

Digital Signal Processing: A Computer Science Perspective
Jonathan Y. Stein
Copyright © 2000 John Wiley & Sons, Inc.
Print ISBN 0-471-29546-9 Online ISBN 0-471-20059-X

Part I

Signal Analysis

Signals

We are now ready to commence our study of signals and signal processing systems, the former to be treated in Part I of this book and the latter in Part II. Part III extends the knowledge thus gained by presentation of specific algorithms and computational architectures, and Part IV applies all we will have learned to communications and speech signal processing.

At times one wants to emphasize signals as basic entities, and to consider systems as devices to manipulate them or to measure their parameters. The resulting discipline may then be called *signal analysis*. At other times it is more natural to consider systems as the more fundamental ingredients, with signals merely inputs and outputs to such systems. The consequence of this viewpoint is called *signal processing*. This term is also most commonly used when it is not clear which aspect one wishes to stress.

In this chapter we introduce the concept of a signal. We will see that there are analog signals and digital signals, and that under certain conditions we can convert one type into the other. We will learn that signals can be described in terms of either their time or frequency characteristics, and that here too there are ways to transform one description into the other. We present some of the simplest signals, and discover that arbitrary signals can be represented in terms of simple ones. On the way we learn how to perform arithmetic on signals, and about the connection between signals and vectors.

2.1 *Signal Defined*

The first question we must ask when approaching the subject of signal analysis is ‘What exactly do we mean by *signal*?’ The reader may understand intuitively that a signal is some function of time that is derived from the physical world. However, in scientific and technological disciplines it is customary to provide formal mathematical definitions for the main concepts, and it would be foolish to oppose this tradition. In order to answer the question satisfactorily, we must differentiate between analog and digital signals.

Definition: signal

An *analog signal* s is a finite real-valued function $s(t)$ of a continuous variable t (called time), defined for all times on the interval $-\infty < t < +\infty$. A *digital signal* s is a bounded discrete-valued sequence s_n with a single index n (called discrete time), defined for all times $n = -\infty \dots +\infty$. ■

The requirement that analog signals be *real-valued*, rather than integer or complex, has its origin in the notion that *real-world* signals, such as speeds, voltages, and acoustic pressures, are simple continuous variables. Complex numbers are usually considered purely mathematical inventions that can never appear in nature. Digital signals are constrained more by the requirement of representability in a digital computer than by physical realizability. What we mean here by ‘discrete’ is that the possible values are *quantized* to discrete values, such as integers or all multiples of 2^{-b} . ‘Bounded’ means that there are only a finite number of possible signal values. Bounded discrete values are exactly the kinds of numbers represented by computer words with some finite number of bits.

Finiteness is another *physical* requirement, and comes in three varieties, namely finite signal value, finite energy, and finite bandwidth. Finite-valuedness simply means that the function desiring to be a signal must never diverge or become mathematically singular. We are quite confident that true physical quantities never become infinite since such behavior would require infinite energy or force or expense of one type or another. Digital signals are necessarily bounded in order to be representable, and so are always finite valued. The range over which a signal varies is called its *dynamic range*. Finite energy and finite bandwidth constraints are similarly grounded, but the concepts of energy and bandwidth require a little more explanation for the uninitiated.

Energy is a measure of the *size* of a signal, invented to enable the analyst to compare the infinitely many possible signals. One way to define such a measure might be to use the highest value the signal attains (and thus finite energy would imply finite signal value). This would be unsatisfactory because a generally small signal that attains a high value at one isolated point in time would be regarded as larger than a second signal that is almost always higher than the first. We would certainly prefer a measure that takes all times into account. Were signals to have only positive values we could possibly use the average signal value, but since they are not the average is ineffectual as many seemingly large signals (e.g., $A \sin(\omega t)$ with large A) have zero average due to positive and negative contributions cancelling. The simplest satisfactory measure is given by the following definition.

Definition: energy

The energy of an analog or digital signal s is defined to be

$$E_s = \int_{-\infty}^{\infty} s^2(t) dt \quad \text{A} \quad \Bigg| \quad \text{D} \quad E_s = \sum_{n=-\infty}^{\infty} s_n^2 \quad (2.1)$$

the sum (or integral for the analog case) of the signal's values squared. ■

This measure is analogous to the squared length of multidimensional vectors, and is proportional to the physical quantity known as energy when the signal is a velocity, voltage, or current. The energy we have just defined is also directly related to the expense involved in producing the signal; this being the basis for the physical requirement of finite energy. The square root of the energy defines a kind of average signal value, called the **Root Mean Squared (RMS)** value.

Bandwidth is a measure not of size but of speed, the full discussion of which we must postpone until after the notion of spectrum has been properly introduced. A signal that fluctuates rapidly has higher bandwidth than one that only varies slowly. Requiring finite bandwidth imposes a smoothness constraint, disallowing sudden jump discontinuities and sharp corners. Once again such functions violate what we believe nature considers good taste. Physical bodies do not disappear from one place and appear in another without traveling through all points in between. A vehicle's velocity does not go from zero to some large value without smoothly accelerating through intermediate speeds. Even seemingly instantaneous ricochets are not truly discontinuous; filming such an event with a high-speed camera would reveal intermediate speeds and directions.

Finally, the provision *for all times* really means *for all times of interest*, and is imposed in order to disallow various pathological cases. Certainly a body no longer has a velocity once destroyed, and a voltage is meaningless once the experimental apparatus is taken apart and stored. However, we want the experimental values to *settle down* before we start observing, and wish our phenomena to exist for a reasonable amount of time after we stop tending to them.

Now that we fully understand the definition of *signal*, we perceive that it is quite precise, and seemingly inoffensive. It gives us clear-cut criteria for determining which functions or sequences are signals and which are not, all such criteria being simple physical requirements that we would not wish to forgo. Alas this definition is more honored in the breach than the observance. We shall often relax its injunctions in the interests of mathematical

simplicity, and we permit ourselves to transgress its decrees knowing full well that the ‘signals’ we employ could never really exist.

For example, although the definition requires signals to be *real-valued functions*, we often use complex values in order to simplify the algebra. What we really mean is that the ‘real’ signal is the real part of this complex signal. This use of an ‘imaginary’ complex signal doesn’t overly bother us for we know that we *could* reach the same conclusions using real values, but it would take us longer and we would be more apt to make mistakes. We even allow entities that aren’t actually functions at all, when it saves us a few lines of proof text or program code!

Our definition relies on the existence of a *time* variable. At times the above definition is extended to functions of other time-like independent variables, and even to functions of more than one variable. In particular, *image processing*, that deals with functions of two spatial coordinates, invokes many signal processing concepts. However, in most of this book we will not consider image processing to be part of signal processing. Although certain basic ideas, notably filtering and spectral analysis, *are* common to both image and signal processing, the truly strong techniques of each are actually quite different.

We tend to scoff at the requirement for finite-valuedness and smoothness, routinely utilizing such nonphysical constructs as tangents and square waves, that possess an infinite number of discontinuities! Once again the reader should understand that real-world signals can only approximate such behavior, and that such refractory functions are introduced as mathematical scaffolding.

Of course signals are defined over an infinite range of times, and consequently for a signal’s energy to be finite the signal must be zero over most times, or at least decay to zero sufficiently rapidly. Strictly requiring finite energy would rule out such useful signals as constants and periodic functions. Accordingly this requirement too is usually relaxed, with the understanding that outside the interval of time we observe the signal, it may well be set to zero. Alternatively, we may allow signals to be nonzero over infinite times, but to have finite *power*. Power is the energy per time

$$P_s(\tau) = \lim_{T \rightarrow 0} \frac{1}{T} \int_{\tau - \frac{T}{2}}^{\tau + \frac{T}{2}} s^2(t) dt \quad \mathbf{A} \left| \mathbf{D} \quad P_{s_\nu} = \lim_{N \rightarrow 0} \frac{1}{N} \sum_{n=\nu - \frac{N}{2}}^{\nu + \frac{N}{2}} s_n^2 \quad (2.2)$$

which is time-dependent in general.

Hence although the definition we gave for *signal* is of good intent, its dictates go unheeded; there is scarcely a single clause in the definition that

we shan't violate at some time or other. In practice entities are more often considered signals because of the utility in so doing, rather than based on their obeying the requirements of this definition (or any other).

In addition to all its possibly ignorable requirements, our definition also leaves something out. It is quiet about any possible connection between analog and digital signals. It turns out that a digital signal can be obtained from an analog signal by **A**nalog to **D**igital conversion (the 'A/D' of Figure 1.3) also known as *sampling* and *digitizing*. When the sampling is properly carried out, the digital signal is somehow equivalent to the analog one. An analog signal can be obtained from a digital signal by **D**igital to **A**nalog conversion (the 'D/A' block), that surprisingly suffers from a dearth of alternative names. Similar remarks can be made about equivalence. A/D and D/A conversion will be considered more fully in Section 2.7.

EXERCISES

- 2.1.1 Which of the following are *signals*? Explain which requirement of the definition is possibly violated and why it is acceptable or unacceptable to do so.
1. the height of Mount Everest
 2. $(e^{it} + e^{-it})$
 3. the price of a slice of pizza
 4. the 'sinc' function $\frac{\sin(t)}{t}$
 5. Euler's totient function $\phi(n)$, the number of positive integers less than n having no proper divisors in common with n
 6. the water level in a toilet's holding tank
 7. $\lfloor t \rfloor$ the greatest integer not exceeding t
 8. the position of the tip of a mosquito's wing
 9. \sqrt{t}
 10. the Dow Jones Industrial Average
 11. $\sin(\frac{1}{t})$
 12. the size of water drops from a leaky faucet
 13. the sequence of values x_n in the interval $[0 \dots 1]$ defined by the *logistics recursion* $x_{n+1} = \lambda x_n(1 - x_n)$ for $0 \leq \lambda \leq 4$
- 2.1.2 What is the power of $s(t) = A \sin(\omega t)$? The RMS value?
- 2.1.3 A signal's *peak factor* is defined to be the ratio between its highest value and its RMS value. What is the peak factor for $s(t) = A \sin(\omega t)$? The sum of N sinusoids of different frequencies?

2.1.4 Define a size measure M for signals *different* from the energy (or RMS value). This measure should have the following properties.

- The zero signal must have zero measure $M_0 = 0$, and no other signal should have zero measure.
- If signal y is identical to signal x shifted in time then $M_y = M_x$.
- If $y_n = \alpha x_n$ for all times, then $M_y > M_x$ if $\alpha > 1$ and $M_y < M_x$ if $\alpha < 1$.
- If $y_n > x_n$ almost all of the time, then $M_y > M_x$.

What advantages and disadvantages does your measure have in comparison with the energy?

2.2 The Simplest Signals

Let us now present a few signals, ones that will be useful throughout our studies. The simplest signal is the *unit constant*, that is, $s(t) = 1$ in analog time or $s_n = 1$ in digital time.

$$s(t) = 1 \quad \text{A} \quad \Bigg| \quad \text{D} \quad s_n = 1 \quad (2.3)$$

Although this is the simplest signal we can imagine, it has infinite energy, and therefore violates one of the finiteness constraints. Hence technically it isn't really a signal at all! Arbitrary constant signals can be obtained by multiplying the unit constant signal by appropriate values. The constant signal, depicted in Figure 2.1, although admittedly trivial, can still be useful. We will often call it **D**irect **C**urrent (DC), one of the many electronics

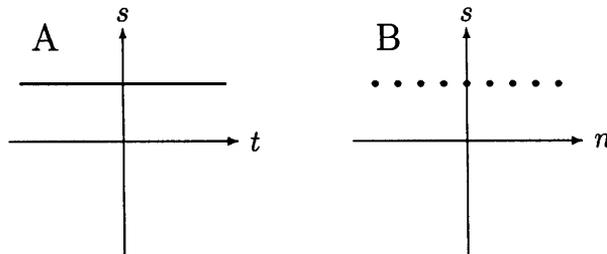


Figure 2.1: The constant signal. In (A) we depict the analog constant and in (B) the digital constant.

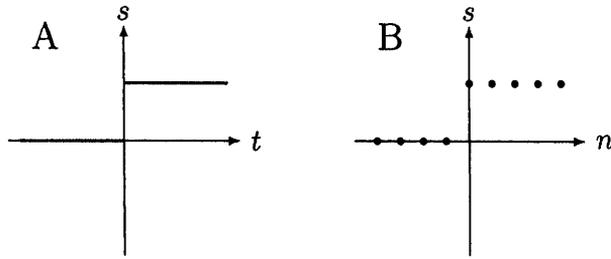


Figure 2.2: The unit step signal. In (A) we depict the analog step $u(t)$ and in (B) the digital step u_n .

terms imported into signal processing. The gist is that a battery's voltage is constant, $v(t) = V_0$, and consequently induces a current that always flows in one direction. In contrast the voltage from a wall outlet is sinusoidal, $v(t) = V_0 \sin(\omega t)$, and induces an **Alternating Current (AC)**.

We cannot learn much more from this signal, which although technically a 'function of time' in reality is not time dependent at all. Arguably the simplest time-dependent signal is the *unit step*, which changes value at only one point in time (see Figure 2.2). Mathematically, the analog and digital unit step signals are:

$$u(t) = \Theta(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases} \quad \mathbf{A} \quad \mathbf{D} \quad u_n = \Theta(n) = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases} \quad (2.4)$$

respectively. In some of the literature the step function is called *Heaviside's step function*. Once again the finite energy requirement is unheeded, and in the analog version we have a jump discontinuity as well. Here we have set our clocks by this discontinuity, that is, we arranged for the change to occur at time zero. It is a simple matter to translate the transition to any other time; $u(t - T)$ has its discontinuity at $t = T$ and u_{n-N} has its step at $n = N$. It is also not difficult to make step functions of different sizes $Au(t)$ and Au_n , and even with any two levels $Au(t) + B$ and $Au_n + B$. The unit step is often used to model phenomena that are 'switched on' at some specific time.

By subtracting a digital unit step shifted one to the right from the unshifted digital unit step we obtain the digital unit impulse. This signal, depicted in Figure 2.3.B, is zero everywhere except at time zero, where it is unity. This is our first true signal, conforming to all the requirements of our definition. In Chapter 6 we will see that the unit impulse is an invaluable tool in the study of systems. Rather than invent a new mathematical

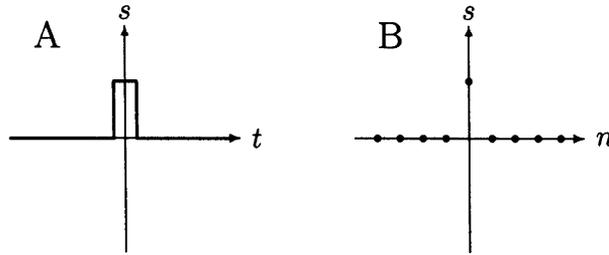


Figure 2.3: The unit impulse. In (A) we depict an analog impulse of unity width. In (B) the digital unit impulse $\delta_{n,0}$.

symbol for this signal, we utilize one known as the *Kronecker delta* $\delta_{n,m}$. This doubly indexed entity is defined to be one, if and only if its indices are equal; otherwise it is zero. In terms of the Kronecker delta, the digital unit impulse is $s_n = \delta_{n,0}$.

The full Kronecker delta corresponds to a **Shifted Unit Impulse (SUI)**

$$s_n = \delta_{n,m} \quad (2.5)$$

that is zero for all times except for time $n = m$, when it equals one. The importance of the set of all SUIs will become clear in Section 2.5.

