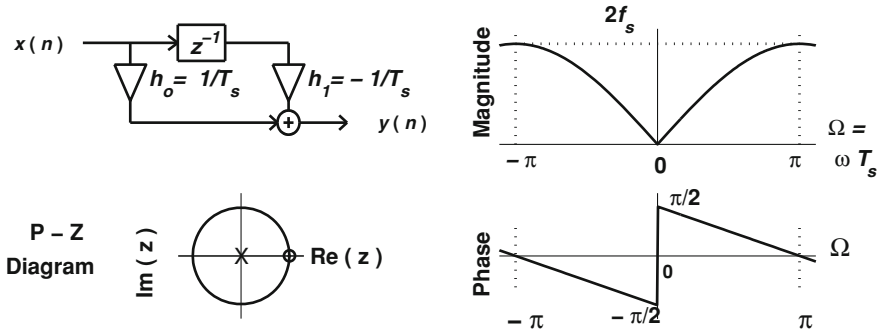


Fig. 2.28 Digital differentiator-I with frequency response and p-z diagram

$$\begin{aligned}
 y(n) &= h(n) * x(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \\
 &= h_0(n) + h_1x(n-1) + \cdots + h_{N-1}x[n-(N-1)]
 \end{aligned}$$

It can be seen that (2.32) represents a FIR filter with two non-zero samples impulse response $\{h(n) = (1/T_s)\delta(n) - (1/T_s)\delta(n-1)\}$, i.e., $h(0) = -h(1) = 1/T_s$. From this one can find the system transfer function by taking the z-transform of both sides of (2.32):

$$\begin{aligned}
 Y(z) &= \frac{1}{T_s} [X(z) - X(z)z^{-1}] = [(1 - z^{-1})/T_s]X(z) \\
 \Rightarrow H(z) &= \frac{Y(z)}{X(z)} = \frac{(1 - z^{-1})}{T_s}
 \end{aligned}$$

Now consider the magnitude and phase response:

$$\begin{aligned}
 H(e^{j\Omega}) &= H(z) \big|_{z=\exp(j\Omega)} = \frac{1}{T_s} [1 - e^{-j\Omega}] \\
 &= \frac{1}{T_s} e^{-\frac{j}{2}\Omega} [e^{+\frac{j}{2}\Omega} - e^{-\frac{j}{2}\Omega}] \\
 &= \frac{1}{T_s} e^{-\frac{j}{2}\Omega} [2j \sin(\Omega/2)] \\
 &= \left(\frac{2j}{T_s}\right) e^{-\frac{j}{2}\Omega} \sin\left(\frac{1}{2}\Omega\right)
 \end{aligned}$$

Hence, the magnitude response is given by:

$$|H(e^{j\Omega})| = \left(\frac{2}{T_s}\right) \left|\sin\left(\frac{1}{2}\Omega\right)\right| = 2f_s \left|\sin\left(\frac{1}{2}\omega T_s\right)\right| = 2f_s |\sin(\pi f/f_s)|,$$

and the phase response is:

$$\phi = \left(-\frac{\Omega}{2} + \frac{\pi}{2} \right) + c,$$

where $c = \pi$ is to compensate for the change of sign in $\sin(\frac{1}{2}\Omega)$.

The magnitude and phase responses are plotted in Fig. 2.28. It is seen that the digital differentiator is a high-pass filter.

Frequency Domain Approach

From Fourier transform *Tables* one can find the transfer function of the continuous-time differentiator as:

$$H_a(\omega) = j\omega \quad (2.25)$$

which can be transformed into the digital domain as follows:

$$H(e^{j\Omega}) = j\Omega/T_s; \quad -\pi \leq \Omega < \pi. \quad (2.26)$$

Figure (2.29) shows the magnitude and phase responses of the above transfer functions (compare with Fig. (2.28)). Note that Fig. (2.29a) represents a theoretical magnitude response, as practical differentiators are band-limited within a cutoff frequency ω_c .

The impulse response of the above filter can be found to be:

$$\begin{aligned} h(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{j\Omega}{T_s} e^{jn\Omega} d\Omega \\ &= \frac{1}{T_s} \frac{\cos(n\pi)}{n} \frac{\sin(n\pi)}{\pi n^2}; \quad -\infty < n < \infty \end{aligned} \quad (2.27)$$

For a practical design, the above impulse response should be shifted by $2M + 1$ samples and then truncated. The shifting operation is described by:

$$h(n) = \frac{1}{T_s} \frac{\cos[(n-M)\pi]}{(n-M/2)} \frac{\sin[(n-M)\pi]}{\pi(n-M)^2}; \quad -\infty < n < \infty, \quad (2.28)$$

while the truncation window $w(n)$ is specified by:

$$h_w(n) = w(n) \cdot h(n); \quad 0 < n < 2M \quad (2.29)$$

If a linear phase is required, then the truncation window should be symmetric so that one can get $h(n) = -h(2M - n)$. The filter in (2.29) can be implemented using the approach in Sect. 2.6.4.6.

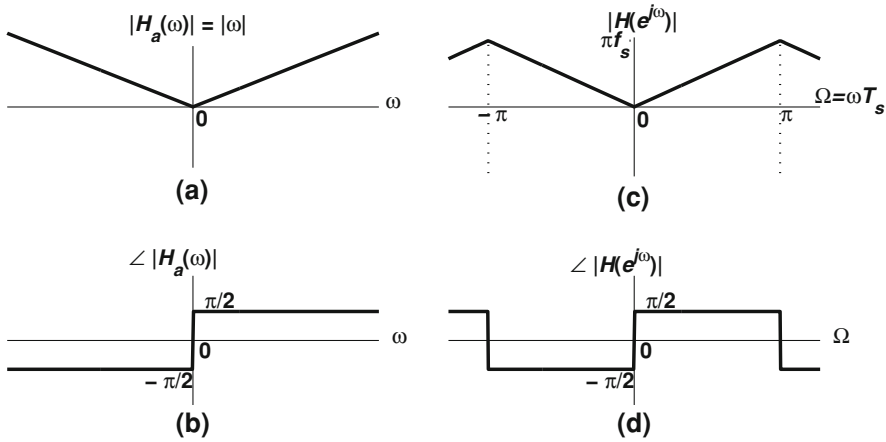


Fig. 2.29 Frequency response for an analog and digital differentiator-II

2.6.6.4 The Digital Matched Filter

Recall from Sect. 1.4.2 that in the analog domain the impulse response $h(t)$ of the optimum filter for receiving a transmitted symbol $b(t)$ ($0 \leq t \leq T$) is given by the T -shifted (delayed) mirror image of the symbol, i.e.,

$$h(t) = b(T - t)\Pi_T(t - T/2).$$

For orthogonal binary transmission, two matched filters are needed, one matched to the “0” and the other matched to the “1”. Figure (2.30) shows a digital matched filter for orthogonal binary signalling with the sampling frequency at the receiver equal to *five times the bit rate*. If bipolar signalling is used (i.e., $+V_{cc}$ represents “1” and $-V_{cc}$ represents “0”), then one sample per bit is possible, but in practical signalling schemes there are normally at least 4 samples per bit. Increasing the sample rate increases the accuracy of detection, especially in noise, and it does not affect the transmission bandwidth (which depends on the data rate $1/T$ only). If the sample rate is N samples per bit, then an N -stage shift register is needed to store the samples before each symbol is detected for each filter (see Fig. (2.30)).

