

Signals convey information. Systems transform signals. This book is about developing an understanding of both. We gain this understanding by dissecting their structure and by examining their interpretation. For systems, we look at the relationship between the input and output signals (this relationship is a **declarative** description of the system) and the procedure for converting an input signal into an output signal (this procedure is an **imperative** description of the system).

A sound is a signal. We leave a description of the physics of sound to texts on physics and instead show how a sound can be usefully decomposed into components that themselves have meaning. For example, a musical chord can be decomposed into a set of notes.

An image is a signal. We do not discuss the biophysics of visual perception; instead we show that an image can be usefully decomposed. We can use such decomposition, for instance, to examine what it means for an image to be sharp or blurred and thus to determine how to blur or sharpen an image.

Signals can be more abstract (less physical) than sound or images. A signal can be, for example, a sequence of commands or a list of names. We develop models for such signals and the systems that operate on them, such as a system that interprets a sequence of commands from a musician and produces a sound.

One way to get a deeper understanding of a subject is to formalize it by developing mathematical models. Such models admit manipulation with a level

of confidence not achievable with less formal models. We know that if we follow the rules of mathematics, a transformed model still relates strongly to the original model. There is a sense in which mathematical manipulation preserves “truth” in a way that is elusive with almost any other intellectual manipulation of a subject. We can leverage this truth preservation to gain confidence in the design of a system, to extract hidden information from a signal, or simply to gain insight.

Mathematically, we model both signals and systems as functions. A **signal** is a function that maps a domain, often time or space, into a range, often a physical measure such as air pressure or light intensity. A **system** is a function that maps signals from its domain—its input signals—into signals in its range—its output signals. Both the domain and the range are sets of signals (**signal spaces**). Thus, systems are functions that operate on functions.

We use the mathematical language of sets and functions to make our models unambiguous, precise, and manipulable. This language has its own notation and rules, which are reviewed in appendix A. We begin to use this language in this chapter. Depending on the situation, we represent physical quantities such as time, voltage, current, light intensity, air pressure, or the content of a memory location by variables that range over appropriate sets. For example, discrete time may be represented by a variable $n \in \text{Naturals}$ (the natural numbers) or $n \in \text{Integers}$ (the integers); continuous time may be represented by a variable $t \in \text{Reals}_+$ (the nonnegative real numbers) or $t \in \text{Reals}$ (the real numbers). Light intensity may be represented by a continuous variable $x \in [0, I]$, a range of real numbers from zero to I , where I is some maximum value of the intensity; a variable in a logic circuit may be represented as $x \in \text{Binary}$ (the binary digits). A binary file is an element of Binary^* , the set of sequences of binary digits. A computer name such as `cory.eecs.Berkeley.edu` may be assigned to a variable in Char^* , the set of sequences of characters.

1.1 *Signals*



Signals are functions that carry information, often in the form of temporal and spatial patterns. These patterns may be embodied in various media; radio and broadcast TV signals are electromagnetic waves, and images are spatial patterns of light intensities of different colors. In digital form, images and video are bit strings. Sensors of physical quantities (such as speed, temperature, or pressure) often convert those quantities into electrical voltages, which are then often converted into digital numbers for processing by a computer. In this text, we study systems that store, manipulate, and transmit signals.

In this section, we study signals that occur in human perception, mechanical and electronic sensors, radio and television, and telephone and computer networks, and the description of physical quantities that change over time or over space. The most common feature of these signals is that their domains are sets representing time and space. However, we also study signals that are repre-

sented as sequences of symbols, in which position within the sequence has no particular association with the physical notions of time or space. Such signals are often used to represent sequences of commands or sequences of events.

We will model signals as functions that map a domain (a set) into a range (another set). Our interest for now is to understand through examples how to select the domain and range of signals and how to visualize them. To fully describe a signal, we need to specify not only its domain and range, but also the rules by which the function assigns values. Because the domain and range are often infinite sets, specification of the rules is rarely trivial. Much of the emphasis of subsequent chapters is on how to characterize these rules.