One might similarly define an analog unit impulse by subtracting analog unit steps, obtaining the Figure 2.3.A. This analog signal flagrantly displays two jump discontinuities, but by now that should not make us feel uncomfortable. However, this is not the signal usually referred to as the analog unit impulse. There is no profound meaning to the width of this signal, since in the analog world the meaning of a unit time interval depends on the time units! What *is* meaningful is the *energy* of the impulse, which is its amplitude squared times its width. There are good reasons to expect that once the width is small enough (i.e., small compared to all significant times in the problem) all impulses with the same energy will have basically the same effect on systems. Accordingly, when one speaks of a ‘unit impulse’ in the analog domain, conventionally this alludes to a ‘unit energy’ impulse. Of course the unit width impulse in Figure 2.3 is a unit impulse in this sense; but so are all the others in Figure 2.4.

The unit energy impulses in the figure are given by:

$$I(t) = \begin{cases} 0 & |t| > T \\ \frac{1}{\sqrt{2T}} & |t| < T \end{cases}$$

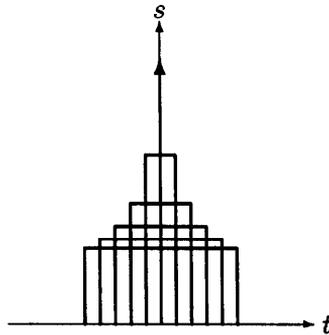


Figure 2.4: Analog unit energy impulses. Since all of these signals have the same energy, the height increases as the width decreases. The vertical arrow is a symbolic way of designating Dirac's delta function.

where T is the width. In the limit $T \rightarrow 0$ we obtain a mathematical entity called Dirac's delta function $\delta(t)$, first used by P.A.M. Dirac in his mathematical description of quantum physics. The name *delta* is purposely utilized to emphasize that this is the 'analog analog' of Kronecker's delta. The word *function* is a misnomer, since Dirac's delta is not a true function at all. Indeed, Dirac's delta is defined by the two properties:

- $\delta(t)$ is zero everywhere except at the origin $t = 0$
- the integral of the delta function is unity $\int_{-\infty}^{\infty} \delta(t) dt = 1$

and clearly there can be no such function! However, Dirac's delta is such an extremely useful abstraction, and since its use *can* be justified mathematically, we shall accept it without further question. Indeed, Dirac's delta is *so* useful, that when one refers without further qualification to the analog unit impulse, one normally means $\delta(t)$.

$$s(t) = \delta(t) \quad \mathbf{A} \mid \mathbf{D} \quad s_n = \delta_{n,0} \quad (2.6)$$

The next signal we wish to discuss is the square wave $\square(t)$, depicted in Figure 2.5.A. It takes on only two values, ± 1 , but switches back and forth between these values periodically. One mathematical definition of the analog square wave is

$$\square(t) = \begin{cases} 1 & [t] \text{ is even} \\ -1 & [t] \text{ is odd} \end{cases} \quad (2.7)$$

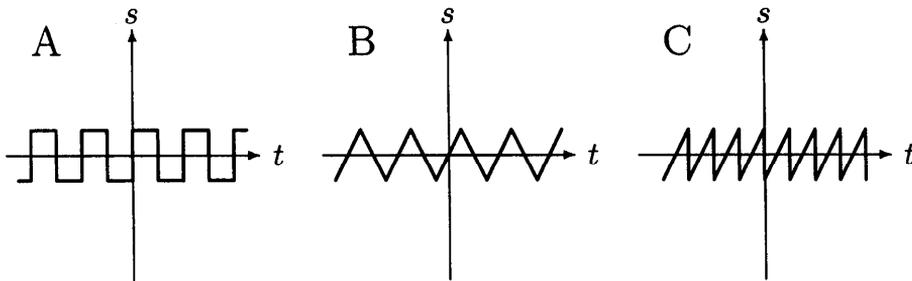


Figure 2.5: Three periodic analog signals. In (A) we depict the square wave, in (B) the triangle wave and in (C) the sawtooth.

where $\lfloor t \rfloor$ (pronounced ‘floor’ of t) is the greatest integer less than or equal to the real number t . We have already mentioned that this signal has an infinite number of jump discontinuities, and it has infinite energy as well! Once again we can stretch and offset this signal to obtain any two levels, and we can also change its *period* from unity to T by employing $\square(t/T)$. We can further generalize the square wave to a rectangular wave by having it spend more time in one state than the other. In this case the percentage of the time in the higher level is called the *duty cycle*, the standard square wave having a 50% duty cycle. For digital signals the minimal duty cycle signal that is not a constant has a single high sample and all the rest low. This is the periodic unit impulse

$$d_n = \sum_m \delta_{n,mP} \quad (2.8)$$

where the period is P samples.

Similarly we can define the analog triangle wave $\Delta(t)$ of Figure 2.5.B and the sawtooth $\mathcal{T}(t)$ of Figure 2.5.C. Both, although continuous, have slope discontinuities. We leave the mathematical definitions of these, as well as the plotting of their digital versions, to the reader. These signals pop up again and again in applications. The square wave and its close brethren are useful for triggering comparators and counters, the triangle is utilized when constant slope is required, and the sawtooth is vital as the ‘time base’ of oscilloscopes and the ‘raster scan’ in television. Equipment known as ‘function generators’ are used to generate these signals.

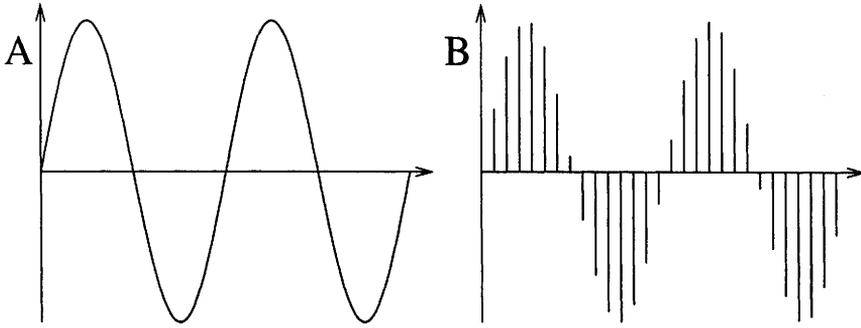


Figure 2.6: Sinusoidal signals. In (A) we depict the analog sinusoid with given amplitude, frequency and phase. In (B) the digital sinusoid is shown.

Of course the most famous periodic signal is none of these, but the sine and cosine functions, either of which we call a *sinusoid*.

$$s(t) = \sin(2\pi ft) \quad \text{A} \left| \text{D} \quad s_n = \sin(2\pi f_d n) \quad (2.9)$$

The connection between the *frequency* f of an analog sinusoid and its period T can be made clear by recalling that the sine function completes a full cycle after 2π radians. Accordingly, the frequency is the reciprocal of the period

$$f = \frac{1}{T}$$

and its units must be *full cycles per second*, also known as Hertz or Hz. The period represents the number of seconds per cycle while the frequency in Hz describes the number of full cycles per second. Since discrete time n carries no units, the *digital frequency* f_d will be essentially a pure number. The periodicity of digital sinusoids will be discussed later.

In order to avoid factors of 2π we often rewrite equation 2.9 as follows.

$$s(t) = \sin(\omega t) \quad \text{A} \left| \text{D} \quad s_n = \sin(\omega_d n) \quad (2.10)$$

Since the argument of a trigonometric function must be in radians (or degrees), the units of the *angular frequency* $\omega = 2\pi f$ must be *radians per second*, and those of the *digital angular frequency* $\omega_d = 2\pi f_d$ simply *radians*.

In many respects $\sin(t)$ is very similar to $\square(t)$ or $\Delta(t)$, but it possesses a major benefit, its smoothness. Sinusoids have neither jump nor slope discontinuities, elegantly oscillating back and forth (see Figure 2.6.A). More general sinusoids can be obtained by appropriate mathematical manipulation

$$A \sin(\omega t + \phi) + B$$

where A is called the *amplitude*, ω the frequency, ϕ the *phase*, and B the *DC* component. Sines of infinite time duration have infinite energy, but are otherwise eminent members of the signal community. Sinusoidal signals are used extensively in all facets of signal processing; communications are carried by them, music is modeled as combinations of them, mechanical vibrations are analyzed in terms of them, clocks are set by comparing to them, and so forth.

Although the signals $\sin(\omega t)$ and $\cos(\omega t)$ look exactly the same when viewed separately, when several signals are involved the relative phases become critical. For example, adding the signal $\sin(\omega t)$ to another $\sin(\omega t)$ produces $2\sin(\omega t)$; adding $\sin(\omega t)$ to $\cos(\omega t)$ creates $\sqrt{2}\sin(\omega t + \frac{\pi}{4})$; but adding $\sin(\omega t)$ to $\sin(\omega t + \pi) = -\sin(\omega t)$ results in zero. We can conclude that when adding sinusoids $1 + 1$ doesn't necessarily equal 2; rather it can be anything between 0 and 2 depending on the phases. This addition operation is analogous to the addition of vectors in the plane, and many authors define *phasors* in order to reduce sinusoid summation to the more easily visualized vector addition. We will not need to do so, but instead caution the reader to take phase into account whenever more than one signal is present.

Another basic mathematical function with a free parameter that is commonly employed in signal processing is the exponential signal

$$s(t) = e^{\Lambda t} \quad \mathbf{A} \left| \mathbf{D} \quad s_n = e^{\Lambda_d n}$$

depicted in Figure 2.7 for negative Λ . For positive Λ and any finite time this function is finite, and so technically it is a well-behaved signal. In practice the function explodes violently for even moderately sized negative times, and unless somehow restricted does not correspond to anything we actually see in nature. Mathematically the exponent has unique qualities that make it ideal for studying signal processing systems.

We shall now do something new; for the first time we will allow complex-valued functions. We do this by allowing the constant in the argument of the exponential to be a pure imaginary number $\Lambda = i\omega$, thus radically chang-

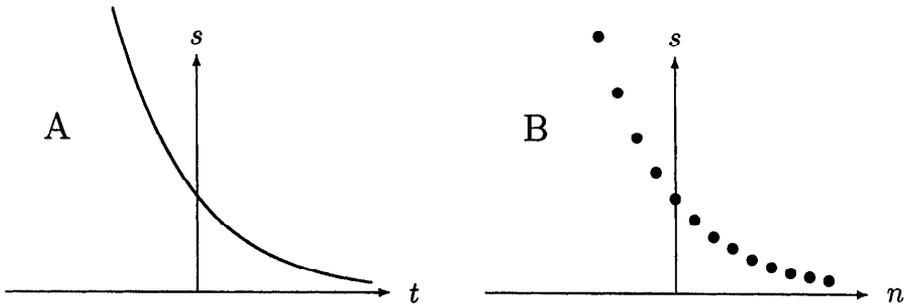


Figure 2.7: Exponentially decreasing signals. In (A) we depict the analog exponential, in (B) the digital.

ing the character of the signal. Recalling the remarkable identity (equation (A.7))

$$e^{i\varphi} = \cos(\varphi) + i\sin(\varphi)$$

we see that exponentials with imaginary coefficients are complex sinusoids.

$$Ae^{i\omega t} = A\cos(\omega t) + iA\sin(\omega t)$$

When we deal with complex signals like $\mathbf{s}(t) = Ae^{i\omega t}$, what we really mean is that the real-world signal is the real part

$$s(t) = \Re \mathbf{s}(t) = A\cos(\omega t)$$

while the imaginary part is just that—imaginary. Since the imaginary part is 90° (one quarter of a cycle) out of phase with the real signal, it is called the *quadrature component*. Hence the complex signal is composed of in-phase (real) and quadrature (imaginary) components.

At first it would seem that using complex signals makes things more *complex* but often the opposite is the case. To demonstrate this, consider what happens when we multiply two sinusoidal signals $s_1(t) = \sin(\omega_1 t)$ and $s_2(t) = \sin(\omega_2 t)$. The resulting signal is

$$s(t) = s_1(t)s_2(t) = \sin(\omega_1 t)\cos(\omega_2 t) + \cos(\omega_1 t)\sin(\omega_2 t)$$

which is somewhat bewildering. Were we to use complex signals, the product would be easy

$$\mathbf{s}(t) = \mathbf{s}_1(t)\mathbf{s}_2(t) = e^{i\omega_1 t}e^{i\omega_2 t} = e^{i(\omega_1 + \omega_2)t}$$

due to the symmetries of the exponential function. The apparent contradiction between these two results is taken up in the exercises.

A further variation on the exponential is to allow the constant in the argument of the exponential to be a complex number with both real and imaginary parts $\Lambda = \lambda + i\omega$. This results in

$$s(t) = Ae^{(\lambda+i\omega)t} = Ae^{\lambda t} \cos(\omega t) + iAe^{\lambda t} \sin(\omega t) \quad (2.11)$$

corresponding to the real signal

$$s(t) = Ae^{\lambda t} \cos(\omega t) \quad (2.12)$$

which combines the exponential with the sinusoid. For negative λ , this is a damped sinusoid, while for positive λ it is an exponentially growing one.

Summarizing, we have seen the following archetypical simple signals:

unit constant	$s(t) = 1$	$s_n = 1$
unit step	$s(t) = u(t)$	$s_n = u_n$
unit impulse	$s(t) = \delta(t)$	$s_n = \delta_{n,0}$
square wave	$s(t) = \square(\omega t)$	$s_n = \square(\omega_d n)$
sinusoid	$s(t) = A \sin(\omega t + \phi)$	$s_n = A \sin(\omega_d n + \phi)$
damped sinusoid	$s(t) = Ae^{-\lambda t} \sin(\omega t + \phi)$	$s_n = A\alpha^{-n} \sin(\omega_d n + \phi)$
real exponential	$s(t) = Ae^{\lambda t}$	$s_n = \alpha^n$
complex sinusoid	$s(t) = Ae^{i(\omega t + \phi)}$	$s_n = Ae^{i(\omega_d n + \phi)}$
damped complex sinusoid	$s(t) = Ae^{(\lambda+i\omega)t}$	$s_n = Ae^{(\lambda+i\omega_d)n}$

EXERCISES

- 2.2.1 Thomas Alva Edison didn't believe that AC electricity was useful, since the current first went one way and then returned. It was Nikola Tesla who claimed that AC was actually better than DC. Why was Edison wrong (hint: energy) and Tesla right (hint: 'transformers')?
- 2.2.2 In the text we depicted digital signals graphically by placing dots at signal values. We will usually use such *dot graphs*, but other formats are prevalent as well. A *comb graph* uses lines from the time axis to the signal point; a *slint graph* (straight line interpolation) simply connects successive signal values; comb-dot and slint-dot combinations are useful when the signal takes on zero values. These formats are depicted in Figure 2.8. Write general routines for plotting digital signals in these formats in whatever computer programming language you usually use. Depending on your programming language you may first have to prepare low-level primitives. Plot the digital sinusoidal signal $s_n = \sin(\omega_d n)$ for various frequencies ω in all of these formats. Decide which you like the best. You may use this format from now on.

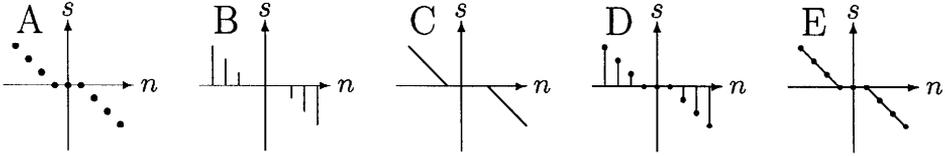


Figure 2.8: Different formats for graphical representation of digital signals. In (A) we depict a signal using our usual *dot graph*. In (B) the same signal is plotted as a *comb graph*. In (C) it is graphed as a *slint graph*. (D) and (E) are *comb-dot* and *slint-dot* representations respectively.