In digital matched filters it is necessary to have good synchronization between the transmitter and the receiver to decide the start time ($t = 0$). The coefficients of the i th matched filter $\{h_i(n) \mid n = 0, 1, \dots, N - 1\}$, which is matched to the i th symbol (bit), are given by:

$$h_i(n) = b_i(N - 1 - n), \quad n = 0, 1, \dots, N - 1$$

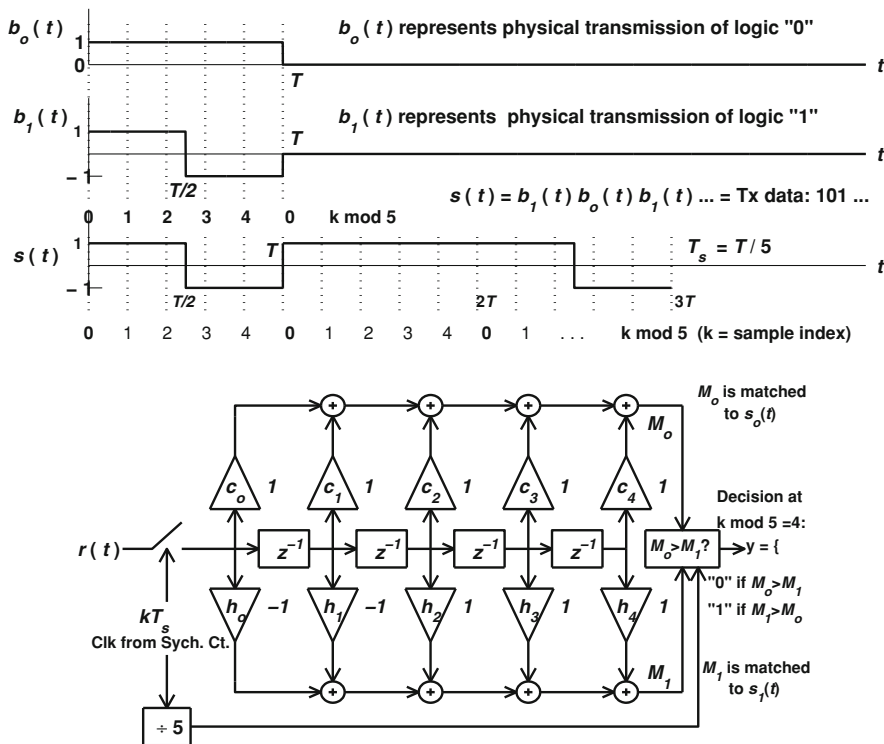


Fig. 2.30 Digital matched filter for orthogonal binary signaling with $T_s = T/5$. Above waveforms for transmitting logic "0" and logic "1". The transmitted signal $s(t)$ represents the data string 1 0 1. Below digital matched filter receiver

The output of the i th matched filter at the k th sampling instant is given by the convolution:

$$M_i(n) = \sum_{n=0}^{N-1} r(k-n)h_i(n)$$

where $r(t)$ is the received signal. Decisions (comparisons) are made only when $kN = N - 1$ [every 5 samples for the scenario represented in Fig. (2.30)]. Note that in Fig. (2.30) the simpler notation $h_n = h_1(n)$ and $c_n = h_0(n)$ is used—under noise-free condition and full synchronization with the transmitter at $k = 4$ (after 5 samples) the shift register has the following contents: $a = 1$, $b = 1$, $c = 1$, $d = -1$, $e = -1$. Hence, $M_0 = 1$, $M_1 = 5 > M_0$, and the decision is that "1" was transmitted. As an exercise try to find the decision of the circuit after 10 and 15 samples.

2.6.7 IIR Digital Filters

2.6.7.1 Structure and Implementation of IIR Digital Filters

An IIR digital filter is a system whose impulse response $h(n)$ has an infinite number of non-zero samples. To implement IIR filters in practice, recursion is often used. That is, to create the output, not only are previous values of the input used, but also previous values of the output. This kind of recursive implementation is essentially a feedback system. The general formula for the I/O relations of a recursive IIR filter are:

$$y(n) = b_0x(n) + b_1x(n-1) + \cdots + b_Mx(n-M) - a_1y(n-1) - \cdots - a_Ny(n-N). \quad (2.30)$$

The above equation is often referred to as a *difference equation*. Taking the z-transform of both sides of the equation yields:

$$\begin{aligned} Y(z) &= b_0X(z) + b_1z^{-1}X(z) + \cdots + b_Mz^{-M}X(z) \\ &\quad - a_1z^{-1}Y(z) - \cdots - a_Nz^{-N}Y(z). \\ \therefore H(z) &= \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + \cdots + b_Mz^{-M}}{1 + a_1z^{-1} - \cdots - a_Nz^{-N}} = \frac{\sum_{i=0}^M b_i z^{-i}}{1 + \sum_{k=1}^N a_k z^{-k}}. \end{aligned} \quad (2.31)$$

Hence, unlike FIR filters, IIR filters can have poles at locations other than $z = 0$. In fact, the above equations can also represent a FIR filter if one puts $a_1 = a_2 = \dots = 0$. Normally, the number of poles N is larger than the number of zeros M , and the order of the filter is decided by the number of its poles. For the IIR filter to be stable, *all poles should be inside the unit circle*.

2.6.7.2 IIR versus FIR Filters

IIR filters usually have lower orders than FIR filters with similar performance with respect to sharpness of cutoff, passband ripple, etc. Because of the lower orders IIR filters tend to require fewer delay elements and digital multipliers for hard-ware implementation, and they require fewer computations in software implementations.

One of the key disadvantages of IIR filters is that because of their inherent use of feedback, they can have stability problems. Quantization effects are also much more serious in IIR filters, again due to their use of feedback. Additionally, it is generally not possible to have a linear phase transfer function with IIR filters.

2.6.7.3 Direct Form Implementation of IIR Digital Filters

From the general equation for an IIR filter:

$$Y(n) = b_0X(n) + b_1z^{-1}X(n) + \cdots + b_Mz^{-M}X(n) \\ - a_1z^{-1}Y(n) - \cdots - a_Nz^{-N}Y(n),$$

one can readily construct a hardware realization, as shown in Fig. (2.31). The structure shown is called the direct form-I, realization. This form requires $M + N + 1$ multipliers, $M + N$ adders, and $M + N$ memory locations.

Now the transfer function for an IIR filter can also be written as:

$$H(z) = \frac{\sum_{i=0}^M b_i z^{-i}}{1 + \sum_{k=1}^N a_k z^{-k}} = \left(\sum_{i=0}^M b_i z^{-i} \right) \left(\frac{1}{1 + \sum_{k=1}^N a_k z^{-k}} \right) . \\ = H_1(z)H_2(z) = H_2(z)H_1(z) .$$

The above expression suggests an alternative realization for the filter, and the new realization is called the direct form-II implementation (Fig. 2.32). Importantly the direct form-II requires only $\max(M, N)$ memory locations (or delay elements) in contrast to the direct form-I, requires $M + N + 1$. The new realization needs the same number of adders and multipliers as the direct form-I. The coefficients $\{a_i\}$ and $\{b_i\}$ are usually chosen to be *real to avoid complex processing*. It can be shown that with real coefficients the poles and zeros are either real or in complex-conjugate pairs.