1.1.1 Audio signals

Our ears are sensitive to sound, which is physically just rapid variations in air pressure. Thus sound can be represented as a function,

$$\text{Sound: Time} \rightarrow \text{Pressure},$$

where *Pressure* is a set consisting of possible values of air pressure, and *Time* is a set representing the time interval over which the signal lasts.*

Example 1.1: A one-second segment of a voice signal is a function of the form

$$\text{Voice: } [0, 1] \rightarrow \text{Pressure},$$

where $[0, 1]$ represents one second of time. An example of such a function is plotted in figure 1.1. Such a plot is often called a **waveform**.

The signal in figure 1.1 varies over positive and negative values, averaging approximately zero. But air pressure cannot be negative, and so the vertical axis does not literally represent air pressure. It is customary to normalize the representation of sound by subtracting the ambient air pressure (about 100,000 newtons per square meter) from the range. Our ears are, after all, not sensitive to constant ambient air pressure. Thus, we assume *Pressure* = *Reals* (the real numbers), in which negative pressure means a drop in pressure in relation to ambient air pressure.

In fact, the possible values of the function *Voice* as shown in figure 1.1 are 16-bit integers, suitable for storage in a computer. We can call the set of 16-bit integers *Integers16* = $\{-32768, \dots, 32767\}$. The audio hardware of the computer is responsible for converting members of the set *Integers16* into air pressure.

*For a review of the notation of sets and functions, see appendix A.

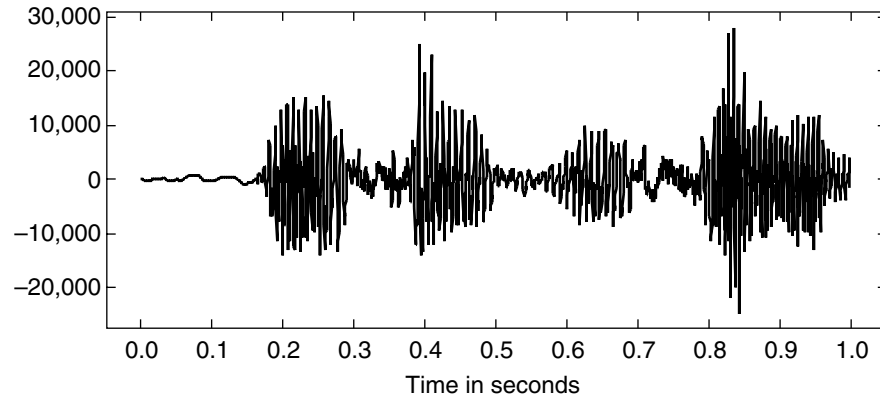


FIGURE 1.1: Waveform of a speech fragment.

The numbers in the computer representation *Integers16* are a subset of *Reals*. The units of air pressure in this representation are arbitrary; therefore, to convert to units of newtons per square meter, we need to multiply these numbers by some constant. The value of this constant depends on the audio hardware of the computer and on its volume setting. □

The horizontal axis in figure 1.1 suggests that time varies continuously from 0 to 1.0. However, a computer cannot directly handle such a continuum. The sound is represented not as a continuous waveform but rather as a list of numbers (for voice-quality audio, 8,000 numbers for every second of speech).^{*} A close-up of a section of the speech waveform is shown in figure 1.2. That plot shows 100 data points (called **samples**). For emphasis, that plot shows a dot for each sample, rather than a continuous curve, and a stem connecting the dot to the horizontal axis. Such a plot is called a **stem plot**. Because there are 8,000 samples per second, the 100 points in figure 1.2 represent $100/8,000$ seconds, or 12.5 milliseconds of speech.

Such signals are said to be **discrete-time signals** because they are defined only at discrete points in time. A discrete-time one-second voice signal in a computer is a function,

$$\text{ComputerVoice: } \text{DiscreteTime} \rightarrow \text{Integers16},$$

where $\text{DiscreteTime} = \{0, 1/8,000, 2/8,000, \dots, 8,000/8,000\}$ is the set of sampling times.

^{*}In a compact disc (CD), there are 44,100 numbers per second of sound per stereo channel.