- 2.2.3 Give mathematical definitions for the analog triangle signal $\Delta(t)$ of Figure 2.5.B and for the analog sawtooth $\text{saw}(t)$ of Figure 2.5.C.
- 2.2.4 What is the integral of the square wave signal? What is its derivative?
- 2.2.5 Using your favorite graphic format plot the digital square wave, triangle wave and sawtooth, for various periods.
- 2.2.6 Perform the following experiment (you will need an assistant). Darken the room and have your assistant turn on a pen-flashlight and draw large circles in the air. Observe the light from the side, so that you see a point of light moving up and down. Now have the assistant start walking while still drawing circles. Concentrate on the vertical and horizontal motion of the point of light, disregarding the depth sensation. You should see a sinusoidal signal. Prove this. What happens when you rotate your hand in the opposite direction? What can you infer regarding negative frequency sinusoids?
- 2.2.7 Dirac's delta function can be obtained as the limit of sequences of functions other than those depicted in Figure 2.4. For example,

$$\begin{array}{ll}
 \text{asymmetric unit impulses} & \mathcal{I}_T(t) = \begin{cases} 0 & t < 0 \\ \frac{1}{\sqrt{T}} & 0 < t < T \\ 0 & t > T \end{cases} \\
 \text{Gaussian functions} & G_\sigma(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \\
 \text{Sinc functions} & \frac{1}{\pi} \text{sinc}_\omega(t) = \frac{\sin(\omega t)}{\pi t} \\
 \text{Lorentzian functions} & \mathcal{L}(t) = \frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + t^2}
 \end{array}$$

Graph these functions for decreasing T , ϵ and increasing σ , ω , graphically showing the appearance of the Dirac delta. What new features appear? Show that in the proper limit these functions approach zero for all nonzero times.

- 2.2.8 The integral of the analog impulse $\delta(t)$ is the unit step $u(t)$, and conversely the derivative of $u(t)$ is $\delta(t)$. Explain these facts and depict graphically.

2.2.9 Explain the following representation of Dirac's delta.

$$\delta(t) = \frac{1}{2} \frac{d}{dx} |x|$$

2.2.10 Show that

$$\int_{-\infty}^{\infty} f(t) \delta(t - t') dt = f(t')$$

both graphically and by using basic calculus. From this result show that $\delta(t)$ must be zero for all nonzero arguments. Compare the above relation with the Fourier identity

$$f(t') = \frac{1}{2\pi} \int_{-\infty}^{\infty} du \int_{-\infty}^{\infty} dt f(t) e^{iu(t-t')}$$

and derive an integral representation for the Dirac delta. What meaning can be given to the derivative of the Dirac delta?

2.2.11 Plot the analog complex exponential. You will need to simultaneously plot two sinusoids in such fashion that one is able to differentiate between them. Extend the routines you wrote in the previous exercise to handle the digital complex exponential.

2.2.12 Explain *why* the real signal corresponding to the product of two complex exponentials is *not* the same as the product of the two real sinusoids.

2.3 Characteristics of Signals

Now that we have some experience with signals, let us discuss some general characteristic signals can have. Signals are characterized as being:

- deterministic or stochastic
- if deterministic: periodic or nonperiodic
- if stochastic: stationary or nonstationary
- of finite or infinite time duration
- of finite bandwidth or of full spectrum

Perhaps the most significant characteristic of a signal is whether it is deterministic or stochastic. Deterministic signals are those that are generated by some nonprobabilistic algorithm. They are thus reproducible, predictable

(at least over short time scales—but see Section 5.5) and well-behaved mathematically. Stochastic signals are generated by systems that contain randomness (see Section 5.6). At any particular time the signal is a *random variable*, (see Appendix A.13), which may have well defined average and variance, but is not completely defined in value. Any particular sequence of measurements of the signal's values at various times captures a specific instantiation of the stochastic signal, but different sequence of measurements under the same conditions would retrieve somewhat different values.

In practice we never see a purely deterministic signal, since even the purest of deterministic signals will inevitably become contaminated with *noise*. 'Pure noise' is the name we give to a quintessential stochastic signal, one that has only probabilistic elements and no deterministic ones. When a deterministic signal becomes contaminated with additive noise, as depicted in Figure 2.9,

$$y(t) = x(t) + n(t)$$

we can quantify its 'noisiness' by the **Signal to Noise Ratio (SNR)**. The SNR is defined as the ratio of the signal energy to the noise energy, and is normally measured in *dB*. (equation (A.16))

$$\text{SNR(dB)} = 10 \log_{10} \frac{E_x}{E_n} = 10 (\log_{10} E_x - \log_{10} E_n) \quad (2.13)$$

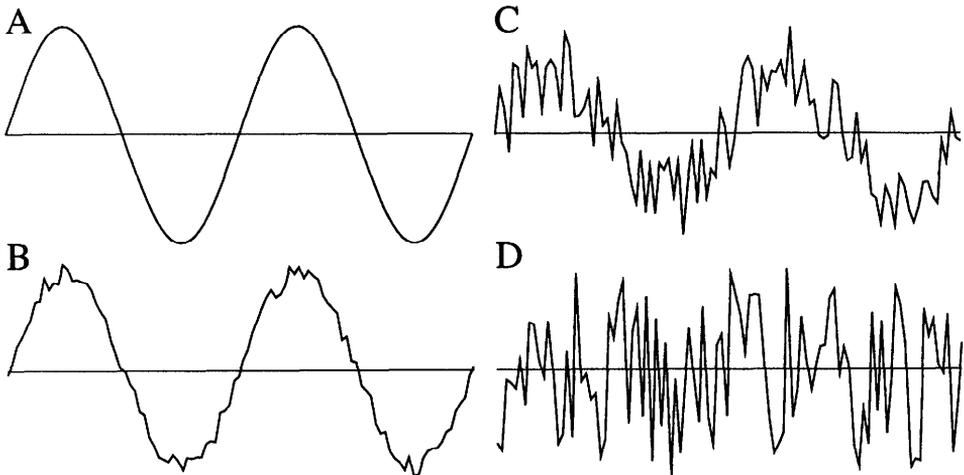


Figure 2.9: Deterministic signal (simple sine) with gradually increasing additive noise. In (A) the deterministic signal is much stronger than the noise, while in (D) the opposite is the case.

When measuring in, we usually talk about the signal as being *above* the noise by SNR(dB).

Not all the signals we encounter are stochastic due solely to contamination by additive noise. Some signals, for example speech, are inherently stochastic. Were we to pronounce a single vowel for an extended period of time the acoustic signal would be roughly deterministic; but true speech is random because of its changing content. Speech is also stochastic for another reason. Unvoiced sounds such as **s** and **f** are made by constricting air passages at the teeth and lips and are close to being pure noise. The **h** sound starts as noise produced in the throat, but is subsequently filtered by the mouth cavity; it is therefore partially random and partially deterministic.

Deterministic signals can be *periodic*, meaning that they *exactly* repeat themselves after a time known as the *period*. The falling exponential is not periodic, while the analog sine $A \sin(2\pi ft)$, as we discussed above, is periodic with period $T = \frac{1}{f}$. The electric voltage supplied to our houses and the acoustic pressure waves from a flute are both nearly perfect sinusoids and hence periodic. The frequency of the AC supplied by the electric company is 60 Hz (sixty cycles per second) in the United States, and 50 Hz (fifty cycles per second) in Europe; the periods are thus $16\frac{2}{3}$ and 20 milliseconds respectively. The transverse flutes used in orchestral music can produce frequencies from middle C (524 Hz) to about three and a half octaves, or over ten times, higher!

While the analog sinusoid is always periodic the digital counterpart is not. Consider an analog signal with a period of 2 seconds. If we create a digital sinusoid by ‘sampling’ it 10 times per second, the digital signal will be periodic with digital period 20. However, if we sample at 10.5 times per second, after 2 seconds we are a half-second out of phase; only after four seconds, (i.e., 21 samples) does the digital signal coincide with its previous values. Were we to sample at some other rate it would take even longer for the digital version to precisely duplicate itself; and if ratio of the period to the sampling interval is not rational this precise duplication will never occur.

Stochastic signals may be stationary, which means that their probabilistic description does not change with time. This implies that all the signal’s statistics, such as the mean and variance, are constant. If a stochastic signal gets stronger or weaker or somehow noisier with time, it is not stationary. For example, speech is a stochastic signal that is highly nonstationary; indeed it is by changing the statistics that we convey information. However, over short enough time intervals, say 30 milliseconds, speech seems stationary because we can’t move our mouth and tongue this fast.

A signal, analog or digital, can be of infinite or finite time duration. We required that signals be *defined* for all times $-\infty < t < \infty$ or $n = -\infty, \infty$, but not that they be nonzero for all times. Real physical signals are of finite energy, and hence are often zero for times much before or after their peak.

In like fashion, signals, analog or digital, can be of infinite or finite bandwidth. According to our original definition an analog signal should be finite bandwidth, but noise and signals with discontinuities are *full spectrum*. The interpretation of this concept for digital signals must be postponed until after clarification of the *sampling theorem*, in the Section 2.8.

EXERCISES

- 2.3.1 Look closely at the graphs of the digital sinusoid $s_n = \sin(\omega n)$ that you prepared in exercise 2.2.2. When is the digital sinusoid periodic? Under what conditions is the period the same as that of the analog sinusoid? Verify the statement in the text regarding nonperiodic digital sinusoids.
- 2.3.2 The purpose of this exercise is to examine the periodicity of the sum of two analog sines. For example, the sum of a sine of period 4 seconds and one of period 6 seconds is periodic with period 12 seconds. This is due to the first sine completing three full periods while the second completes two full periods in 12 seconds. Give an example of a sum that is *not* periodic. Give a general rule for the periodicity. What *can* be said about cases when the sum is not exactly periodic?
- 2.3.3 Plot analog signals composed of the sum of two sinusoids with identical amplitudes and frequencies f_1 and f_2 . Note that when the frequencies are close the resultant seems to have two periods, one short and one long. What are the frequencies corresponding to these periods? Prove your assertion using the trigonometric identities.

2.4 Signal Arithmetic

Some of the requirements in our definition of *signal* were constraints on signal values $s(t)$ or s_n , while some dealt with the signal as a whole. For example, finite valuedness is a constraint on every signal value separately, while finite energy and finite bandwidth requirements mix all the signal values together into one inequality. However, even the former type of requirement is most concisely viewed as a single requirement on the signal s , rather than an infinite number of requirements on the values.

This is one of the economies of notation that make it advantageous to define signals in the first place. This is similar to what is done when one defines complex numbers or n -dimensional vectors (n -vectors); in one concise equation one represents two or even n equations. With a similar motivation of economy we define arithmetic operations on signals, thus enabling us to write single equations rather than a (possibly nondenumerable) infinite number! Hence in some ways signals are just like n -vectors of infinite dimension.

First let us define the multiplication of a signal by a real number

$$\begin{array}{l} y = ax \\ \text{means} \\ y(t) = ax(t) \quad \forall t \end{array} \quad \mathbf{A} \left| \mathbf{D} \quad \begin{array}{l} y = ax \\ \text{means} \\ y_n = ax_n \quad \forall n \end{array} \quad (2.14)$$

that is, we individually multiply every signal value by the real number. It might seem overly trivial even to define this operation, but it really is important to do so. A signal is not merely a large collection of values, it is an entity in its own right. Think of a vector in three-dimensional space (a 3-vector). Of course it is composed of three real numbers and accordingly doubling its size can be accomplished by multiplying each of these numbers by two; yet the effect is that of creating a new 3-vector whose direction is the same as the original vector but whose length is doubled. We can visualize this operation as stretching the 3-vector along its own direction, without thinking of the individual components. In a similar fashion amplification of the signal should be visualized as a transformation of the signal as a whole, even though we may accomplish this by multiplying each signal value separately.

We already know that multiplication of a signal by a real number can represent an amplification or an attenuation. It can also perform an *inversion*

$$\begin{array}{l} y = -x \\ \text{means} \\ y(t) = -x(t) \quad \forall t \end{array} \quad \mathbf{A} \left| \mathbf{D} \quad \begin{array}{l} y = -x \\ \text{means} \\ y_n = -x_n \quad \forall n \end{array} \quad (2.15)$$

if we take the real number to be $a = -1$. Here the minus sign is an ‘operator’, transforming a signal into another, related, signal. The inverted signal has the same energy and bandwidth as the original, and we shall see later on has the same *power spectrum*. Nevertheless, every time the original signal increases, the inverted one decreases; when the signal attains its maximum, the inverted signal attains its minimum.

There is another way to make a signal of the same energy and power spectrum as the original, but somehow backwards. We can *reverse* a signal

using the operator Rev

$$\begin{array}{l}
 y = \text{Rev } x \\
 \text{means} \\
 y(t) = x(-t) \quad \forall t
 \end{array}
 \quad \mathbf{A} \left| \mathbf{D} \right.
 \begin{array}{l}
 y = \text{Rev } x \\
 \text{means} \\
 y_n = x_{-n} \quad \forall n
 \end{array}
 \quad (2.16)$$

which makes it run backwards in time. If you whistle a constant note it will sound the same when reversed, but if you whistle with ascending pitch the reversed signal will have descending pitch. This operation has no counterpart for n-vectors.

Frequently we will need to add two signals,

$$\begin{array}{l}
 z = x + y \\
 \text{means} \\
 z(t) = x(t) + y(t) \quad \forall t
 \end{array}
 \quad \mathbf{A} \left| \mathbf{D} \right.
 \begin{array}{l}
 z = x + y \\
 \text{means} \\
 z_n = x_n + y_n \quad \forall n
 \end{array}
 \quad (2.17)$$

one simply adds the values. This is the familiar addition of two n-vectors, and is similar to the addition of complex numbers as well. Signal addition is commutative ($x + y = y + x$) and associative ($x + (y + z) = (x + y) + z$) and adding a signal to its inversion yields the zero signal. Hence signals, like real numbers, complex numbers, and n-vectors, obey all the normal rules of arithmetic.

We will also need to multiply two signals, and you have probably already guessed that

$$\begin{array}{l}
 z = x y \\
 \text{means} \\
 z(t) = x(t) y(t) \quad \forall t
 \end{array}
 \quad \mathbf{A} \left| \mathbf{D} \right.
 \begin{array}{l}
 z = x y \\
 \text{means} \\
 z_n = x_n y_n \quad \forall n
 \end{array}
 \quad (2.18)$$

one simply multiplies *value by value*. Multiplication of a signal by a number is consistent with this definition of multiplication—just think of the number as a constant signal. However, this multiplication is different from multiplication of 3-vectors or complex numbers. The usual ‘dot product’ multiplication of two 3-vectors yields a scalar and not a 3-vector. There is a *cross* or *vector product* kind of multiplication that yields a vector, but it doesn’t generalize to n-vectors and it isn’t even commutative. Multiplication of complex numbers yields a complex number, but there

$$z = x y \quad \text{does not mean} \quad \Re z = \Re x \Re y \quad \text{and} \quad \Im z = \Im x \Im y$$

which is quite different from *value by value* multiplication of signals.