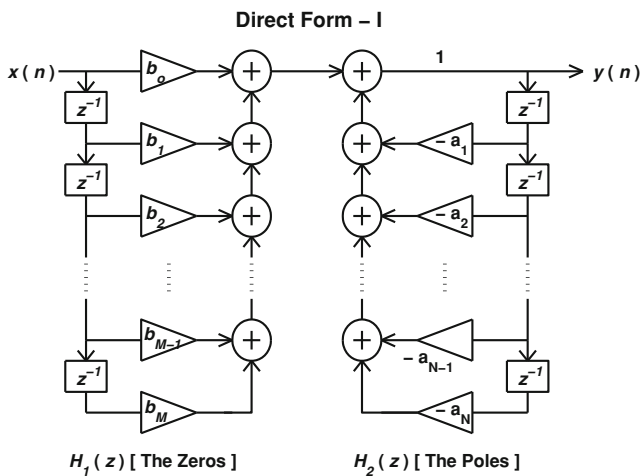
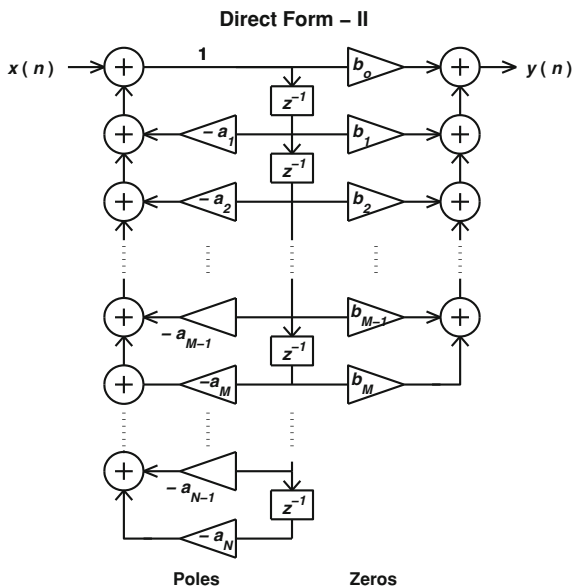


Fig. 2.31 Direct form-I implementation of IIR digital filters

Fig. 2.32 Direct form-II implementation of IIR digital filters. *Note* to find the poles and zeros of any transfer function $H(z)$, it is normal to write it as a function of z , but for implementation it is normal to write it as a function of z^{-1} . The use of z^{-1} terms in the implementation is due to the correspondence of the z^{-1} term to a delay element



2.6.7.4 Practical Implementation of IIR Digital Filters

Since the feedback structure of IIR filters exaggerate quantization errors, these filters are often built up in practice as cascaded first- and second-order units rather than in direct forms - this can be shown to reduce the vulnerability to quantization errors. Ideally one would use only first order sections, but this would necessitate using complex arithmetic. To avoid this latter possibility it is necessary to use second order sections wherever there are complex conjugate poles or zeros.

Example Implement the IIR digital filter whose transfer function is given by:

$$H(z) = \frac{2(z-1)(z+2)(z+3)}{z(z+0.5)(z^2+0.3z+0.1)}.$$

It is seen that there are two complex conjugate poles since $z^2 + 0.3z + 0.1$ has two complex conjugate roots. Hence, it is best to use a cascaded form of implementation and write the transfer function as follows:

$$\begin{aligned} H(z) &= \frac{2(z-1)(z+2)(z+3)}{z(z+0.5)(z^2+0.3z+0.1)} \\ &= 2z^{-1} \underbrace{\left(\frac{1-z^{-1}}{1+0.5z^{-1}} \right)}_{\text{1st-order unit}} \underbrace{\left(\frac{1+5z^{-1}+6z^{-2}}{1+0.3z^{-1}+0.1z^{-2}} \right)}_{\text{2nd-order unit}}. \end{aligned}$$

The implementation diagram is shown in Fig. (2.33).

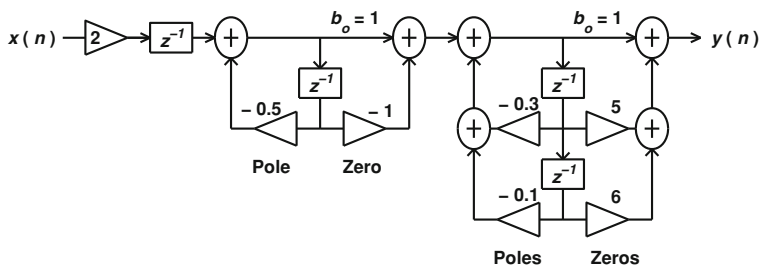


Fig. 2.33 Implementation of $2(z-1)(z+2)(z+3)/[z(z+0.5)(z^2+0.3z+0.1)]$

2.6.8 Design of IIR Digital Filters

Often IIR digital filters are designed by converting analog prototype filters (such as Butterworth, Chebychev, and Elliptic filters) into digital filters via suitable transformation methods. The basic prototype filter type is an analog low-pass filter, and filter transformations are necessary to obtain LP, HPF, BPF, or BSF digital filters.

Typically the required digital filter specifications are initially transformed into their analog counterparts to help in designing the proper analog prototype. For this to be effective the transformation should:

1. preserve stability and
2. map the $j\omega$ axis into the circumference of the unit circle $e^{j\Omega}$.

There are two approaches for transforming digital IIR filters into analog prototypes:

1. Time-domain matching using the impulse response,
2. Frequency-domain matching using the frequency response.

2.6.8.1 Time-Domain Design: Impulse Response Matching

In the impulse response matching method (a.k.a. the impulse invariance method), the impulse response of the digital IIR filter is simply taken to be a sampled version of the impulse response of the analog prototype filter. That is,

$$h(n) = h_a(nT_s), \quad n = 0, 1, 2, \dots$$

If the transfer function of the analog prototype filter $H_a(f)$ is bandlimited to $(-B, B)$ and $f_s/2 \geq B$, then there would be no aliasing. This in turn would imply that the digital transfer function $H(e^{j\Omega})$ would be a scaled and repeated copy of $H_a(f)$. With this understanding it becomes apparent that the impulse response matching method is not suitable to design a digital HPF, due to aliasing problems.

The analog filter s-domain transfer function is $H_a(s) = LT\{h_a(t)\}$. Now the poles of $H_a(s)$ are transformed to the poles of $H(z)$ through the relation $z = e^{sT_s}$. The method can be summarized as follows (see, for example, A. V. Oppenheim and R. Schaffer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989):

1. Expand the analog transfer function by partial fractions as $\sum_{m=1}^M \frac{c_m}{s-p_m}$. Note that p_m 's can be non-distinct.
2. The required transfer function is

$$H(z) = T_s \sum_{m=1}^M \frac{c_m}{1 - e^{p_m T_s} z^{-1}} = T_s \sum_{m=1}^M c_m \frac{z}{z - z_m}$$

where $z_m = e^{p_m T_s}$ are the poles in the z-domain.

Note 1 If there is no aliasing, $H(e^{j\Omega}) = H_a(\omega)$ when $-\pi \leq \Omega \leq \pi$.

Note 2 Since the relation $\Omega = \omega T_s$ is applicable, the $j\omega$ axis is transformed into the circumference of the unit circle $e^{j\omega T_s}$.

Note 3 Since the poles are transformed according to $z = e^{sT_s}$, stability is preserved.

Example Using the impulse invariance method, design a digital Butterworth LPF with the following specifications:

1. $T_s = 0.01$ sec (hence, $f_s = 100$ Hz),
2. $f_c = 10$ Hz (therefore $\omega_c = 20 \pi$ rad/sec),
3. $G_m = 1$, gain ≤ 0.1 (i.e., -20 dB) for $20 \leq f \leq f_s/2 = 50$ Hz).