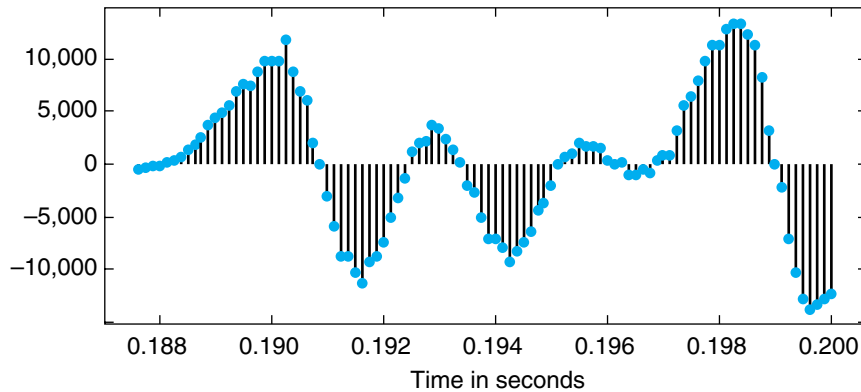


FIGURE 1.2: Discrete-time representation of a speech fragment.

In contrast, **continuous-time signals** are functions defined over a continuous interval of time (technically, a continuum in the set *Reals*). The audio hardware of the computer is responsible for converting the *ComputerVoice* function into a function of the form $Sound: Time \rightarrow Pressure$. That hardware, which converts an input signal into a different output signal, is a system.

Example 1.2: We cannot represent the function *Voice* of Example 1.1 by a mathematical expression. We now consider an example in which there is such an expression. The sound emitted by a precisely tuned and idealized 440-Hz tuning fork over the infinite time interval $Reals = (-\infty, \infty)$ is the function

$$PureTone: Reals \rightarrow Reals,$$

where the time-to-(normalized) pressure assignment is*

$$\forall t \in Reals, \quad PureTone(t) = P \sin(2\pi \times 440t).$$

Here, P is the **amplitude** of the sinusoidal signal *PureTone*. It is a real-valued constant.

Figure 1.3 is a graph of a portion of this pure tone (showing only a subset of the domain *Reals*). In the figure, $P = 1$. □

* If the notation here is unfamiliar, see appendix A.

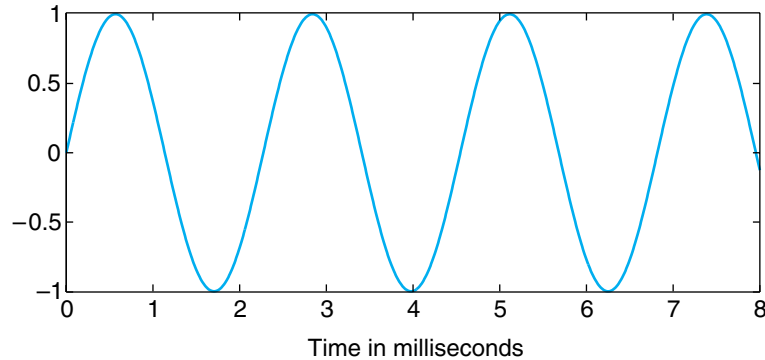


FIGURE 1.3: Portion of the graph of a pure tone with frequency 440 Hz.

The number 440 in this example is the **frequency** of the sinusoidal signal shown in figure 1.3, in units of **cycles per second** or **Hertz (Hz)**.^{*} It simply means that the sinusoid completes 440 cycles per second. Alternatively, the sinusoid completes one cycle in $1/440$ seconds, or about 2.3 milliseconds. The time to complete one cycle, 2.3 milliseconds, is called the **period**.

The *Voice* signal in figure 1.1 is much more irregular than *PureTone* in figure 1.3. According to an important theorem that we study in subsequent chapters, a function like *Voice*, despite its irregularity, is a sum of signals of the form of *PureTone* but with different frequencies. A sum of two pure tones of frequencies—say, 440 Hz and 660 Hz—is the function *SumOfTones: Reals* \rightarrow *Reals*, given by

$$\forall t \in \text{Reals}, \quad \text{SumOfTones}(t) = P_1 \sin(2\pi \times 440t) + P_2 \sin(2\pi \times 660t).$$

Notice that summing two signals amounts to adding the values of their functions at each point in the domain. Of course, two signals can be added only if they have the same domain and compatible ranges (so that addition is defined between an element of one range and an element of the other). The two components are shown in figure 1.4. At any point on the horizontal axis, the value of the sum is simply the addition of the values of the two components.

^{*}The unit of frequency called Hertz is named after physicist Heinrich Rudolf Hertz (1857–1894) for his research in electromagnetic waves.