Although value by value multiplication of signals can be very useful, for instance in ‘mixing’ of signals (see Section 8.5), there is another type

of multiplication, known as *dot product*, that is more important yet. This product is analogous to the usual scalar product of n-vectors, and it yields a real number that depends on the entire signal.

$$\begin{array}{l}
 r = x \cdot y \\
 \text{means} \\
 r = \int_{-\infty}^{\infty} x(t)y(t)dt
 \end{array}
 \quad \mathbf{A} \left| \mathbf{D} \right.
 \begin{array}{l}
 r = x \cdot y \\
 \text{means} \\
 r = \sum_{n=-\infty}^{\infty} x_n y_n
 \end{array}
 \quad (2.19)$$

This is the proper definition for *real* signals; although it can be extended for complex signals. The energy of a signal is the dot product of the signal with itself, while the dot product of two different signals measures their similarity (see Chapter 9). Signals for which the dot product vanishes are said to be *orthogonal*, while those for which it is large are said to be strongly correlated.

For digital signals there is another operator known as the *time advance operator* z ,

$$y = z x \quad \text{means} \quad y_n = x_{n+1} \quad \forall n \quad (2.20)$$

which would certainly be meaningless for vectors in space. What meaning could there possibly be for an operator that transforms the x coordinate of a vector into the y coordinate? However, signals are not static vectors; they are dynamic entities. The time variable is not a dummy variable or index; it is physical time. We can always renumber the axes of a vector, thus scrambling the order of elements, and still understand that the same physical vector is described. For signals such an action is unthinkable. This is the reason that $\text{Rev}(x)$ had no vector counterpart. This is the reason that our original definition of *signal* emphasized that the independent variable or index was *time*.

You can think of z as the ‘just wait a little while and see what happens’ operator. For digital signals the natural amount of time to wait is one unit, from n to $n + 1$. If we wish to peek further forward in time, we can do so. For example, we can jump forward two units of time by first advancing one unit and then one more

$$y = zz x = z^2 x \quad \text{means} \quad y_n = x_{n+2} \quad \forall n$$

and so on.

We may also wish to go backwards in time. This doesn’t require us to invent a time machine, it just means that we wish to recall the value a signal had a moment ago. A little reflection leads us to define the *time delay operator* z^{-1}

$$y = z^{-1} x \quad \text{means} \quad y_n = x_{n-1} \quad \forall n \quad (2.21)$$

so that $z z^{-1} x = z^{-1} z x = x$. The operator z^{-1} will turn out to be even more useful than z , since it is usually easier to remember what just happened than to predict what is about to occur. The standard method for implementing the digital delay of L units of time is through a FIFO buffer of length L . A signal value that enters the FIFO at time n exits at time $n + L$, and so the output of the FIFO is delayed exactly L time units with respect to its input. When used in this fashion the FIFO is called a *delay line*.

We can make these operators more concrete with a simple example. In exercise 2.1.1.13 we introduced a family of recursively defined signals, often called the *logistics signals*

$$x_{n+1} = \lambda x_n(1 - x_n) \quad (2.22)$$

where the x_n are all in the range $0 \leq x_n \leq 1$. In order to enforce this last restriction we must restrict λ to be in the range $0 \leq \lambda \leq 4$. A particular signal in this family is determined by giving x_0 and λ . It is most instructive to generate and plot values for various x_0 and λ , and the reader will be requested to do so as an exercise. In this case the operation of the time advance operator can be simply specified

$$zx = \lambda x(1 - x)$$

which should be understood as an equation in signals. This stands for an infinite number of equations of the form (2.22), one for each n . However, we needn't return to these equations to understand it. We start with $1 - x$, which really means $1 + (-x)$. $(-x)$ is the inversion of the signal x ; we add to it the signal 1 that is the constant signal whose value is 1 for all times. Addition between signals is value by value of course. Next we multiply this signal by the original signal, using signal multiplication, value by value. Finally we multiply this resulting signal by a real number λ . So for this special case, the time advance operator can be specified in terms of simple signal arithmetic.

Operators can be combined to create new operators. The finite difference operator Δ is defined as

$$\Delta \equiv 1 - z^{-1} \quad (2.23)$$

that is, for any digital signal s , the following holds for all time n .

$$\Delta s_n = s_n - s_{n-1}$$

The finite difference operator for digital signals is vaguely similar to the differentiation operator for continuous signals. Common characteristics include linearity and the fact that they are identically zero only for a constant. Δ is a

linear operator since for any two signals x and y , $\Delta(x+y) = \Delta x + \Delta y$ and for any number c and signal x , $\Delta cx = c\Delta x$. $\Delta s = 0$ (the zero signal) if and only if the signal is constant. In other ways finite differences are similar to, but not identical to derivatives. For example, $\Delta(xy) = x\Delta y + \Delta x z^{-1}y$. In some things finite differences are completely different, e.g., $\Delta\alpha^n = \alpha^n(1 - \alpha^{-1})$.

This last example leads us to an important property of the time delay operator. For the exponential signal $s_n = e^{\Lambda n}$ it is easy to see that

$$s_{n-1} = e^{\Lambda(n-1)} = e^{-\Lambda}e^{\Lambda n} = e^{-\Lambda}s_n$$

so that

$$z^{-1}s = e^{-\Lambda}s$$

i.e., the operation of time delay on the exponential signal is equivalent to multiplication of the signal by a number. In linear algebra when the effect of an operator on a vector is to multiply it by a scalar, we call that vector an ‘eigenvector’ of the operator. Similarly we can say that the exponential signal is an *eigensignal* of the time delay operator, with eigenvalue $e^{-\Lambda}$.

The fact that the exponential is an eigensignal of the time delay operator will turn out to be very useful. It would have been even nicer were the sinusoid to have been an eigensignal of time delay, but alas equation (A.23) tells us that

$$s_{n-1} = \sin(\omega(n-1)) = \sin(\omega n)\cos(\omega) - \cos(\omega n)\sin(\omega)$$

which mixes in phase-shifted versions of the original signal. The sinusoid *is* the eigensignal of a more complex operator, one that contains two time delays; this derives from the fact that sinusoids obey second-order differential equations rather than first-order ones like the exponential. Nonetheless, there is a trick that saves the day, one that we have mentioned before. We simply work with complex exponentials, which *are* eigensignals of time delay, remembering at the end to take the real part. This tactic is perhaps the main reason for the use of complex signals in DSP.

EXERCISES

- 2.4.1 Show that the exponential signal $s_n = Ae^{\Lambda n}$ is an eigensignal of the time *advance* operator. What is its eigenvalue? The real sinusoid $s_n = A\sin(\omega n + \phi)$ is the eigensignal of an operator that contains z^{-1} and z^{-2} . Can you find this operator?
- 2.4.2 What is the effect of the time advance operator on the unit impulse? Express the general SUI $\delta_{n,m}$ in terms of $\delta_{n,0}$ and the time delay operator.

- 2.4.3 Compare the energy of a time delayed, advanced, or reversed signal with that of the original signal. What is the energy of $y = ax$ in terms of that of x ? What can be said about the energy of the sum of two signals? For example, consider summing two sinusoids of the same frequency but different amplitudes and phases. What about two sinusoids of different frequencies? Why is there a difference between these two cases?
- 2.4.4 Plot the logistics signal of equation (2.22) using several different x_0 for each λ . Try $\lambda = 0.75$ and various x_0 —what happens after a while? Next try $\lambda = 1.5, 2.0$, and 2.75 . How is the long time behavior different? Can you predict the behavior as a function of λ ? Are there any starting points where the previous behavior is still observed? Next try $\lambda = 3.2, 3.5, 3.55, 3.5675$, and 3.75 . What is the asymptotic behavior (for almost all x_0)?
- 2.4.5 Using the program from the previous exercise try $\lambda = 3.826, 3.625$ and 3.7373 . What is the asymptotic behavior? Try $\lambda = 4$. How is this different?
- 2.4.6 Canons are musical compositions composed of several related *voices* heard together. The ‘canonical’ relations require the voices to repeat the theme of the first voice:
- time offset:** after a time delay,
 - key shift:** in a different key,
 - diminution:** at twice normal speed,
 - augmentation:** at half normal speed,
 - inversion:** with high and low tones interchanged,
 - crab order:** time reversed,
- or with combinations of these. Describe the signal processing operators that transform the basic theme into the various voices. In order for the resulting canon to sound pleasing, at (almost) every instant of time the voices must be harmonically related. Can you write a program that composes canons?
- 2.4.7 In the text we discussed the usefulness of considering a signal as a single entity. This exercise deals with the usefulness of considering a signal as a collection of values. A *streaming signal* is a digital signal that is made available as time progresses. When the signal is not being streamed one must wait for the signal to be completely prepared and placed into a file before processing. Explain the usefulness of streaming digital audio. In certain computer languages a *stream* is defined to be a sequentially accessed file. Compare this use of ‘stream’ with the previous one.

2.5 The Vector Space of All Possible Signals

In Section 2.2 we presented the simplest of signals; in this section we are going to introduce you to all the rest. Of course there are an infinite number of different signals, but that doesn't mean that it will take a long time to introduce them all. How can this be? Well, there are an infinite number of points in the plane, but we can concisely describe every one using just two real numbers, the x and y coordinates. There are an infinite number of places on earth, but all can be located using longitude and latitude. Similarly there are an infinite number of different colors, but three numbers suffice to describe them all; for example, in the RGB system we give red, green, and blue components. All events that have already taken place or will ever take place in the entire universe can be located using just four numbers (three spatial coordinates and the time). These concise descriptions are made possible by identifying *basis elements*, and describing all others as weighted sums of these. When we do so we have introduced a *vector space* (see Appendix A.14). The points in the plane and in space are well known to be two-dimensional and three-dimensional vector spaces, respectively.

In the case of places on earth, it is conventional to start at the point where the equator meets the prime meridian, and describe how to reach any point by traveling first north and then east. However, we could just as well travel west first and then south, or northeast and then southwest. The choice of basic directions is arbitrary, as long as the second is not the same as the first or its reverse. Similarly the choice of x and y directions in the plane is arbitrary; instead of RGB we can use CMY (cyan, magenta, and yellow), or HSV (hue, saturation, and value); and it is up to us to choose the directions in space to arrive at any point in the universe (although the direction in time is *not* arbitrary).

Can all possible signals be described in terms of some set of basic signals? We will now convince you that the answer is affirmative by introducing the vector space of signals. It might seem strange to you that signals form a vector space; they don't seem to be magnitudes and directions like the vectors you may be used to. However, the colors also form a vector space, and they aren't obviously magnitudes and directions either. The proper way to dispel our skepticism is to verify that signals obey the basic axioms of vector spaces (presented in Appendix A.14). We will now show that not only do signals (both the analog and digital types) form a vector space, but this space has an inner product and norm as well! The fact that signals form a vector space gives them algebraic structure that will enable us to efficiently describe them.

Addition: Signal addition $s = s_1 + s_2$ according to equation (2.17),

Zero: The constant signal $s_n = 0$ for all times n ,

Inverse: The inversion $-s$ according to equation (2.15),

Multiplication: Multiplication by a real number as in equation (2.14),

Inner Product: The dot product of equation (2.19),

Norm: The energy as defined in equation (2.1),

Metric: The energy of the difference signal obeys all the requirements.

Since signals form a vector space, the theorems of linear algebra guarantee that there is a basis $\{v_k\}$, i.e., a set of signals in terms of which *any* signal s can be expanded.

$$s = \sum_k c_k v_k \quad (2.24)$$

The use of the summation sigma assumes that there are a finite or denumerable number of basis signals; when a nondenumerable infinity of basis signals is required the sum must be replaced by integration.

$$s = \int c(k)v(k) dk \quad (2.25)$$

From linear algebra we can show that every vector space has a basis, but in general this basis is not unique. For example, in two-dimensional space we have the natural basis of unit vectors along the horizontal ‘ x ’ axis and vertical ‘ y ’ axis; but we could have easily chosen any two perpendicular directions. In fact we can use any two nonparallel vectors, although orthonormal vectors have advantages (equation (A.85)). Similarly, for the vector space of signals there is a lot of flexibility in the choice of basis; the most common choices are based on signals we have already met, namely the SUIs and the sinusoids. When we represent a signal by expanding it in the basis of SUIs we say that the signal is in the *time domain*; when we use the basis of sinusoids is used we say that the signal is in the *frequency domain*.

We are not yet ready to prove that the sinusoids are a basis; this will be shown in Chapters 3 and 4. In this section we demonstrate that the SUIs are a basis, i.e., that arbitrary signals can be uniquely constructed from SUIs. We start with an example, depicted in Figure 2.10, of a digital signal that is nonzero only between times $n = 0$ and $n = 8$. We build up this signal by first taking the unit impulse $\delta_{n,0}$, multiplying it by the first signal value s_0 , thereby obtaining a signal that conforms with the desired signal at time

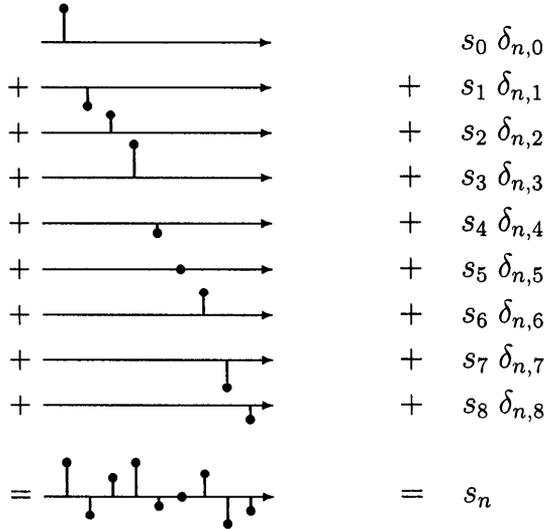


Figure 2.10: Comb-dot graph depicting building up a digital signal from shifted unit impulses.

$n = 0$ but which is zero elsewhere. Next we take the shifted unit impulse $\delta_{n,1}$, which is nonzero only for $n = 1$, and multiply it by s_1 , thus obtaining a signal that agrees with s_n for $n = 1$ but is otherwise zero. Adding together these two signals we obtain a signal that is identical to the desired signal both at time $n = 0$ and at time $n = 1$ but otherwise zero. We proceed in a similar fashion to build up a signal that is identical to the desired signal for all times.

In a similar fashion we can expand *any* digital signal in terms of SUIs

$$s_n = \sum_{m=-\infty}^{\infty} s_m \delta_{n,m} \quad (2.26)$$

thus proving that these signals span the entire space. Now, it is obvious that no two SUIs overlap, and so the SUIs are orthogonal and linearly independent (no $\delta_{n,m}$ can be expanded in terms of others). Therefore the SUIs are a linearly independent set that spans the entire space, and so they are a basis.