Solution:

It is necessary to first find an analog Butterworth LPF with the above specifications. The gain should be less than -20 dB for a normalized frequency of $f_c \geq 20/10 = 2$ (normalized w.r.t f_c). From the graph in *Tables*—Stopband gain for a B-LPF, the filter order is found to be $n = 4$. From *Tables*—Denominator Polynomial Coefficients for Normalized LPF's, the transfer function of this normalized filter is seen to be:

$$H_N(s) = \frac{a_o}{1 + 2.6131s_N + 3.4142s_N^2 + 2.6131s_N^3 + s_N^4}$$

Now $H_N(s)|_{s=0} = a_o = G_{dc} = G_m = 1$, and $s_N = s/\omega_c$. Substituting these values for a_o and s_N into the above equation, yields:

$$H_a(s) = \frac{1.55e7}{1.55e7 + 6.48e5s + 1.34e4s^2 + 164.1s^3 + s^4}$$

$H_a(s)$ can then be expanded using partial fractions. This can be easily done in MATLAB using $B = [1 \ 164.1 \ 1.34e4 \ 6.48e5 \ 1.55e7]$, $A = [1.55e7]$, $[R, P, K] = \text{residue}(A, B)$ where R gives the coefficients c_m , P gives the poles

p_m , and \mathbb{K} is empty if $\text{degree}(\mathbb{A}) < \text{degree}(\mathbb{B})$. For the Butterworth filter above, the poles and coefficients are found to be the following complex conjugate pairs:

$$\begin{aligned}c_1 &= -27.8 + j11.6, \\p_1 &= -23.6 + j58.3, \\c_2 &= c_1^*, \\p_2 &= p_1^*, \\c_3 &= 27.8 - j73.7, \\p_3 &= -58.4 + j22.3, \\c_4 &= c_3^*, \\p_4 &= p_3^*.\end{aligned}$$

The z-domain poles are:

$$\begin{aligned}z_1 &= \exp(p_1 T_s) = 0.65 + j0.43, \\z_2 &= z_1^* [\text{since}(e^x)^* = e^{(x^*)}], \\z_3 &= 0.54 + j0.12,\end{aligned}$$

and $z_4 = z_3^*$. Therefore,

$$\begin{aligned}H(z) &= T_s \left[\frac{c_1 z}{z - z_1} + \frac{c_1^* z}{z - z_1^*} + \frac{c_3 z}{z - z_3} + \frac{c_3^* z}{z - z_3^*} \right] \\&= T_s \left[\frac{r_1 z^2 - r_2 z}{z^2 - r_3 z + r_4} + \frac{q_1 z^2 - q_2 z}{z^2 - q_3 z + q_4} \right]\end{aligned}$$

where $r_1 = -0.55$, $r_2 = -0.26$, $r_3 = 1.3$, $r_4 = 0.6$, $q_1 = 0.55$, $q_2 = 0.47$, $q_3 = 1.08$, and $q_4 = 0.3$. The above *grouping of terms is to have real coefficients* for easier hardware implementation.

MATLAB the above problem can also be solved using the MATLAB command $[Az \ Bz] = \text{impinvar}(A, B, fs)$.

To check that the impulse responses are similar use:

```
t = 0:Ts:1
sys1 = tf(A,B)
h1 = impulse(sys1,t)
sys2 = tf(Az/Ts,Bz,Ts)
h2 = impulse(sys2,t)
plot(t,h1,t,h2':'')
```

Note In filter design the following MATLAB commands may prove useful:
 $B = \text{roots}(A)$: Finds the roots of a polynomial whose coefficients are in A [e.g., $B = \text{roots}([1 \ 2 \ 3])$ finds the roots of (descending order)].

$A = \text{poly}(B)$: Converts the roots (in B) to a polynomial.

$C = \text{conv}(A, B)$: Multiply two polynomials A and B (Also, find the convolution sum).

`X=fzero('fun',xo)`: Finds a zero of the function “fun” (which is either a library function or defined in another file) near `xo` [e.g., `fzero('x/3-sin(x)',pi/2)` gives 2.2789].

`zplane(A,B)`: Plots the pole-zero diagram of a transfer function $H(z) = A(z)/B(z)$.

`h=freqs(A,B,w)`: computes the complex frequency response of the analog filter $H(s) = A(s)/B(s)$ at the angular frequencies in the vector `w`.

`h=freqz(A,B,f,fs)`: computes the complex frequency response of the digital filter $H(z) = A(z)/B(z)$ at the frequencies in the vector `f`, with sampling frequency `fs`.

2.6.8.2 Frequency-Domain Design: Frequency Response Matching

The impulse invariance method enforced a time-domain correspondence between the analog and digital impulse responses. The frequency response matching approach enforces a frequency-domain correspondence between the analog and digital transfer functions. Assume that one starts with an analog filter transfer function $H_a(s)$, which needs to be transformed into a digital transfer function $H(z)$. Unlike the time domain matching approach, it is not possible to achieve an *exact* correspondence between $H_a(s)$ and $H(z)$ -this is because the $H(z)$ displays repetition in frequency whereas $H_a(s)$ does not.

As a first attempt at achieving a frequency domain correspondence between the digital and analog filters, one can use the natural mapping:

$$H(e^{sT_s}) = H_a(s) \quad \text{for } -\pi \leq \omega T_s \leq \pi.$$

Hence,

$$H(e^{j\omega T_s}) = H_a(j\omega) \quad \text{for } -\pi \leq \omega T_s \leq \pi.$$

The relationship $z = e^{sT_s}$ is enforced in this approach. Thus,

$$s = \frac{\ln(z)}{T_s}.$$

It is apparent that this transformation is non-linear and it is therefore impossible in general to obtain $H(z)$ as a rational function of z . However, an approximation to this relation is possible. If z is near 1, the Taylor expansion of $\ln(z)$ is $(z - 1) - (z - 1)^2/2 + O\{(z - 1)^3\}$, while the expansion of $(z - 1)/(z + 1)$ is $[(z - 1) - (z - 1)^2 + O\{(z - 1)^3\}]/2$. Hence, ignoring the smaller terms, the following approximation can be used: $\ln \approx 2(z - 1)/(z + 1)$. Then s can be approximated by:

$$s \approx \frac{2}{T_s} \frac{z - 1}{z + 1}.$$

This relation is called the *bilinear transform*. It should be noted that the approximation in used above is true only when z is *near* 1, which means that the low frequency region maps quite reliably from the analog domain to the digital domain. At higher frequencies, however, there is considerable error in the approximation. It will be seen subsequently that the bilinear transform actually introduces a warping of the analog frequency axis, and the warping is most severe at high frequencies. This warping causes the analog frequency range $0 \rightarrow \infty$ to be mapped into the digital frequency range $0 \rightarrow \pi$.

To gain further insights into the nature of the bilinear transform one can put $z = re^{j\Omega}$ [recall that the frequency response of any digital system $H(z)$ can be found by substituting $z = re^{j\Omega}$]. Substituting the polar form for $z (= re^{j\Omega})$ in the bilinear transform $s \approx (2/T_s)(z - 1)/(z + 1)$ gives:

$$\begin{aligned} s &\approx \frac{2 re^{j\Omega} - 1}{T_s re^{j\Omega} + 1} \\ &= \frac{2 [r \cos(\Omega) - 1] + jr \sin(\Omega)}{T_s [r \cos(\Omega) + 1] + jr \sin(\Omega)} \\ &= \frac{2}{T_s} \left[\frac{r^2 - 1}{1 + r^2 + 2r \cos(\Omega)} + j \frac{2r \sin(\Omega)}{1 + r^2 + 2r \cos(\Omega)} \right]. \end{aligned}$$

Since $s = \sigma + j\omega$:

$$\sigma = \frac{2}{T_s} \frac{r^2 - 1}{1 + r^2 + 2r \cos(\Omega)} \quad \text{and} \quad \omega = \frac{2}{T_s} \frac{2r \sin(\Omega)}{1 + r^2 + 2r \cos(\Omega)}$$

Now for $\sigma = 0$, $|z| = r = 1$ and so $z = e^{j\Omega}$. Hence, the $s = j\omega$ axis in the analog domain is transformed into the $z = e^{j\Omega}$ unit circle, where the new relation between Ω and ω is given by:

$$\begin{aligned} \omega &= \frac{2}{T_s} \frac{\sin(\Omega)}{1 + \cos(\Omega)} = \frac{2}{T_s} \tan\left(\frac{\Omega}{2}\right) \\ \therefore \Omega &= 2 \tan(\omega T_s / 2). \end{aligned} \tag{2.32}$$

The above equation is non-linear, as illustrated graphically in Fig. (2.34). The infinite analog frequency domain ($\omega = 0 \rightarrow \infty$) is mapped onto the principle digital frequency range $\Omega = 0 \rightarrow \pi$.