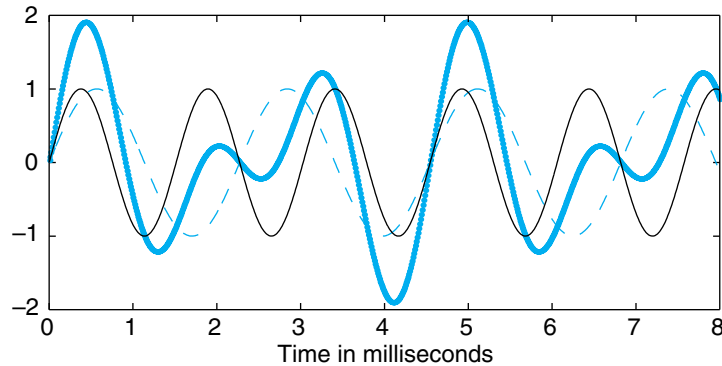


FIGURE 1.4: Sum (color line) of two pure tones, one at 440 Hz (dashed line) and the other at 660 Hz (black line).

Household electrical power

PROBING FURTHER

In the United States, household current is delivered on three wires: a **neutral wire** and two **hot wires**. The voltage between either hot wire and the neutral wire is around 110 to 120 volts, RMS (root mean square, the square root of the average of the voltage squared). The voltage between the two hot wires is around 220 to 240 volts, RMS. The higher voltage is used for appliances that need more power, such as air conditioners. Here, we examine exactly how this works.

The voltage between the hot wires and the neutral wire is sinusoidal, with a frequency of 60 Hz. Thus, for one of the hot wires, it is a function $x: \text{Reals} \rightarrow \text{Reals}$, where the domain represents time and the range represents voltage, and

$$\forall t \in \text{Reals}, \quad x(t) = 170 \cos(60 \times 2\pi t).$$

This 60-Hz sinusoidal waveform completes one cycle in a period of $T = 1/60$ seconds. Why is the amplitude 170 volts, rather than 120? Because the 120 voltage is **RMS (root mean square)**; that is,

$$\text{voltage}_{RMS} = \sqrt{\frac{1}{T} \int_0^T x^2(t) dt} \text{ volts,}$$

the square root of the average of the square of the voltage. This is calculated to be 120.

continued on next page

**PROBING
FURTHER**

The voltage between the second hot wire and the neutral wire is a function $y: \mathbb{R} \rightarrow \mathbb{R}$, where

$$\forall t \in \mathbb{R}, \quad y(t) = -170 \cos(60 \times 2\pi t) = -x(t).$$

It is the negative of the other voltage at any time t . This sinusoidal signal is said to have a **phase shift** of 180 degrees, or π radians, in comparison with the first sinusoid. Equivalently, it is said to be 180 degrees **out of phase**.

We can now see how to get the higher voltage for power-hungry appliances. We simply use the two hot wires, rather than one hot wire and the neutral wire. The voltage between the two hot wires is the difference, a function $z: \mathbb{R} \rightarrow \mathbb{R}$, where

$$\forall t \in \mathbb{R}, \quad z(t) = x(t) - y(t) = 340 \cos(60 \times 2\pi t).$$

This corresponds to 240 volts, RMS. A plot is shown in figure 1.5.

The neutral wire should not be confused with the ground wire in a three-prong plug. The ground wire is a safety feature to allow current to flow into the earth rather than, say, through a person.