Hence we see that the SUIs form a basis of the vector space of digital signals. Since there are (denumerably) infinite signals in the basis, we see that the vector space of signals is of infinite dimension. Similar statements are true for analog signals as well. In Figure 2.11 we demonstrate approximating a function using shifted unit width analog impulses. We leave it for the reader to complete the argument to show that any analog signal can be

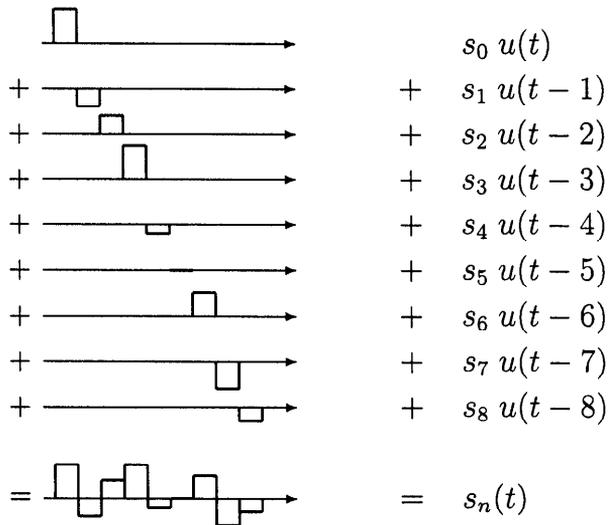


Figure 2.11: Building up an analog signal from shifted unit width impulses.

expanded in terms of shifted Dirac deltas. Dirac deltas are consequently a basis of a vector space of (nonnumerably) infinite dimension. The deltas (whether Kronecker or Dirac) form a basis that induces the time domain representation of the signal.

EXERCISES

2.5.1 Show that the triangle inequality is obeyed for signals.

$$\sum (s_{1_n} - s_{3_n})^2 \geq (\sum s_{1_n} - s_{2_n})^2 + (\sum s_{2_n} - s_{3_n})^2$$

2.5.2 Show that the set of digital signals of finite time duration is a finite dimension vector space.

2.5.3 Express a general digital signal x_n as a sum involving only the impulse at time zero and time delay operators.

2.5.4 Let's try to approximate the 3-vector $\underline{v} = (v_x, v_y, v_z)$ by a vector parallel to the x axis $\alpha_x \hat{x}$. The best such approximation requires that the error vector $\underline{\epsilon} = \underline{v} - \alpha_x \hat{x}$ be of minimal squared length. Show that this criterion leads to $\bar{\alpha}_x = v_x$ and that the error vector lies entirely in the y - z plane. Similarly, show that best approximation of \underline{v} by a vector in the x - y plane $\alpha_x \hat{x} + \alpha_y \hat{y}$, requires $\alpha_x = v_x$ and $\alpha_y = v_y$, and for the error vector must be parallel to the z axis. When can the error become zero?

- 2.5.5 The previous exercise leads us to define the coefficients v_i as those real numbers that minimize the approximation error. Use this same approach to find the expansion of a given signal $s(t)$ in terms of a set of normalized signals $v_k(t)$, by requiring the error signal to be of minimal energy. Show that this approach demystifies the use of equation (2.19) as the dot product for signals.
- 2.5.6 Show how to expand analog signals in terms of shifted Dirac delta functions, by starting with Figure 2.11 and sending the impulse width to zero.
- 2.5.7 Explain why the set of all analog signals forms a vector space. What new features are there? What is the dimensionality of this vector space? In what sense are there more analog signals than digital ones?
- 2.5.8 Show that the set of all analog periodic signals with the same period is a vector space. Is it denumerably or nondenumerably infinite in dimension?

2.6 *Time and Frequency Domains*

According to our definition a signal is a function of a signal variable, or a singly-indexed sequence. Doesn't that mean that digital signal processing is some subset of mathematics, similar to analysis (calculus)?

Technically *yes*, of course, but in a deeper sense *not at all*. The first requirement for a signal was for it to be a physical quantity; a requirement that imparts a special flavor to signal processing, quite distinct from the seasonings with which mathematical treatments of analysis are spiced.

The differential calculus was originally invented to help in the abstract mathematical treatment of the kinematics of ideal bodies. As such, the emphasis is on derivatives and the basic functions used are polynomials. Consider the kinematical quantity $s = s_0 + v_0t + \frac{1}{2}at^2$ —this function is not a physically plausible signal as it stands, since although continuous, for large times it diverges! Physically realizable functions should remain bounded for all times, which rules out all polynomials except constants.

The fundamental law of differential calculus states that any function (well not *any* function, but we won't worry about that now) can be described in the following way. First pick some time of interest, which we will call t_0 . Find the value of the function at that point, $f(t_0)$. Close enough to t_0 the function is always approximately $f(t_0)$ due to continuity constraints. To go a little further away from t_0 we need the first derivative. The first derivative describes what the function looks like close enough to t_0 since all well-behaved functions are approximately linear over a small enough interval

$f(t) \approx f(t_0) + \left. \frac{df}{dt} \right|_{t_0} (t - t_0)$. If you want to know what the function does even further away, find the second derivative evaluated at t_0 , and then the third derivative, etc. Higher and higher derivatives allow one to stray further and further from the original point in time. Knowing all derivatives at any one point in time is equivalent to knowing the function's values at all times. This law is called *Taylor's Theorem* and is the very fabric of the classical analysis way of looking at functions. It approximates functions using polynomials as the basis for the vector space of functions.

The fundamental law of signal processing proclaims a different way of representing signals. 'Real-world' signals have finite energy and occupy some finite bandwidth. Hence polynomials are not a natural basis for describing them. The signal processing approximation is global rather than local, i.e., for any finite order is about as good (or bad) simultaneously for all times $-\infty < t < +\infty$. Rather than using derivatives and polynomials, the signal processing way of looking at the world emphasizes *spectrum* and its basic signals are sinusoids. The signal processing law (the *Fourier transform*) states that all signals can be approximated by summing together basic sinusoids.

Because of this unique way of representing signals, signal processing tends to be quite schizophrenic. One has to continuously jump back and forth between the *time domain* representation, which gives the value of the signal for all times, and the *frequency domain* representation, where the harmonic content of the signal at every frequency is given.

Spectrum is simply a shorter way of saying 'frequency domain representation', and the idea is probably not new to you. You surely realize that the operation of a prism on white light consists of its decomposition into different frequencies (colors). You certainly have tuned in a station on the radio by changing the center frequency being demodulated. You may even have an audio system with a graphic equalizer enables amplifying certain component acoustic frequencies more than others.

The spectrum of a signal that consists of a pure sine wave has a single line at the frequency of this sine. The sum of two sines corresponds to two lines in the frequency domain. If the sum is weighted the relative heights of these lines will reflect this. In general, any signal that can be constructed by weighted combination of a finite number of sines will have a discrete spectrum with lines corresponding to all the frequencies and weights.

Not all signals have spectra comprised of discrete lines. For example, the analog unit width impulse has a sinc-shaped spectrum, where the sinc function

$$\text{sinc}(f) \equiv \frac{\sin(f)}{f}$$

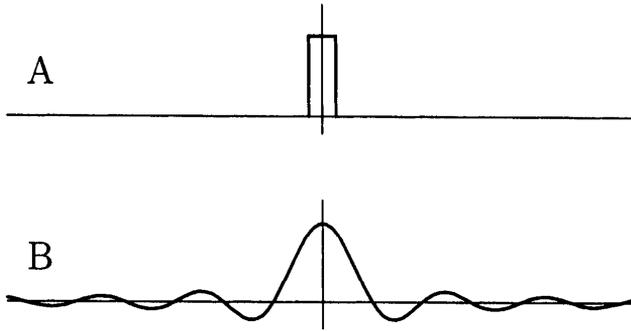


Figure 2.12: The unit width analog impulse and its spectrum. In (A) we depict the unit width impulse in the time domain, and in (B) its (sinc-function) frequency domain representation. The latter is the raw spectrum including negative frequencies.

(see Figure 2.12). The meaning of negative spectral values and negative frequencies will become clear later on. The spectrum has a strong DC component because the impulse is nonnegative. In order to make the infinitesimally sharp corners of the impulse, an infinite range of frequencies is required. So although this spectrum decreases with increasing frequency, it never becomes zero. Its bandwidth, defined as the spectral width wherein *most* of the energy is contained, is finite.

Signal processing stresses the dual nature of signals—signals have time domain and frequency domain (spectral) characteristics. Although the signal (time domain) and its Fourier transform (frequency domain) contain exactly the same information, and indeed either can be constructed from the other, some signal processing algorithms are more natural in one domain than in the other. This dual way of looking at signals is what makes signal processing different from mathematical analysis.

EXERCISES

- 2.6.1 Experiment with plotting signals composed of several sinusoids with various frequencies and amplitudes. Can you recognize the original frequencies in the resulting waveform? What do you observe when one sinusoid is much stronger than the others? When all the frequencies are multiples of a common frequency? When the frequencies are very close together? When they are well separated? When does the signal seem unpredictable?
- 2.6.2 Taylor expand a sine wave (you can do this by hand since you only need to know the derivatives of sinusoids). Fourier expand a parabola (it will probably be easiest to use numeric Fourier transform software). What can you say about the compactness of these descriptions?

- 2.6.3 The Taylor expansion can be interpreted as the expansion of arbitrary continuous functions in a basis of polynomials. Are the functions $f_0(x) = 1$, $f_1(x) = x$, $f_2(x) = x^2$, $f_3(x) = x^3, \dots$ a basis? Are they an orthonormal basis?
- 2.6.4 Let's examine a more complex signal with a discrete line spectrum. The V.34 probe signal is composed of 21 sinusoids $\sin(2\pi ft + \phi)$ with frequencies f that are multiples of 150 Hz, and phases ϕ given in the following table.

f(Hz.)	ϕ (deg)	f(Hz.)	ϕ (deg)	f(Hz.)	ϕ (deg)
150	0	1500	0	2850	0
300	180	1650	180	3000	180
450	0	1950	0	3150	180
600	0	2100	0	3300	180
750	0	2250	180	3450	180
1050	0	2550	0	3600	0
1350	0	2700	180	3750	0

Plot a representative portion of the final signal. What is special about the phases in the table? (Hint: Try altering a few phases and replotting. Observe the maximum absolute value of the signal.)

2.7 Analog and Digital Domains

At the end of Section 2.1 we mentioned that one can go back and forth between analog and digital signals. A device that converts an analog signal into a digital one is aptly named an **Analog to Digital** converter or *A/D* (pronounced *A to D*) for short. The reverse device is obviously a **Digital to Analog** converter or *D/A* (*D to A*). You will encounter many other names, such as sampler, digitizer and codec, but we shall see that these are not entirely interchangeable. In this and the next two sections we will explain that A/D and D/A devices *can* work, leaving the details of *how* they work for the following two sections.

In explaining the function of an A/D there are two issues to be addressed, corresponding to the two axes on the graph of the analog signal in Figure 2.13. You can think of the A/D as being composed of two quantizers, the *sampler* and the *digitizer*. The sampler samples the signals at discrete times while the digitizer converts the signal values at these times to a digital representation.

Converting a continuously varying function into a discrete time sequence requires sampling the former at specific time instants. This may lead to a loss

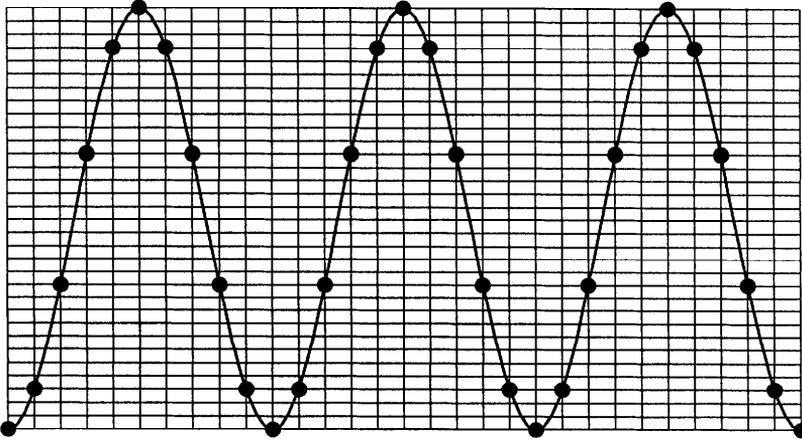


Figure 2.13: Conversion of an analog signal into a corresponding digital one involves quantizing both axes, sampling time and digitizing signal value. In the figure we see the original analog signal overlaid with the sampled time and digitized signal value grid. The resulting digital signal is depicted by the dots.

of information, since many different continuous functions can correspond to the same sampled sequence, but under certain conditions there is no such loss. The key to understanding this surprising result is the *sampling theorem*. This theorem tells us what happens when we create a discrete time signal by sampling an analog signal at a uniform rate. The sampling theorem will be discussed in the next section.

Converting the continuous real values of the analog signal into bounded digital ones requires rounding them to the nearest allowed level. This will inevitably lead to a loss of precision, which can be interpreted as adding (real-valued) noise to each value $a_n = d_n + \nu_n$, where ν_n can never exceed one half the distance to nearby quantization levels. The effect of this noise is to degrade the **Signal to Noise Ratio (SNR)** of the signal, a degradation that decreases in magnitude when the number of available levels is increased.

Digital signals obtained from analog ones are sometimes called PCM streams. Let's understand this terminology. Imagine wiping out (zeroing) the analog signal at all times that are not to be sampled. This amounts to replacing the original continuously varying signal by a sequence of pulses of varying amplitudes. We could have reached this same result in a slightly different way. We start with a train of pulses of constant amplitude. We then vary the amplitude of each incoming pulse in order to reflect the amplitude of the analog signal to be digitized. The amplitude changes of the original signal

are now reflected in the varying heights of the pulses. The process of varying some aspect of a signal in order to carry information is called *modulation*. In this case we have modulated the amplitudes of the pulse stream, and so have produced **Pulse Amplitude Modulation** (PAM). Other aspects of the pulse stream could have been varied as well, resulting in **Pulse Width Modulation** (PWM), and **Pulse Position Modulation** (PPM). We now wish to digitally record the amplitude of each pulse, which we do by giving each a *code*, e.g. the binary representation of the closest quantization level. From this code we can accurately (but not necessarily precisely) reconstruct the amplitude of the pulse, and ultimately of the original signal. The resulting sequence of numbers is called a **Pulse Code Modulation** (PCM) stream.

EXERCISES

- 2.7.1 It would seem that sampling always gives rise to some loss of information, since it always produces gaps between the sampled time instants; but sometimes we can accurately guess how to fill in these gaps. Plot a few cycles of a sinusoid by connecting a finite number of points by straight lines (linear interpolation). How many samples per cycle are required for the plot to look natural, i.e., for linear interpolation to accurately predict the missing data? How many samples per cycle are required for the maximum error to be less than 5%? Less than 1%?
- 2.7.2 Drastically reduce the number of samples per cycle in the previous exercise, but generate intermediate samples using quadratic interpolation. How many true samples per cycle are required for the predictions to be reasonably accurate?
- 2.7.3 The sampling theorem gives a more accurate method of interpolation than the linear or quadratic interpolation of the previous exercises. However, even this method breaks down at some point. At what number of samples per cycle can *no* method of interpolation work?

2.8 Sampling

We will generally sample the analog signal at a uniform rate, corresponding to a *sampling frequency* f_s . This means that we select a signal value every $t_s = \frac{1}{f_s}$ seconds. How does t_s influence the resulting digital signal? The main effects can be observed in Figures 2.14–2.17.

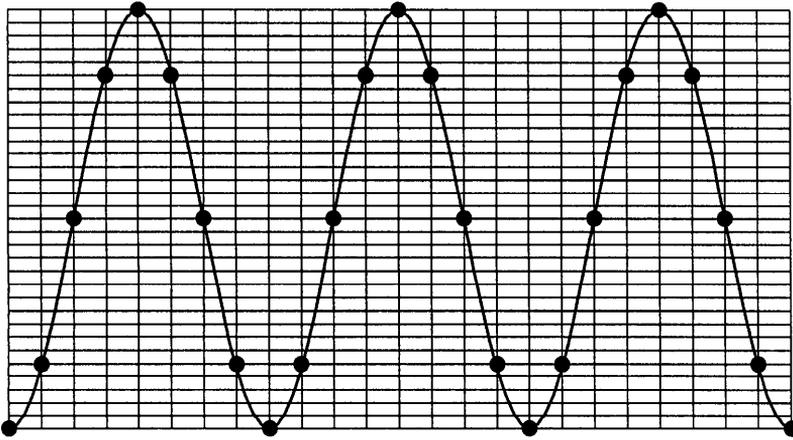


Figure 2.14: Conversion of an analog signal into the corresponding digital one with a lower sampling rate. As in the previous figure, the original analog signal has been overlaid with the sampled time and digitized signal value grid. However, the time interval between samples t_s is longer.