If $\sigma < 0$, then $r < 1$ (hence the LHS of the s -plane is mapped inside the unit circle), and for $\sigma > 0$, $r > 1$.

It can be shown that filter design using the bilinear transform is independent of T_s [2]; hence, one can put $T_s = 2$ (for convenience) and use the following transform instead:

$$s = \frac{z - 1}{z + 1} \quad \left[\text{with} \quad \Omega = 2 \tan^{-1}(\omega) = 2 \tan^{-1}(2\pi f) \right]. \tag{2.33}$$

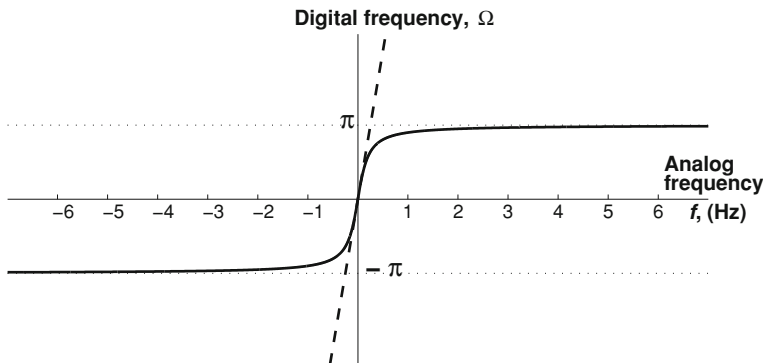


Fig. 2.34 Frequency domain relationship between analog frequency and digital frequency under the natural transformation $z = e^{sT_s}$, where $\Omega = \omega T_s$ (dashed curve, $T_s = 2$) and the bilinear approximation $s = (2/T_s)(z - 1)/(z + 1)$, where $\Omega = 2 \tan^{-1}(\omega)$ (solid curve)

The bilinear transformation can be applied to transform all analog Filters, while the impulse invariance method cannot be applied to HPFs. This is so because there is no problem with aliasing, as this transform maps the whole $j\omega$ axis to the unit circle. That is, it is a *1-to-1 mapping*. The impulse invariance transform $z = e^{sT_s}$, on the other hand maps every sector of length 2π on the $j\omega$ axis to the unit circle - it is a *multi-to-1*. The price paid for the resilience to aliasing problems is the deformation or warping of the original analog frequency response. It should be noted that this warping is for the frequency response of the transformed analog filter only; it does not imply that the input signal spectrum is also warped.

Example Design a low-pass digital IIR filter with cutoff frequency $\Omega_c = 0.6$ using a third order Butterworth analog filter.

Solution: From *Tables-Denominator Polynomial Coefficients for Normalized LPF's*, the transfer function of the analog B-LPF is given by:

$$H_a(s_N) = \frac{1}{1 + 2s_N + 2s_N^2 + s_N^3}$$

De-normalizing yields,

$$H_a(s) = \frac{1}{1 + 2(s/\omega_c) + 2(s/\omega_c)^2 + (s/\omega_c)^3}$$

The cutoff frequency is obtained as follows: $\omega_c = \tan(\Omega_c/2) = \tan(0.3) = 0.3093 \approx 0.3$ rad/s. Hence,

$$H(z) = \frac{1}{1 + 6.6 \frac{z-1}{z+1} + 22.2 \left(\frac{z-1}{z+1} \right)^2 + 37 \left(\frac{z-1}{z+1} \right)^3}.$$

2.6.8.3 MATLAB IIR Filter Design Using the Bilinear Transformation

one can design IIR digital filters in MATLAB using analog prototypes selected according to our requirements. The most important MATLAB filters are `butter`, `cheby1`, `cheby2`, and `ellip`.

Example Using MATLAB, design the following digital IIR filters. Use the bilinear transform and start with either a Butterworth, Chebychev or Elliptic prototype filter:

1. LPF at $f_s = 10$ kHz with cutoff $f_c = 2,000$ Hz, maximum ripple allowed in the passband is $r = 1$ dB, and minimum stopband attenuation = 60 dB starting at 3,000 Hz. Sharpest transition with lowest order is required.
2. HPF with $f_c = 2,000$ Hz, minimum stopband attenuation = 60 dB for frequencies below 1,000 Hz, and other specifications as above.
3. BPF with passband 2,000–3,000 Hz, minimum stopband attenuation = 60 dB for frequencies below $f_1 = 1,000$ Hz or above $f_2 = 4,000$ Hz, and other specifications as above.
4. BSF with stopband 1,000–4,000 Hz, minimum stopband attenuation = 60 dB for frequencies between 2,000 and 3,000 and other specifications as above.

Solution:

1. Since the specification requires the sharpest transition and lowest possible order, an elliptic filter is the appropriate choice for the prototype. The corresponding digital IIR digital elliptic filter has $F_c = f_c/(f_s/2) = 2000/(10k/2) = 0.4$, $F_2 = f_2/(f_s/2) = 0.6$, $r_d = 1$, and $A_{dB} = 60$. The order is obtained as follows:

```
[n fo] = ellipord(Fc,F2,3,AdB)
```

which gives $n = 5$ and $fo = 0.4$. Having found the order one can determine the filter transfer function:

```
[b a]= ellip(n,1,60,fo)
```

where b and a are vectors representing the numerator and the denominator coefficients, respectively.

2. $F_2 = f_2/(f_s/2) = 0.2$, $F_c = 0.4$, Since the other specifications are as for the previous example, an elliptic prototype filter must again be used. Its order is found with the command:

```
[n fon] = ellipord(fcn,f2n,3,AdB)
```

which gives $n = 4$, $f_o = 0.4$. The filter transfer function is determined with the command:

```
[b a] = ellip(n,rdB,AdB,[F1 F2], 'high')
```

3. The solution to this and the following examples proceed similarly to the previous ones. $F_{c1} = 0.4$, $F_{c2} = 0.6$, $F_c = [F_{c1} \ F_{c2}]$, $F_1 = 0.2$, $F_2 = 0.8$, $F = [F_1 \ F_2]$,
`[n fo] = ellipord(Fc, F, 3, AdB)`
 which gives $n = 3$ and $f_o = [0.4 \ 0.6]$.
`[b a] = ellip(n, rdB, AdB, fo)`
4. $F_{c1} = 0.2$, $F_{c2} = 0.8$, $F_c = [F_{c1} \ F_{c2}]$, $F_1 = 0.4$, $F_2 = 0.6$, $F = [F_1 \ F_2]$,
`[n fo] = ellipord(Fc, F, 3, AdB)`
 which gives $n = 3$ and $f_o = [0.4 \ 0.6]$
`[b a] = ellip(n, rdB, AdB, fo, 'stop')`

Figure (2.35) shows the magnitude response of the above filters plotted against the normalized frequency $v = f/f_s$ [note that normalization in MATLAB is different from many other parts of the literature; in its plots MATLAB typically uses the ratio $f/(f_s/2)$ for normalized frequency instead of f/f_s].

2.6.8.4 MATLAB FIR/ IIR Filter Design and Analysis Toolbox

If the following statement is entered on the MATLAB command line:

```
>> fdatool
```

a filter design toolbox will pop up. This toolbox supports many different digital filter design techniques and provides a user-friendly filter design interface. It also enables the user to export the designed filter coefficients to the MATLAB workplace.