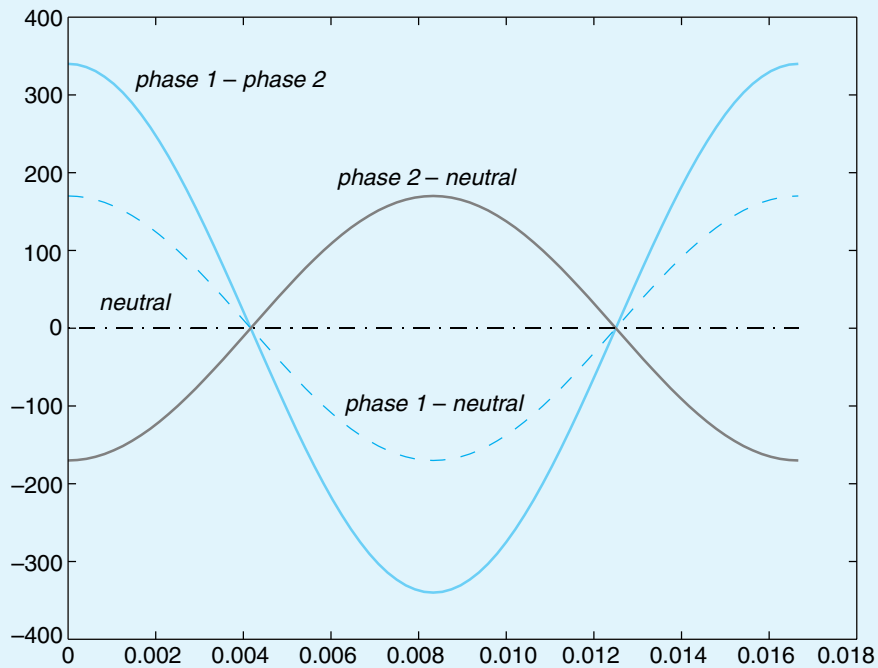


FIGURE 1.5: The voltages between the two hot wires and the neutral wire (*phase 1–neutral* and *phase 2–neutral*) and between the two hot wires (*phase 1–phase 2*) in household electrical power in the United States.

1.1.2 Images

If an image is a grayscale picture on an 11×8.5 inch sheet of paper, the picture is represented by a function

$$\text{Image}: [0, 11] \times [0, 8.5] \rightarrow [0, B_{max}], \quad (1.1)$$

where B_{max} is the maximum grayscale value (0 is black and B_{max} is white). The set $[0, 11] \times [0, 8.5]$ defines the space of the sheet of paper. More generally, a grayscale image is a function

$$\text{Image}: \text{VerticalSpace} \times \text{HorizontalSpace} \rightarrow \text{Intensity},$$

where $\text{Intensity} = [\text{black}, \text{white}]$ is the intensity range from *black* to *white* measured in some scale. An example is shown in figure 1.6.

For a color picture, the reflected light is sometimes measured in terms of its RGB values (the magnitudes of the red, green, and blue colors), and so a color picture is represented by a function

$$\text{ColorImage}: \text{VerticalSpace} \times \text{HorizontalSpace} \rightarrow \text{Intensity}^3.$$

The RGB values assigned by *ColorImage* at any point (x, y) in its domain is the triple $(r, g, b) \in \text{Intensity}^3$, given by

$$(r, g, b) = \text{ColorImage}(x, y).$$

Different images are represented by functions with different spatial domains (the size of the image may be different), different ranges (we may consider a more or less detailed way of representing light intensity and color than grayscale or RGB values), and different assignments of color values to points in the domain.

Because a computer has finite memory and finite word length, an image is stored by discretizing both the domain and the range, as in the function *ComputerVoice*. Thus, for example, your computer may represent an image by storing a function of the form

$$\text{ComputerImage}: \text{DiscreteVerticalSpace} \times \text{DiscreteHorizontalSpace} \rightarrow \text{Integers}_8,$$



FIGURE 1.6: Grayscale image (left) and its enlarged pixels (right).

where

$$\text{DiscreteVerticalSpace} = \{1, 2, \dots, 300\},$$

$$\text{DiscreteHorizontalSpace} = \{1, 2, \dots, 200\}, \text{ and}$$

$$\text{Integers8} = \{0, 1, \dots, 255\}.$$

It is customary to say that *ComputerImage* stores 300×200 pixels, whereby a **pixel** is an individual picture element. The value of a pixel is *ComputerImage* $(\text{row}, \text{column}) \in \text{Integers8}$, where $\text{row} \in \text{DiscreteVerticalSpace}$ and $\text{column} \in \text{DiscreteHorizontalSpace}$. In this example, the range *Integers8* has 256 elements; therefore, in the computer these elements can be represented by an eight-bit integer (hence the name of the range, *Integers8*). An example of such an image is shown in figure 1.6, in which the right image is part of the left image magnified to show the discretization implied by the individual pixels.

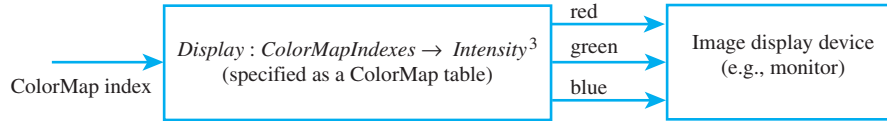


FIGURE 1.7: In a computer representation of a color image that uses a colormap, pixel values are elements of the set $ColorMapIndexes$. The function $Display$ converts these indexes to an RGB (red, green, blue) representation.