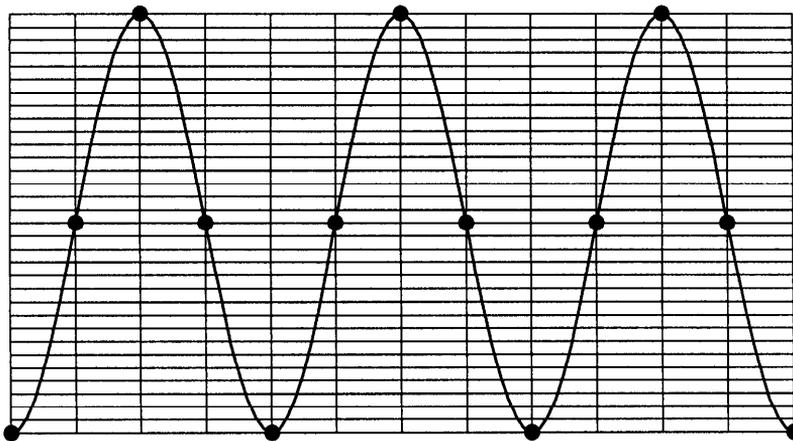


Figure 2.15: Conversion of an analog signal into the corresponding digital one with yet a lower sampling rate. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. Although there are only four samples per cycle, the original signal is still somewhat recognizable.

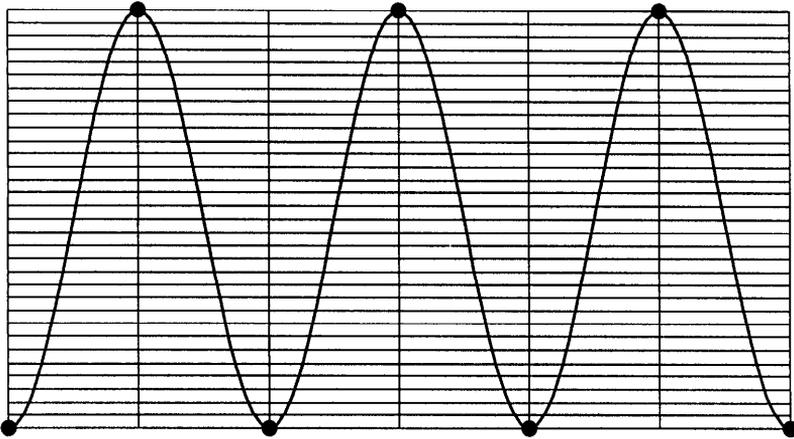


Figure 2.16: Conversion of an analog signal into a digital one at the minimal sampling rate. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. Although there are only two samples per cycle, the frequency of the original sine wave is still retrievable.

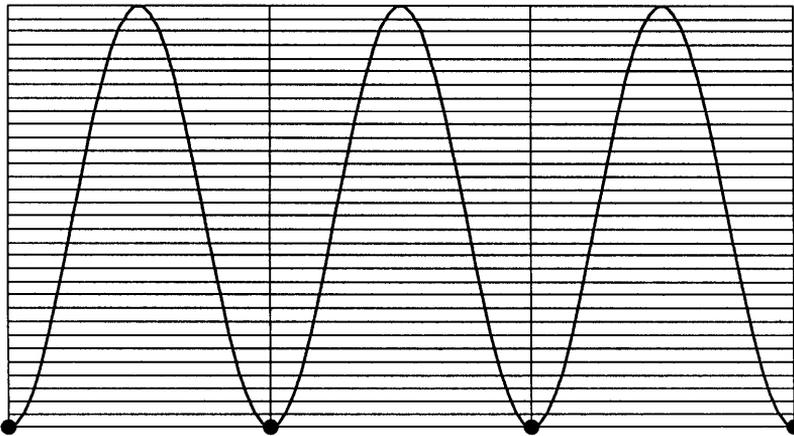


Figure 2.17: Conversion of an analog signal into a digital one at too low a sampling rate. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. With only one sample per cycle, all information is lost.

In Figures 2.14 and 2.15 the sampling rate is eight and four samples per cycle respectively, which is high enough for the detailed shape of the signal to be clearly seen (it is a simple sinusoid). At these sampling rates even simple linear interpolation (connecting the sample points with straight lines) is not a bad approximation, although peaks will usually be somewhat truncated. In Figure 2.16, with only two samples per cycle, we can no longer make out the detailed form of the signal, but the basic frequency is discernible. With only a single sample per cycle, as in Figure 2.17, even this basic frequency is lost and the signal masquerades as DC.

Have you ever watched the wagon wheels in an old *western*? When the wagon starts to move the wheels start turning as they should; but then at some speed they anomalously seem to stand still and then start to spin backwards! Then when the coach is going faster yet they straighten out for a while. What is happening? Each second of the moving picture is composed of some number (say 25) still pictures, called frames, played in rapid succession. When the wheel is rotating slowly we can follow one spoke advancing smoothly around the axle, from frame to frame. But when the wheel is rotating somewhat faster the spoke advances so far between one frame and the next that it seems to be the next spoke, only somewhat behind. This gives the impression of retrograde rotation. When the wheel rotates exactly the speed for one spoke to move to the next spoke's position, the wheel appears to stand still.

This phenomenon, whereby sampling causes one frequency to look like a different one, is called *aliasing*. The sampled pictures are consistent with different interpretations of the continuous world, the real one now going under the alias of the apparent one. Hence in this case the sampling caused a loss of information, irreversibly distorting the signal. This is a general phenomenon. Sampling causes many analog signals to be mapped into the same digital signal. This is because the digitized signal only records the values of the continuous signal at particular times $t = nt_s$; all analog signals that agree at these points in time, but differ in between them, are aliased together to the same digital signal.

Since sampling always maps many analog signals into the same digital signal, the question arises—are there conditions under which A/D does *not* cause irreparable damage? That is, is there any way to guarantee that we will be able to recover the value of the analog signal *at all times* based on the sampled signal alone? We expect the answer to be negative. Surely the analog signal can take on arbitrary values at times not corresponding to sample periods, and therefore many different analog signals correspond to the same digital one. An affirmative answer would imply a one-to-one

correspondence between analog signals obeying these conditions and the digital signals obtained by sampling them.

Surprisingly the answer *is* affirmative; but what stipulation can confound our simple logic? What restrictions can ensure that we incur no loss of information when representing a continuous function at discrete points only? What conditions on the signal will allow us to correctly guess the value of a function between two times separated by t_s where it is known? The answer is *finite bandwidth*.

Theorem: The Sampling Theorem

Assume that the analog signal $s(t)$ is sampled with a sampling frequency $f_s = 1/t_s$ producing the digital signal $s_n = s(nt_s)$.

A. If the sampling frequency is over twice that of the highest frequency component of the signal $f_s > f_{max}$, then the analog signal can be reconstructed for any desired time.

B. The reconstructed value of the analog signal at time t

$$s(t) = \sum_{n=-\infty}^{\infty} s_n \operatorname{sinc}(\pi f_s(t - nt_s)) \quad (2.27)$$

is a linear combination of the digital signal values with $\operatorname{sinc}(t) \equiv \sin(t)/t$ weighting. ■

At first sight the sampling theorem seems counterintuitive. We specify the values of a signal at certain discrete instants and claim to be able to exactly predict its value at other instants. Surely the signal should be able to oscillate arbitrarily in between sampling instants, and thus be unpredictable. The explanation of this paradox is made clear by the conditions of the sampling theorem. The bandwidth limitation restricts the possible oscillations of the analog signal between the sample instants. The signal cannot do more than smoothly interpolate between these times, for to do so would require higher frequency components than it possesses.

The minimal sampling frequency (a little more than twice the highest frequency component) is called the Nyquist frequency $f_N \equiv 2f_{max}$ in honor of Harry Nyquist, the engineer who first published the requirement in 1928. It wasn't until 1949 that mathematician Claude Shannon published a formal proof of the sampling theorem and the reconstruction formula. An inaccurate, but easy to remember, formulation of the contributions of these two men is that Nyquist specified *when* an A/D can work, and Shannon dictated *how* a D/A should work.

To better understand the Nyquist criterion consider the simple case of a single sinusoid. Here the minimum sampling frequency is twice per cycle. One of these sample instants will usually be in the positive half-cycle and the in the negative one. It is just this observation of positive and negative half-cycles that makes the sampling theorem work. It is intuitively obvious that sampling at a lesser rate could not possibly be sufficient, since entire half cycles will be lost. Actually even sampling precisely twice per cycle is not sufficient, since sampling at precisely the zero or peaks conceals the half-cycles, which is what happened in Figure 2.17. This is why the sampling theorem requires us to sample at a strictly higher rate.

The catastrophe of Figure 2.17 is a special case of the more general phenomenon of *aliasing*. What the sampling theorem tells us is that discrete time signals with sampling rate f_s uniquely correspond to continuous time signals with frequency components less than $\frac{f_s}{2}$. Sampling any continuous time signal with higher-frequency components still provides a discrete time signal, but one that uniquely corresponds to another, simpler signal, called the *alias*. Figure 2.18 demonstrates how a high-frequency sinusoidal signal is aliased to a lower frequency one by sampling. The two signals agree at the sample points, but the simpler interpretation of these points is the lower-frequency signal.

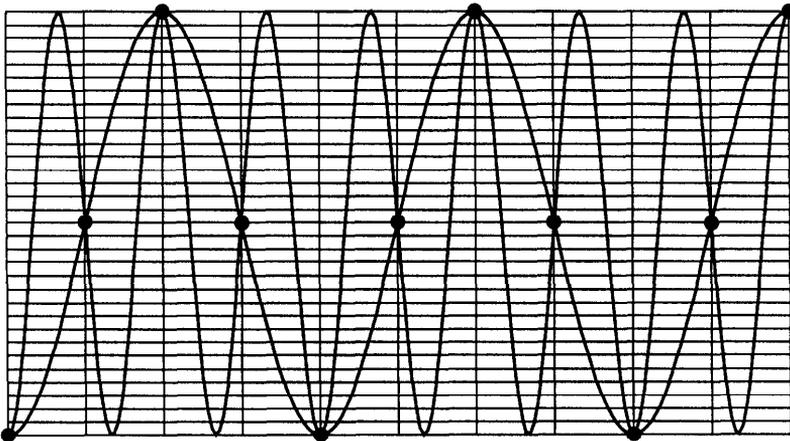


Figure 2.18: Aliasing of high-frequency analog signal into lower-frequency one. The high-frequency signal has only a sample every one and a half cycles, i.e., it corresponds to a digital frequency of $\frac{3}{4}$. The lower-frequency sinusoid is sampled at four samples per cycle, i.e., $\varphi = \frac{1}{4}$.

It is conventional to define a *digital frequency* in the following way

$$\varphi \equiv \frac{f}{f_s}$$

and the sampling theorem tells us that we must have $\varphi < \frac{1}{2}$. Consistently using this digital frequency frees us from having to think about real (analog) frequencies and aliasing. All the DSP will be exactly the same if a 2 Hz signal is sampled at 10 Hz or a 2 MHz signal is sampled at 10 MHz.

Before continuing we should mention that the sampling theorem we have been discussing is not the final word on this subject. Technically it is only the ‘low-pass sampling theorem for uniform time intervals’. If the signals of interest have small bandwidth but are centered on some high frequency, it is certainly sufficient to sample at over twice the highest frequency component, but only necessary to sample at about twice the bandwidth. This is the content of the *band-pass sampling theorem*. It is also feasible in some instances to sample nonuniformly in time, for example, at times $0, \frac{1}{2}, 2, 2\frac{1}{2}, 4, \dots$. For such cases there are ‘nonuniform sampling theorems’.

Now that we understand the first half of the sampling theorem, we are ready to study the reconstruction formula in the second half. We can rewrite equation (2.27) as

$$s(t) = \sum_{n=-\infty}^{\infty} s_n h(t - nt_s) \quad (2.28)$$

where $h(t) \equiv \text{sinc}(\pi f_s t)$ is called the *sampling kernel*. As a consequence the reconstruction operation consists of placing a sampling kernel at every sample time nt_s , weighting it by the sampled value there s_n , and adding up all the contributions (see Figure 2.19). The sine in the numerator of the sinc is zero for all sample times nt_s , and hence the sampling kernel obeys $h(nt_s) = \delta_{n,0}$. From this we immediately conclude $s(nt_s) = s_n$ as required. Consequently, the reconstruction formula guarantees consistency at sample times by allowing only the correct digital signal value to contribute there. At no other times are the sampling kernels truly zero, and the analog signal value is composed of an infinite number of contributions.

In order for the reconstruction formula to be used in practice we must somehow limit the sum in (2.28) to a finite number of contributions. Noting that the kernel $h(t)$ decays as $\frac{1}{\pi f_s t}$ we can approximate the sum by restricting the duration in time of each sample’s contribution. Specifically, if we wish to take into account only terms larger than some fraction p , we should limit each sample’s contributions to $\pm \frac{1}{\pi p}$ samples from its center. Conversely this restriction implies that each point in time to be interpolated will only receive

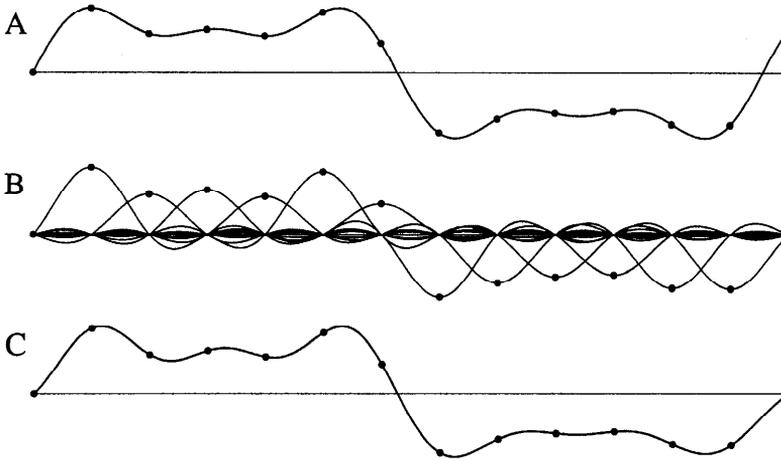


Figure 2.19: The reconstruction formula depicted graphically. In (A) we see an analog signal and the samples digitized slightly higher than twice the highest frequency component. (B) shows the sinc kernels weighted by the sample value placed at each sample time; note that at sample times all other sincs contribute zero. In (C) we sum the contributions from all kernels in the area and reconstruct the original analog signal.

a finite number of contributions (from those sample instants no further than $\frac{1}{\pi p}$ away).