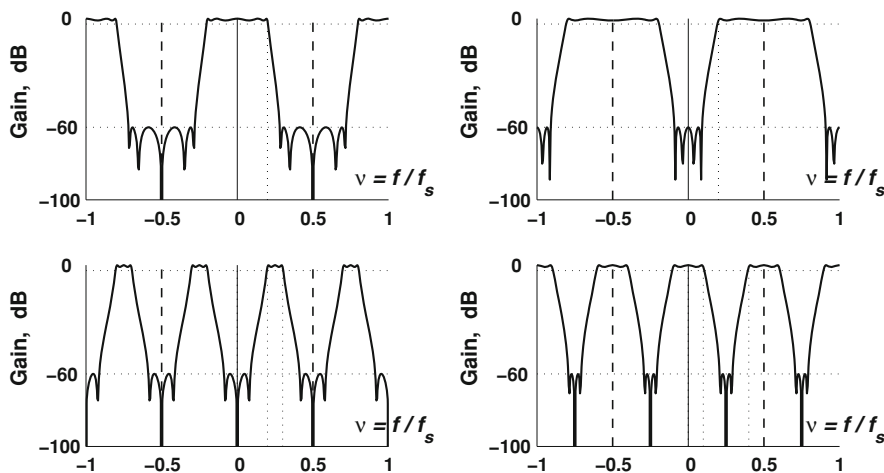


Fig. 2.35 Magnitude response of elliptic IIR filters: LPF, HPF, BPF, and BSF

2.6.9 Applications of IIR Digital Filters

2.6.9.1 The Digital Integrator

If the sampling interval T_s is small, then a digital summer provides a good approximation to an integrator. That is:

$$\int_0^T x(t)dt \approx \sum_{n=0}^N x(n)T_s = T_s \sum_{n=0}^N x(n) \quad (\text{where } N = T/T_s). \quad (2.34)$$

Now

$$y(k) = \sum_{n=0}^k x(n) = x(k) + \sum_{n=0}^{k-1} x(n) = x(k) + y(k-1). \quad (2.35)$$

Hence,

$$\begin{aligned} Y(z) &= X(z) + z^{-1}Y(z) \\ \therefore H(z) &= \frac{1}{1 - z^{-1}}. \end{aligned} \quad (2.36)$$

Therefore, the integrator transfer function is $H_i(z) = T_s H(z)$. The transfer function magnitude and phase is plotted in Fig. (2.36) and it is apparent from the magnitude plot that the digital integrator is low-pass in nature. It is the inverse of the digital differentiator, which is a high-pass filter.

MATLAB the pole-zero diagram can be plotted in MATLAB using `zplane(A,B)`, where A is the vector of numerator coefficients and B is the vector of denominator coefficients. In the above example $A = [1 \ 0]$ and $B = [1 \ -1]$ since $H(z) = z/(z - 1)$.

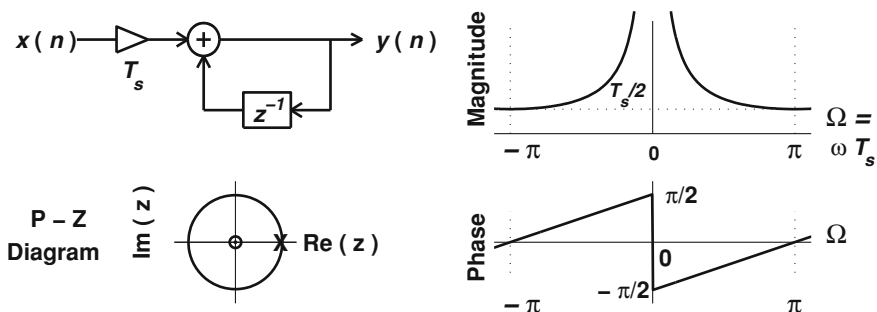


Fig. 2.36 The digital integrator with its frequency response and p-z diagram

2.6.9.2 The Alpha Filter

It was seen in Sect. 2.6.9.1 that the integrator difference equation is $y(n) = T_s[x(n) + y(n-1)]$. If properly scaled, *the integrator can be adapted to become an averager*. Modifying (2.35) to introduce this scaling yields:

$$y(n) = (1 - \alpha)x(n) + \alpha y(n-1). \quad (2.37)$$

where α is a real positive integer which is less than 1. The choice of α depends on the importance of the current sample $x(n)$ as compared to the previous average of data. The transfer function is given by:

$$H(z) = \frac{1 - \alpha}{1 - \alpha z^{-1}} = \frac{(1 - \alpha)z}{z - \alpha},$$

with one pole at $z = \alpha$ and one zero at $z = 0$. The implementation of the alpha filter is depicted in Fig. (2.37). Using *Tables-z Transform Pairs and Theorems*, the impulse response of this filter is given by

$$h(n) = (1 - \alpha)\alpha^n u(n),$$

hence, the output of the system in the time-domain is specified by:

$$\begin{aligned} y(n) &= x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \\ &= (1 - \alpha)[x(n) + \alpha x(n-1) + \alpha^2 x(n-2) + \dots] \end{aligned}$$

Noting that $(1 - \alpha)[1 + \alpha + \alpha^2 + \dots] = 1$, the above I/O formula is reminiscent of the moving average FIR filter, except that the number of “coefficients” is now infinite. This IIR averager is much easier to build than the N -tap FIR moving average filter, with almost *similar performance*. See Fig. (2.37) for a comparison using the same financial data as was used in Fig. (2.27). As in the FIR example of moving average, a few early samples in the plot should be ignored.

Like the integrator, this system behaves as a LPF. *Since averaging is not sensitive to whether the phase response is linear or not*, the alpha filter has a great deal of appeal for many applications (e.g., in SNR estimation for communication systems).

2.6.9.3 The Sinusoidal Digital Oscillator

An oscillator is a system that generates a specific waveform without an input, except perhaps for an initial impulse or a particular setup of the initial conditions.

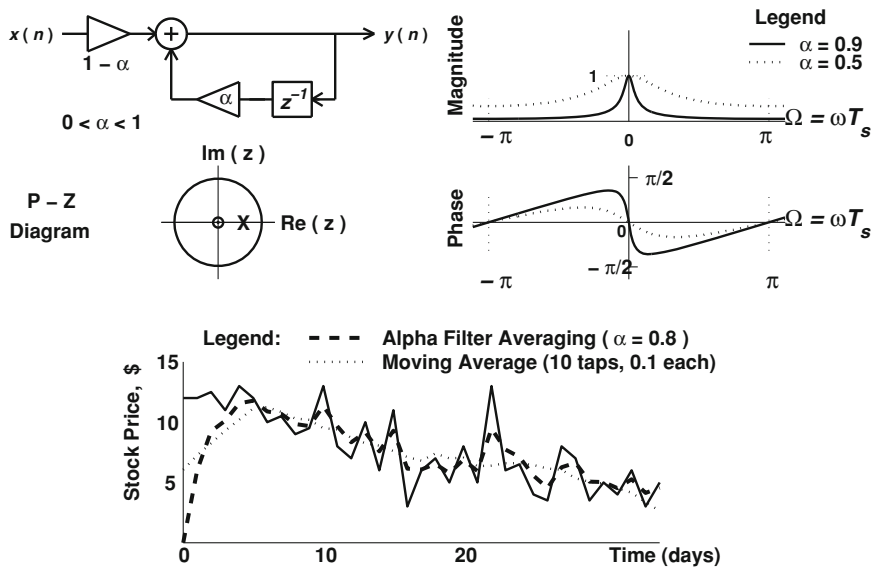


Fig. 2.37 Alpha filter. *Above* Circuit and its frequency response. *Below* data averaging

To design a digital sinusoidal oscillator, one essentially needs a filter whose impulse response is sinusoidal. From *Tables- z Transform Pairs and Theorems*, such a filter transfer function is specified by:

$$h(n) = \sin(bn)u(n) \xleftrightarrow{ZT} H(z) = \frac{\sin(b)z}{z^2 - 2\cos(b)z + 1} \quad (2.38)$$

To build $H(z)$, one can write it as $H(z) = \sin(b)z^{-1}/[1 - 2\cos(b)z^{-1} + z^{-2}]$, the implementation of which is shown in Fig. (2.38). In this expression b corresponds to the normalized radian frequency $\Omega_o = \omega_o T_s$, and hence the frequency of oscillation is:

$$f_o = \omega_o/2\pi = (b/T_s)/2\pi = (b/2\pi)f_s \text{ Hz},$$

provided that $|b| < \pi$. The two poles of this system are the roots of the equation $z^2 - 2\cos(b)z + 1 = 0$, which are given by:

$$p_{1,2} = \cos(b) \pm \sqrt{\cos^2(b) - 1} = \cos(b) \pm j\sin(b) = e^{\pm j b}. \quad (2.39)$$

Hence, the poles are *exactly on the circumference* of the unit circle, as shown in Fig. (2.39), and the system is strictly speaking, *neither stable nor unstable*.