A computer could store a color image in one of two ways. One way is to represent it as a function,

$$ColorComputerImage: DiscreteVerticalSpace \times DiscreteHorizontalSpace \rightarrow Integers8^3, \quad (1.2)$$

so that each pixel value is an element of $\{0, 1, \dots, 255\}^3$. Such a pixel can be represented as three eight-bit integers. A common method that saves memory is to use a **colormap**. Define the set $ColorMapIndexes = \{0, \dots, 255\}$, together with a $Display$ function,

$$Display: ColorMapIndexes \rightarrow Intensity^3. \quad (1.3)$$

$Display$ assigns to each element of $ColorMapIndexes$ the three (r, g, b) color intensities. This is depicted in the block diagram in figure 1.7. Use of a colormap reduces the memory required to store an image by a factor of three because each pixel is now represented by a single eight-bit number. But only 256 colors can be represented in any given image. The function $Display$ is typically represented in the computer as a lookup table (see lab 2 in the Lab Manual at www.aw.com/lee_varaiya).



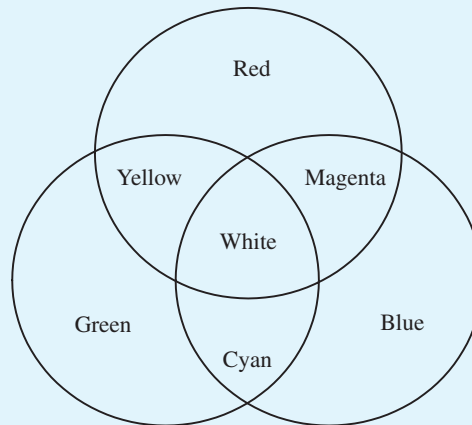
1.1.3 Video signals

A **video** is a sequence of images displayed at a certain rate (frequency). National Television System Committee (NTSC) video (the standard for analog video in the United States) displays 30 images (called **frames**) per second. The eye is unable to distinguish between successive images displayed at this frequency, and so a television broadcast appears to be continuously varying in time.

**PROBING
FURTHER***Color and light*

The human eye is sensitive to electromagnetic waves of certain frequency, which we call light. The frequency f in Hertz of a purely sinusoidal electromagnetic wave is related to its *wavelength* λ in meters by the formula $f = c/\lambda$, where c is the speed of light (about 3×10^8 meters/second). The wavelengths of visible light range from about 350–400 nanometers, or 10^{-9} meters, to 750–800 nanometers (nm). We experience light of different wavelengths as having different colors: violet (350 nm), indigo, blue, green, yellow, orange, and red (800 nm).

The retina has three groups of cones, each sensitive to one of the three primary colors: red, green, and blue. Other colors are perceived when these three groups are stimulated in different combinations, as shown in the diagram. By combining its red, green, and blue light sources in different amounts, a computer monitor can create the perception of all colors. The color white is obtained by adding all three primary colors, and the absence of any light is perceived as black. This “additive” model of color perception was proposed in 1802 by Thomas Young and H. L. F. Helmholtz.



In a computer, if the amount of each primary color is represented by an eight-bit word and each color is represented by three eight-bit words, the computer could generate $2^8 \times 2^8 \times 2^8 = 2^{24} = 16,777,216$ colors. An eight-bit color map, in contrast, can generate only 256 colors.

Painting works by subtraction: different pigments of color absorb (subtract) light of different wavelengths. The primary subtractive colors are magenta, yellow, and cyan. Similarly, if you wear rose-tinted glasses, things look pink because the glasses filter out the nonred wavelengths.

The ear and the eye are very different perceptual systems. If we listen to a sound consisting of the sum of two pure tones, we can distinguish the two tones. However, we cannot perceive the difference between, say, a yellow light source and an appropriate combination of red and green sources. The ear can be modeled as a linear time-invariant system (see chapter 5); the eye cannot.