Proceeding in this fashion we obtain the following algorithm:

Given: a sampled signal x_n ,
 its sampling interval t_s ,
 a desired time t , and
 a cut-off fraction p

$w \leftarrow \text{Round}(\frac{1}{\pi p})$

Initialize $i \leftarrow 0$

$n_{mid} \leftarrow \frac{t}{t_s}$

$n_{lo} \leftarrow n_{mid} - w$

$n_{hi} \leftarrow n_{mid} + w$

$x \leftarrow 0$

for $n \leftarrow n_{lo}$ to n_{hi}

$x \leftarrow x + x_n \text{sinc}(\pi \frac{t - nt_s}{t_s})$

EXERCISES

- 2.8.1 The wagon wheel introduced in the text demonstrates the principle of aliasing in a popular context. What is the observed frequency as a function of intended frequency.
- 2.8.2 Redraw Figures 2.13–2.17 with sample times at different phases of the sinusoid. Is a sine wave sampled at exactly twice per cycle (as in Figure 2.16) always recoverable?
- 2.8.3 Redraw Figures 2.13–2.17 with a noninteger number of samples per cycle. What new effects are observed? Are there any advantages to such sampling? Doesn't this contradict the sampling theorem?
- 2.8.4 Plot an analog signal composed of several sinusoids at ten times the Nyquist frequency (vastly oversampled). Overlay this plot with the plots obtained for slightly above and slightly below Nyquist. What do you observe?
- 2.8.5 Write a program for sampling rate conversion based on the algorithm for reconstruction of the analog signal at arbitrary times.

2.9 Digitization

Now we return to the issue of signal value quantization. For this problem, unfortunately, there is no panacea; there is no critical number of bits above which no information is lost. The more bits we allocate per sample the less noise we add to the signal. Decreasing the number of bits monotonically reduces the SNR.

Even more critical is the matching of the spacing of the quantization levels to the signal's dynamic range. Were the spacing set such that the signal resided entirely between two levels, the signal would effectively disappear upon digitizing. Assuming there are only a finite number of quantization levels, were the signal to vary over a much larger range than that occupied by the quantization levels, once again the digital representation would be close to meaningless. For the time being we will assume that the digitizer range is set to match the dynamic range of the signal (in practice the signal is usually amplified to match the range of the digitizer).

For the sake of our discussion we further assume that the analog signal is linearly digitized, corresponding to b bits. This means that we select the signal level $l = -(2^{b-1} - 1) \dots + 2^{b-1}$ that is closest to $s(t_n)$. How does b influence the resulting digital signal? The main effects can be observed in Figures 2.20–2.24.

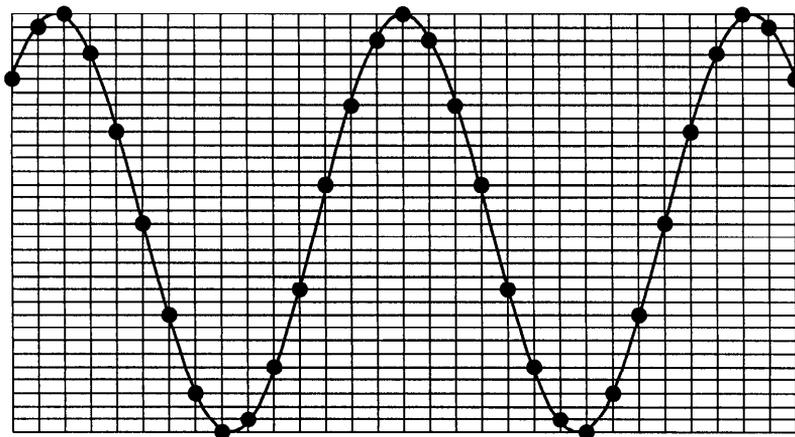


Figure 2.20: Conversion of an analog signal into a corresponding digital one involves quantizing both axes, sampling time and digitizing signal value. In the figure we see the original analog signal overlaid with the sampled time and digitized signal value grid. The resulting digital signal is depicted by the dots.

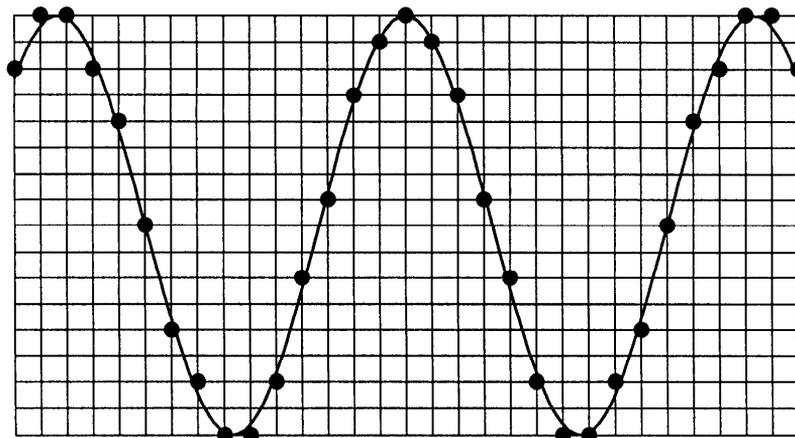


Figure 2.21: Conversion of an analog signal into the corresponding digital one with fewer digitizing levels. As in the previous figure the original analog signal has been overlaid with the sampled time and digitized signal value grid. However, here only 17 levels (about four bits) are used to represent the signal.

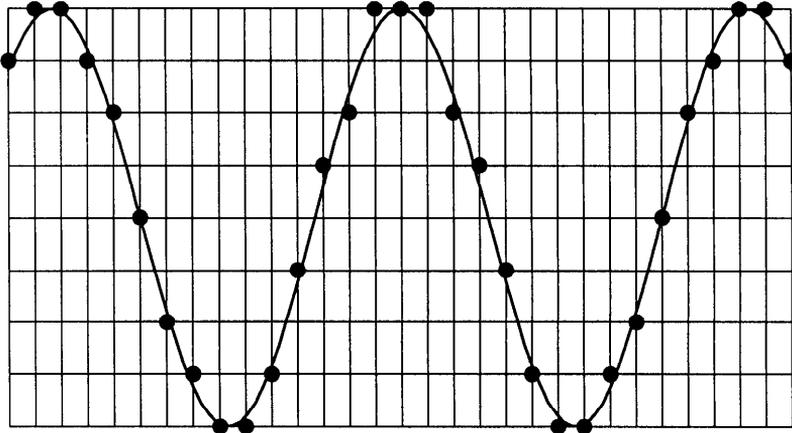


Figure 2.22: Conversion of an analog signal into the corresponding digital one with fewer digitizing levels. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. Here only nine levels (a little more than three bits) are used to represent the signal.

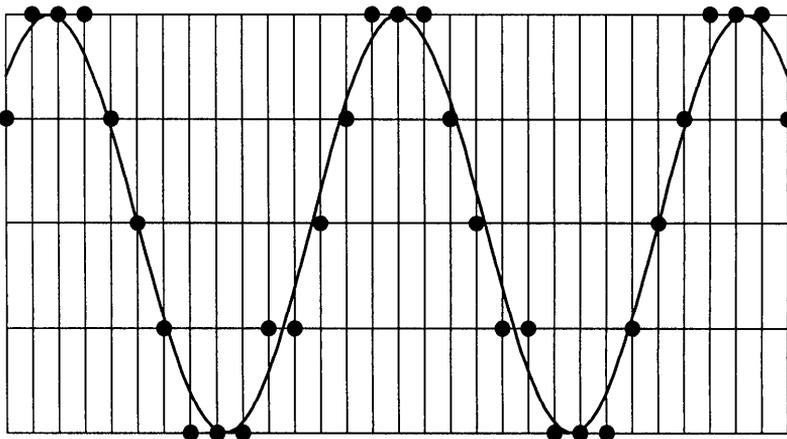


Figure 2.23: Conversion of an analog signal into the corresponding digital one with fewer digitizing levels. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. Here only five levels (about two bits) are used to represent the signal.

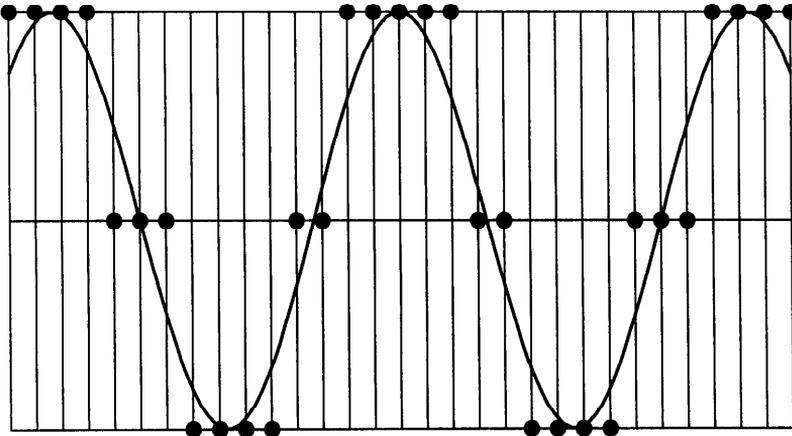


Figure 2.24: Conversion of an analog signal into the corresponding digital one with the minimum number of digitizing levels. Once again the original analog signal has been overlaid with the sampled time and digitized signal value grid. Here only three levels (one and a half bits) are used to represent the signal.

Reflect upon the discrete time signal before signal value quantization (the pulses before coding). This sequence of real numbers can be viewed as the sum of two parts

$$a_n = d_n + \nu_n \quad \text{where} \quad d_n \equiv \text{Round}(a_n)$$

and so d_n are integers and $|\nu_n| \leq \frac{1}{2}$. Assuming a_n to be within the range of our digitizer the result of coding is to replace a_n with d_n , thus introducing an error ν_n (see Figure 2.25). Were we to immediately reconvert the digital signal to an analog one with a D/A converter, we would obtain a signal similar to the original one, but with this noise added to the signal.

The proper way of quantifying the amount of quantization noise is to compare the signal energy with the noise energy and compute the SNR from equation (2.13). For a given analog signal strength, as the quantization levels become closer together, the relative amount of noise decreases. Alternatively, from a digital point of view, the quantization noise is always a constant $\pm \frac{1}{2}$ levels, while increasing the number of bits in the digital representation increases the digital signal value. Since each new bit doubles the number of levels and hence the digital signal value

$$\text{SNR(dB)} \approx 10 \left(\log_{10}(2^b)^2 - \log_{10} 1^2 \right) = 20b \log_{10} 2 \approx 6b \quad (2.29)$$

that is, each bit contributes about 6 dB to the SNR. The exact relation will be derived in exercise 2.9.2.

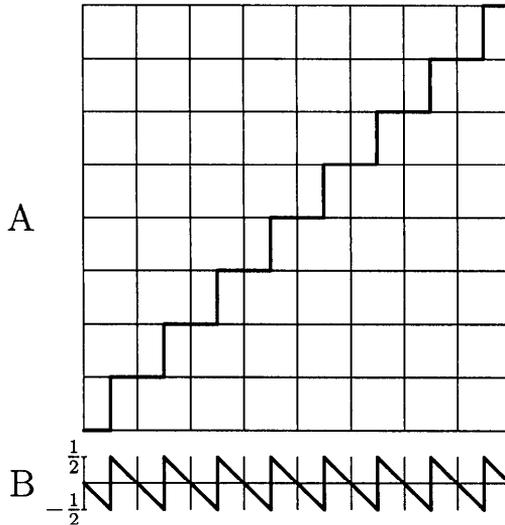


Figure 2.25: Noise created by digitizing an analog signal. In (A) we see the output of a digitizer as a function of its input. In (B) the noise is the rounding error, i.e., the output minus the input.

We have been tacitly assuming a digitizer of infinite range. In practice all digitizers have a maximum number of bits and thus a minimum and maximum level. The interval of analog signal values that are translated into valid digital values is called the *dynamic range* of the digitizer. Analog signal values outside the allowed range are clipped to the maximum or minimum permitted levels. Most digitizers have a fixed number of bits and a fixed dynamic range; in order to minimize the quantization noise the analog signal should be amplified (or attenuated) until it optimally exploits the dynamic range of the digitizer. Exceeding the dynamic range of the digitizer should be avoided as much as possible. Although moderate amounts of saturation are not usually harmful to the digitizer hardware, signal clipping is introduced. For a signal with high **Peak to Average Ratio (PAR)**, one must trade off the cost of occasional clipping with the additional quantization noise.

Signal to noise ratios only have significance when the ‘noise’ is truly random and uncorrelated with the signal. Otherwise we could divide a noiseless signal into two equal signals and claim that one is the true signal, the other noise, and the SNR is 0 dB! We have been tacitly assuming here that the quantization noise is truly noise-like and independent of the signal, although this is clearly not the case. What is the character of this ‘noise’?

Imagine continuously increasing the input to a perfect digitizer from the minimum to the maximum possible input. The output will only take

quantized values, essentially rounding each input to the closest output level. Hence the output as a function of the input will produce a graph that looks like a staircase, as in Figure 2.25.A. Accordingly the rounding error, the output minus the input, will look like a sawtooth, as in Figure 2.25.B. Thus the quantization ‘noise’ is predictable and strongly correlated with the signal, not random and uncorrelated as we tacitly assumed. This result seems contradictory—if the noise signal is predictable, then it isn’t *noise* at all. Were the error to be truly predictable, then one could always compensate for it, and digitizing would not harm the signal at all. The resolution of this paradox is simple. The noise signal is indeed correlated to the analog signal, but independent of the digitized signal. After digitizing the analog signal is unavailable, and the noise becomes, in general, unpredictable.

EXERCISES

- 2.9.1 Dither noise is an analog noise signal that can be added to the analog signal before digitizing in order to lessen perceived artifacts of round-off error. The dither must be strong enough to effectively eliminate spurious square wave signals, but weak enough not to overly damage the SNR. How much dither should be used? When is dither needed?
- 2.9.2 Refine equation (2.29) and derive $\text{SNR} = (2 \log_{10} 2b + 1.8) \text{dB}$ by exploiting the statistical uniformity of the error, and the definition of standard deviation.
- 2.9.3 Plot the round-off error as a function of time for sinusoids of amplitude 15, and frequencies 1000, 2000, 3000, 1100, 1300, 2225, and 3141.5 Hz, when sampled at 8000 samples per second and digitized to integer levels (-15, -14, ..., 0, ..., 14, 15). Does the error look noise-like?

2.10 Antialiasing and Reconstruction Filters

Recall that in Figure 1.3 there were two filters marked *antialiasing filter* and *reconstruction filter* that we avoided discussing at the time. Their purpose should now be clear. The antialiasing filter should guarantee that no frequencies over Nyquist may pass. Of course no filter is perfect, and the best we can hope for is adequate attenuation of illegal frequencies with minimal distortion of the legal ones. The reconstruction filter needs to smooth out the D/A output, which is properly defined only at the sampling instants,

and recreate the proper behavior at all times. In this section we will briefly discuss these filters.

Assume that the highest frequency of importance in the signal to be sampled is f_{max} . Strictly speaking the sampling theorem allows us to sample at any frequency above the Nyquist frequency $f_N = 2f_{max}$, but in practice we can only sample this way if there is absolutely nothing above f_{max} . If there are components of the signal (albeit unimportant ones) or other signals, or even just background noise, these will fold back onto the desired signal after sampling unless removed by the antialiasing filter. Only an *ideal* antialiasing filter, one that passes perfectly all signals of frequency less than f_{max} and blocks completely all frequencies greater than f_{max} , would be able to completely remove the undesired signals; and unfortunately, as we shall learn in Section 7.1, such an ideal filter cannot be built in practice.