Fig. 2.38 Implementation schemes for a sinusoidal digital oscillator

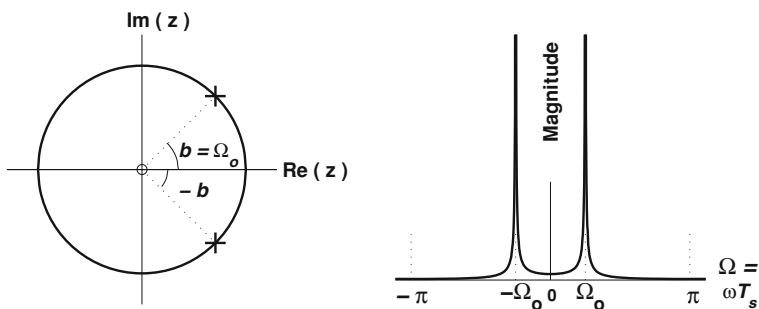
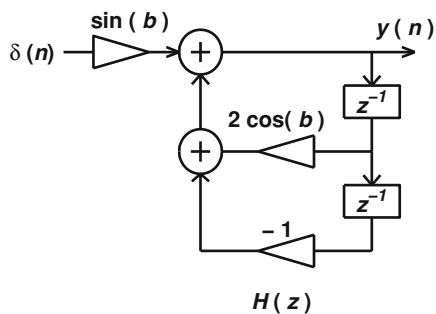


Fig. 2.39 Magnitude response and pole-zero diagram of the digital oscillator

2.6.9.4 The Digital Resonator

The magnitude response of the digital oscillator was seen to consist of two very sharp two spikes. Analogously, one can design a digital resonator, which is a narrowband BPF centered around a resonant frequency f_o . Such a resonator could be useful for extracting a fundamental sinusoid (with frequency f_o) corrupted by harmonics or other sinusoids. Note that if one attempts to design a resonator using classical filter design techniques, one needs a prohibitively large filter order. To ensure stability, the poles should be inside the unit circle, i.e., $p_{1,2} = re^{\pm j\Omega_o}$, with r very near to 1. A possible transfer function for the resonator can then be:

$$H_1(z) = \frac{1}{(z - p_1)(z - p_2)} = \frac{1}{z^2 - 2r \cos(\Omega_o)z + r^2}. \quad (2.40)$$

The magnitude response for this resonator is shown in Fig. (2.40) as the dotted curve. Note that this curve has non-zero values well beyond the frequency of interest, Ω_o .

A transfer function which also satisfies the above criteria with better stopband attenuation is:

$$H(z) = \frac{(z - z_1)(z - z_2)}{(z - p_1)(z - p_2)} = \frac{(z - z_1)(z - z_2)}{z^2 - 2r \cos(\Omega_o)z + r^2}. \quad (2.41)$$

Note that the transfer function in (2.41) has in addition to a pair of poles, two zeros. These zeros are chosen to null the frequency response $H(e^{j\Omega})$ at the digital frequency borders $f = 0$ and $f = \pm f_s/2$. That is:

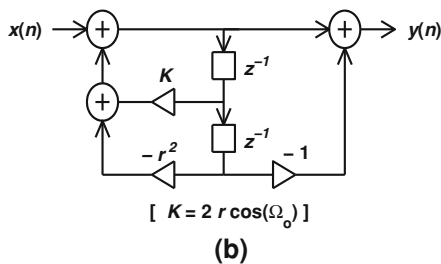
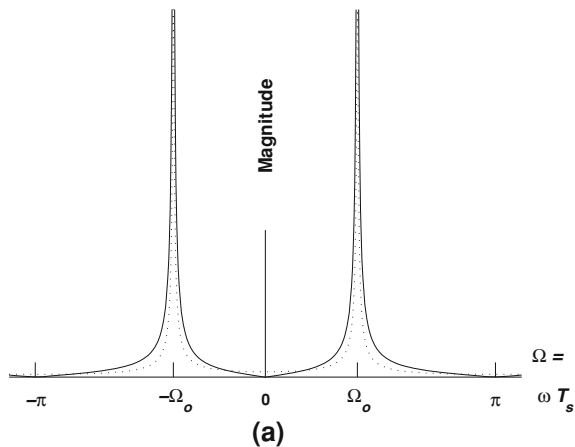
1. $H(e^{j0}) = H(1) = 0$ can be enforced by putting $z_1 = 1$.
2. $H(e^{\pm j\Omega_s/2}) = H(e^{\pm j\pi}) = H(-1) = 0$ can be enforced by putting $z_2 = -1$.

Having the two zeros at these two positions, the frequency response is “pulled down” more effectively away from the positions of frequency resonance. That is, the stopband attenuation improves.

To facilitate implementation of the resonator, it is helpful to write its transfer function as a function of z^{-1} :

$$H(z) = \frac{z^2 - 1}{z^2 - 2r \cos(\Omega_o)z + r^2} = \frac{z^2 - 1}{1 - 2r \cos(\Omega_o)z^{-1} + r^2 z^{-2}}.$$

Fig. 2.40 A digital resonator. **a** Magnitude response with $f_o = 2$ Hz, $r = 0.9$, $f_s = 10$ Hz (Solid with 2 zeros; dotted: without zeros). **b** Implementation using Direct Form-II



This transfer function can be implemented as shown in Fig (2.40). If r is very close to 1, then the 3-dB (half-power) bandwidth B_f and the maximum gain G_m of this system are approximated, respectively, by:

$$B_f \approx \frac{1-r}{\pi} f_s (\text{Hz}) \quad \text{and} \quad G_m \approx \frac{1}{(1-r^2) \sin(\Omega_o)}$$

Equivalently, the bandwidth can be written as:

$B_\omega = 2\pi B_f = 2(1-r)f_s$ (rad/s), (where $\omega = 2\pi f$, radian frequency), $B_v = B_f/f_s = \frac{1-r}{\pi}$, (where $v = ff_s$, normalized frequency), and $B_\Omega = B_\omega/f_s = 2(1-r)$, (where $\Omega = \omega/f_s = 2\pi f/f_s = 2\pi v$, normalized frequency).

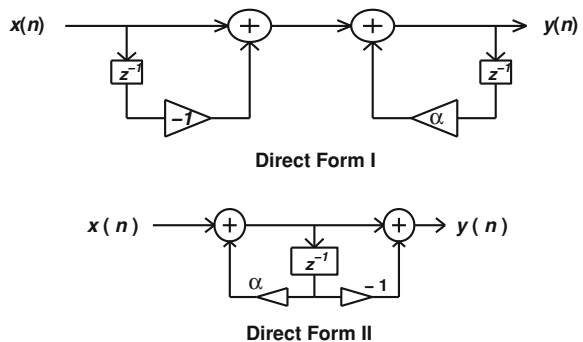
2.6.9.5 A Digital DC Blocker

In some applications an undesirable DC voltage can appear in the information signal. For example, in some audio applications a DC offset can be added to the recorded sound from the microphone. The ADC may also add some unwanted DC to the digitized signal. This DC component carries no information, and in audio applications cannot even be heard. Nonetheless, in some cases it can hinder processing and cause instabilities. For example, the DC component may drive the signal outside the dynamic range of the processing system which will cause signal clipping. Hence, DC removal is often desirable before other forms of processing are pursued.

For DC removal, it is necessary for the magnitude response to be zero at $f = 0$ Hz and unity elsewhere. This kind of frequency response can only be approximated in practice. To block DC one can place a zero at $z = 1$, and to ensure that there is approximately unity gain for non-zero frequencies, one additionally needs a pole very near to the zero at $z = 1$, and inside the unit circle. The following transfer function has the necessary form:

$$H(z) = \frac{1 - z^{-1}}{1 - \alpha z^{-1}}, \quad \alpha \text{ close to } 1. \quad (2.42)$$

Fig. 2.41 A digital DC Blocker



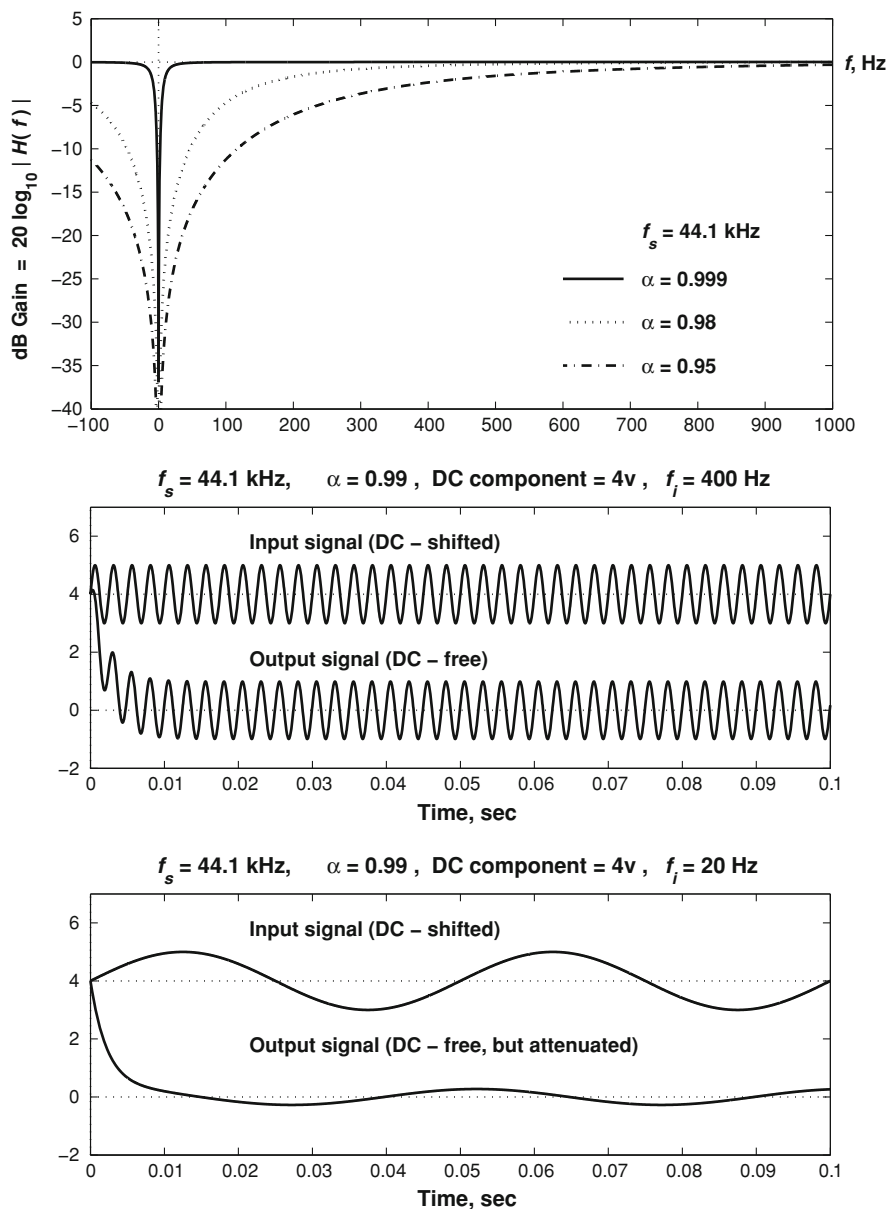


Fig. 2.42 Frequency and time-domain performance of the DC Blocker

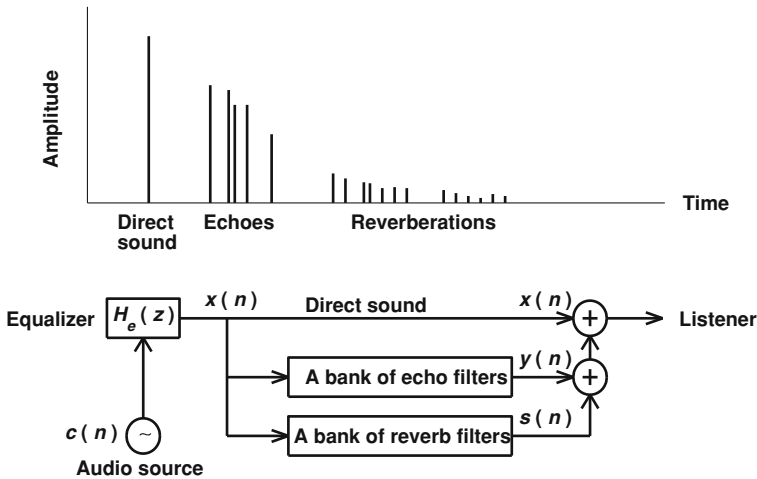


Fig. 2.43 Audio effects. *Above* an audio signal and its reflections in a listening venue. *Below* simulation of specific acoustics in another venue

Figure (2.41) shows a block diagram of a DC blocker, while Fig. (2.42) shows its practical performance with sinusoidal dc-shifted signals. It is evident that this performance is dependent on how close α is to 1. When $\alpha = 0.99$, the filter removes the DC from the (low-frequency) 20 Hz sinusoid, but it also introduces a phase change and an attenuation.

2.6.9.6 An Application of FIR / IIR Digital Filters: Simulation of Acoustic Effects

A piece of music played in a concert hall does not sound the same as if it is played in a living room. This is due to the echoes (early reflections) and the reverberations (late reflections) which vary from setting to setting. One can actually simulate a concert hall in a living room using the following steps:

1. Equalize the audio transfer function of the room, $H_r(z)$. That is, eliminate any special effects that the living room is inherently creating. This is done by first sending an audio impulse $\delta(n)$, and then measuring the impulse response $h_r(n)$. The transfer function is then given by $H_r(z) = \text{fft}\{h_r(n)\}$. Then one designs an equalizing filter $H_e(z) = 1/H_r(z)$, with say the Remez algorithm.
2. Simulate the echoes and the reverberations as follows:
 - For echoes: $y(n) = x(n) + \alpha x(n - N)$. This is so because an echo is a direct reflection of the delayed input signal. Hence, $H(z) = 1 + \alpha z^{-N}$. This is an FIR filter. In practice, choose $NT_s \geq 0.05$

- For reverberations: $s(n) = x(n) + \beta \cdot s(n - M)$. This is so because a reverberation is an accumulation of previous reflections of the signal. Hence, $H(z) = 1/(1 - \beta z^{-M})$. This is an IIR filter with M poles.

Figure (2.43) shows a block diagram of the above procedure.

References

1. Harris, F.J.: On the use of windows for harmonic analysis with the discrete Fourier transform. Proc. IEEE **66**, 51–83 (1978)
2. Oppenheim, A.V., Schafer R.: Discrete-Time Signal Processing, Prentice-Hall, USA (1989)

Digital Signal Processing

An Introduction with MATLAB and Applications

Hussain, Z.M.; Sadik, A.Z.; O'Shea, P.

2011, XXI, 350 p., Hardcover

ISBN: 978-3-642-15590-1