Realizable antialiasing filters pass low frequencies, start attenuating at some frequency f_1 , and attenuate more and more strongly for higher and higher frequencies, until they effectively block all frequencies above some f_2 . We must be sure that the spectral areas of interest are below f_1 since above that they will become attenuated and distorted; however, we can't use $2f_1$ as our sampling frequency since aliasing will occur. Thus in order to utilize realizable filters we must sample at a frequency $2f_2$, higher than the sampling theorem strictly requires. Typically sampling frequencies between 20% and 100% higher ($1.2f_N \leq f_s \leq 2f_N$) are used. The extra spectral 'real-estate' included in the range below $\frac{f_s}{2}$ is called a *guard band*.

The D/A reconstruction filter's purpose is slightly less obvious than that of the antialiasing filter. The output of the D/A must jump to the required digital value at the sampling time, but what should it do until the next sampling time? Since we have no information about what the analog signal does, the easiest thing to do is to stay constant until the next sampling time. Doing this we obtain a piecewise constant or 'boxcar' signal that doesn't approximate the original analog signal very well. Alternatively, we might wish to linearly interpolate between sampling points, but there are two difficulties with this tactic. First, the linear interpolation, although perhaps better looking than the boxcar, is not the proper type of interpolation from the signal processing point of view. Second, and more importantly, interpolation of any kind is noncausal, that is, requires us to know the next sample value before its time. This is impossible to implement in real-time hardware. What we *can* do is create the boxcar signal, and then filter it with an analog filter to smooth the sharp transitions and eliminate unwanted frequencies.

The antialiasing and reconstruction filters may be external circuits that the designer must supply, or may be integral to the A/D and D/A devices

themselves. They may have fixed cutoff frequencies, or may be switchable, or completely programmable. Frequently DSP software must set up these filters along with initialization and setting sampling frequency of the A/D and D/A. So although we shall not mention them again, when designing, building, or programming a DSP system, don't forget your filters!

EXERCISES

- 2.10.1 Simulate aliasing by adding sinusoids with frequencies above Nyquist to properly sampled sinusoidal signals. (You can perform this experiment using analog signals or entirely on the computer.) Make the aliases much weaker than the desired signals. Plot the resulting signals.
- 2.10.2 If you have a DSP board with A/D and D/A determine how the filters are implemented. Are there filters at all or are you supposed to supply them externally? Perhaps you have a 'sigma-delta' converter that effectively has the filter built into the A/D. Is there a single compromise filter, several filters, or a programmable filter? Can you control the filters using software? Measure the antialiasing filter's response by injecting a series of sine waves of equal amplitude and increasing frequency.
- 2.10.3 What does speech sound like when the antialiasing filter is turned off? What about music?

2.11 Practical Analog to Digital Conversion

Although in this book we do not usually dwell on hardware topics, we will briefly discuss circuitry for A/D and D/A in this section. We have two reasons for doing this. First, the specifications of the analog hardware *are* of great important to the DSP software engineer. The DSP programmer understand what is meant by such terms as 'one-bit sampling' and 'effective bits' in order to properly design and debug software systems. Also, although we all love designing and coding advanced signal processing algorithms, much of the day-to-day DSP programming has to do with interfacing to the outside world, often by directly communicating with A/D and D/A devices. Such communication involves initializing, setting parameter values, checking status, and sending/receiving data from specific hardware components that the programmer must understand well. In addition, it is a fact of life that A/D

components occasionally fail, especially special-purpose fast A/D converters. The DSP software professional should know how to read the signs of a failing A/D, and how to test for deficiencies and to evaluate performance.

Perhaps the simplest A/D to start with is the so-called *flash converter*, the block diagram of which is given in Figure 2.26. The triangles marked ‘comp’ are *comparators* that output ‘one’ when the voltage applied to the *in* input is higher than that applied to the reference input *ref*, and ‘zero’ otherwise. For a b bit A/D converter we require 2^b such comparators (including the highest one to indicate an *overflow* condition). The reference inputs to the comparators must be as precise as possible, and for this reason are often derived from a single voltage source.

Every sampling time a voltage x is applied to the input of the digitizer. All the comparators whose reference voltages are less than x will fire, while those with higher references will not. This behavior reminds one of a mercury thermometer, where the line of mercury reaches from the bottom up to a line corresponding to the correct temperature, and therefore this encoding is called a *thermometer code*. The thermometer code requires 2^b bits to encode 2^b values, while standard binary encoding requires only b bits. It would accordingly be not only nonstandard but also extremely inefficient to use it directly. The function of the block marked ‘thermometer to binary decoder’ in the diagram is to convert thermometer code into standard binary. It is left as an exercise to efficiently implement this decoder.

The main drawback of the flash converter is its excessive cost when a large number of bits is desired. A straightforward implementation for 16-bit resolution would require 2^{16} reference voltages and comparators and a 2^{16} by 16 decoder! We could save about half of these, at the expense of increasing the time required to measure each voltage, by using the following tactic. As a first step we use a single comparator to determine whether the incoming voltage is above or below half-scale. If it is below half-scale, we then determine its exact value by applying it to a bank of $\frac{1}{2}2^b = 2^{b-1}$ comparators. If it is above half-scale we first shift up the reference voltages to all of these 2^{b-1} comparators by the voltage corresponding to half-scale, and only then apply the input voltage. This method amounts to separately determining the MSB, and requires only $2^{b-1} + 1$ comparators.

Why should we stop with determining the MSB separately? Once it has been determined we could easily add another step to our algorithm to determine the second most significant bit, thus reducing to $2^{b-2} + 3$ the number of comparators needed. Continuing recursively in this fashion we find that we now require only b stages, in each of which we find one bit, and only b comparators in all. Of course other compromises are possible,

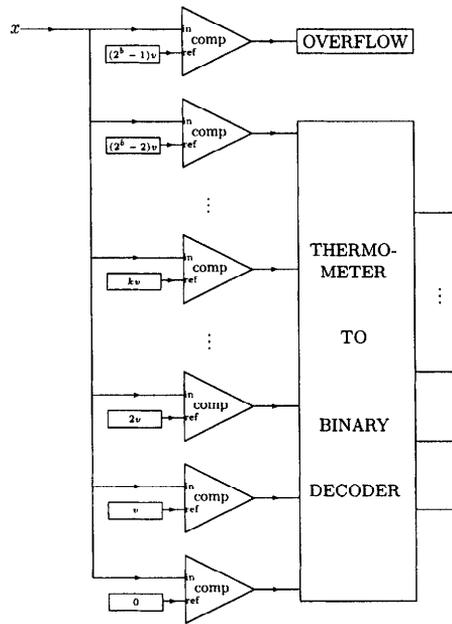


Figure 2.26: Schematic diagram of a flash converter A/D.

for example, n most significant bits can be determined by a coarse flash converter, and then the remaining $b - n$ bits by an appropriately shifted fine converter. These methods go by the name *serial-parallel* or *half-flash* converters.

In order to use such a device we would have to ensure that the input voltage remains constant during the various stages of the conversion. The time taken to measure the voltage is known as the *aperture time*. Were the voltage to fluctuate faster than the aperture time, the result would be meaningless. In order to guarantee constancy of the input for a sufficient interval a *sample and hold* circuit is used. The word 'hold' is quite descriptive of the circuit's function, that of converting the continuously varying analog signal into a piecewise constant, boxcar signal.

When a sample and hold circuit is employed, we can even reduce the number of comparators employed to one, at the expense of yet a further increase in aperture time. We simply vary the reference voltage through all the voltages in the desired range. We could discretely step the voltage through 2^b discrete levels, while at the same time incrementing a counter; the desired level is the value of this counter when the reference voltage

first passes the sample and hold voltage. Stepping through 2^b levels can be a complex and time-intensive job, and can be replaced by a continuously increasing ramp. The counter is replaced by a mechanism that measures the time until the comparator triggers. A sawtooth waveform is usually utilized in order to quickly return to the starting point. This class of A/D converters is called a *counting converter* or a *slope converter*.

High-precision counting converters are by their very nature extremely slow. *Successive-approximation* converters are faster for the same reason that half-flash are faster than flash converters. The principle is to start with a steep slope, thus quickly determining a rough approximation to the input voltage. Once the reference passes the input it is reduced one level and further increased at a slower rate. This process continues until the desired number of bits has been obtained.

Now that we understand some of the principles behind the operation of real-world A/D devices, we can discuss their performance specifications. Obviously the device chosen must be able to operate at the required sampling rate, with as many bits of accuracy as further processing requires. However, bits are not always bits. Imagine a less-than-ethical hardware engineer, whose design fails to implement the required number of bits. This engineer could simply add a few more pins to his A/D chip, not connecting them to anything in particular, and claim that they are the least significant bits of the converter. Of course they turn out to be totally uncorrelated to the input signal, but that may be claimed to be a sign of noise. Conversely, if a noisy input amplifier reduces the SNR below that given by equation (2.29) we can eliminate LSBs without losing any signal-related information. A/D specifications often talk about the number of *effective bits* as distinct from the number of output bits. Effective bits are bits that one can trust, the number of bits that are truly input-signal correlated. We can find this number by reversing the use of equation (2.29).

The number of effective bits will usually decrease with the frequency of the input signal. Let's understand why this is the case. Recall that the A/D must actually observe the signal over some finite interval, known as the aperture time, in order to determine its value. For a low-frequency signal this is not problematic since the signal is essentially constant during this entire time. However, the higher the frequency the more the signal will change during this interval, giving rise to *aperture uncertainty*. Consider a pure sine wave near where it crosses the axis. The sine wave is approximately linear in this vicinity, and its slope (derivative) is proportional to the frequency.

From these considerations it is easy to see that

$$fT_{\text{aperture}} \leq 2^{-b} \quad (2.30)$$

so that the effective bits decrease with increasing frequency.

The sigma-delta, or one-bit, digitizer is a fundamentally different kind of A/D device. Although the principles have been known for a long time, sigma-delta digitizing has become fashionable only in the last few years. This is because its implementation has only become practical (read *inexpensive*) with recent developments in VLSI practice.

With *delta-PCM* one records the differences ('delta's) between successive signal values rather than the values themselves. It is clear that given the initial value and a sequence of such differences the original signal may be recovered. Hence delta-PCM carries information equivalent to the original PCM. The desirability of this encoding is realized when the signal does not vary too rapidly from sample to sample. In this case these differences will be smaller in absolute value (and consequently require fewer bits to capture) than the signal values themselves. This principle is often exploited to compress speech, which as we shall see in Section 19.8 contains more energy at low frequencies.

When the signal does vary too much from sample to sample we will constantly overflow the number of bits we have allotted to encode the difference. To reduce the possibility of this happening we can increase the sampling rate. Each doubling of the sampling rate should reduce the absolute value of the maximum difference by a factor of two and accordingly decrease the number of bits required to encode it by one. We therefore see a trade-off between sampling frequency and bits; we can sample at Nyquist with many bits, or *oversample* with fewer bits. Considering only the number of bits produced, slower is always better; but recalling that the number of comparators required in a flash converter increases exponentially in the number of bits encoded, faster may be cheaper and more reliable. In addition there is another factor that makes an oversampled design desirable. Since we are oversampling, we can implement the antialiasing filter digitally, making it more dependable and flexible.

It would seem that we have just made our A/D more complex by requiring digital computation to be performed. However, reconstructing the original signal from its delta encoding requires digital computation in any case, and the antialiasing filter can be combined with the reconstruction. The overall computation is a summing (represented mathematically by the letter sigma) of weighted differences (deltas) and consequently these designs are called *sigma-delta* converters.

Carried to its logical extreme delta encoding can be limited to a one-bit representation of the analog signal, an encoding designated *delta modulation*. As in a conventional A/D we observe the signal at uniformly spaced intervals, but now we record only whether the signal has increased or decreased as compared to the last sampling interval. When the signal is sufficiently oversampled, and now we may require extremely high sampling frequencies, we can still recover the original signal. This is the principle behind what is advertised as *one-bit* sampling.

Before leaving our discussion of hardware for moving between analog and digital domains, we should mention D/A designs. D/A devices are in general similar to A/D ones. The first stage of the D/A is the antidigitizer (a device that converts the digital representation into an appropriate analog voltage). In principle there need be no error in such a device, since all digitized levels are certainly available in the continuous world. Next comes the antisampler, which must output the antidigitized values at the appropriate clock times. Once again this can, in principle, be done perfectly. The only quandary is what to output in between sampling instants. We could output zero, but this would require expensive quickly responding circuits, and the resulting analog signal would not really resemble the original signal at all. The easiest compromise is to output a boxcar (piecewise constant) signal, a sort of anti-sample-and-hold! The signal thus created still has a lot of ‘corners’ and accordingly is full of high-frequency components, and must be smoothed by an appropriate low-pass filter. This ‘anti-antialiasing filter’ is what we called the ‘reconstruction filter’ in Figure 1.3. It goes by yet a third name as well, the *sinc* filter, a name that may be understood from equation (2.27).

EXERCISES

- 2.11.1 Design a thermometer to binary converter circuit for an eight level digitizer (one with eight inputs and three outputs). You may only use logical gates, devices that perform the logical NOT, AND, OR, and XOR of their inputs.
- 2.11.2 A useful diagnostic tool for testing A/D circuits is the *level histogram*. One inputs a known signal that optimally occupies the input range and counts the number of times each level is attained. What level histogram is expected for a white noise signal? What about a sinusoid? Write a program and find the histograms for various sounds (e.g., speech, musical instruments).
- 2.11.3 An A/D is said to have *bad transitions* when certain levels *hog* more of the input range than they should. An A/D is said to have a *stuck bit* when an output bit is constant, not dependent on the input signal. Discuss using sawtooth and sinusoidal inputs to test for these malfunctions.

- 2.11.4 A signal that is too weak to be digitized can sometimes be captured using a technique known as dithering whereby a small amount of random noise is added before digitizing. Explain and demonstrate how dithering works.
- 2.11.5 Delta encoding is often allocated fewer bits than actually needed. In this cases we must round the differences to the nearest available level. Assuming uniform spacing of quantization levels, how much noise is introduced as a function of the number of bits. Write a program to simulate this case and try it on a speech signal. It is often the case that smaller differences are more probable than larger ones. How can we exploit this to reduce the quantization error?
- 2.11.6 Fixed step size delta modulation encodes only the sign of the difference between successive signal values, $d_n = \text{sgn}(s_n - s_{n-1})$, but can afford to oversample by b , the number of bits in the original digitized signal. Reconstruction of the signal involves adding or subtracting a fixed δ , according to $\hat{s}_n = \hat{s}_{n-1} + d_n\delta$. What problems arise when δ is too small or too large? Invent a method for fixes these problems and implement it.
- 2.11.7 Prove equation (2.30).

Bibliographical Notes

The material in this chapter is treated to some extent in all of the elementary books on DSP. Probably the first book devoted to DSP was the 1969 text by Gold and Rader [79]. The author, like many others, first learned DSP from the classical text by Oppenheim and Schaffer [185] that has been updated and reissued as [186]. A more introductory text co-authored by Oppenheim is [187]. Another comprehensive textbook with a similar 'engineering approach' is by Proakis and Manolakis [200]. A very comprehensive but condensed source for almost everything related to DSP is the handbook edited by Mitra and Kaiser [241].

More accessible to non-engineers, but at a much more elementary level and covering much less ground is the book by Marven and Ewers [159]. Steiglitz has written a short but informative introductory book [252]. Finally, McLellan, Schaffer and Yoder have compiled a course for first year engineering students that includes demos and labs on a CD [167].