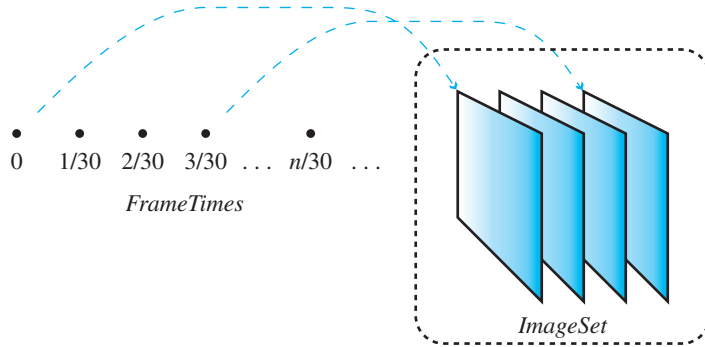


FIGURE 1.8: Illustration of the function *Video*.

Thus, the domain of a video signal is discrete time, $FrameTimes = \{0, 1/30, 2/30, \dots\}$, and its range is a set of images, $ImageSet$. For **analog video**, each image in $ImageSet$ is a function of the form

$$VideoFrame: DiscreteVerticalSpace \times HorizontalSpace \rightarrow Intensity^3.$$

An analog video signal is discretized in the vertical direction but not in the horizontal direction.* The image is composed of a set of horizontal lines called **scan lines**, in which the intensity varies along the line. The horizontal lines are stacked vertically to form an image.

A video signal is therefore a function

$$Video: FrameTimes \rightarrow ImageSet. \quad (1.4)$$

For any time $t \in FrameTimes$, the image $Video(t) \in ImageSet$ is displayed. The signal *Video* is illustrated in figure 1.8.

An alternative way of specifying a video signal is by the function *Video'*, whose domain is a product set as follows:

$$Video': FrameTimes \times DiscreteVerticalSpace \times HorizontalSpace \rightarrow Intensity^3.$$

Like *Video* in figure 1.8, *Video'* is depicted in figure 1.9. The RGB value assigned to a point (x, y) at time t is

$$(r, g, b) = Video'(t, x, y). \quad (1.5)$$

*This is a simplification. Most analog video images are **interlaced**, which means that successive frames use different sets for *DiscreteVerticalSpace* so that scan lines in one frame lie in between the scan lines of the previous frame. Also, the range $Intensity^3$ has an interesting structure that ensures compatibility between black-and-white and color television sets.

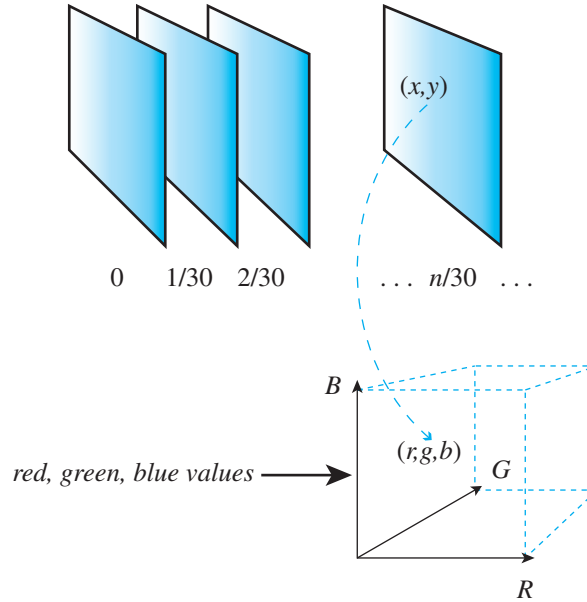


FIGURE 1.9: Illustration of the function $Video'$.

If the signals specified in (1.4) and (1.5) represent the same video, then for all $t \in FrameTimes$ and $(x, y) \in DiscreteVerticalSpace \times HorizontalSpace$,

$$(Video(t))(x, y) = Video'(t, x, y). \quad (1.6)$$

It is worth pausing to understand the notation used in (1.6). $Video$ is a function of t , and so $Video(t)$ is an element in its range $ImageSet$. Because elements in $ImageSet$ themselves are functions, $Video(t)$ is a function. The domain of $Video(t)$ is the product set $DiscreteVerticalSpace \times HorizontalSpace$, and so $(Video(t))(x, y)$ is the value of this function at the point (x, y) in its domain. This value is an element of $Intensity^3$. On the right side of (1.6), $Video'$ is a function of (t, x, y) , and so $Video'(t, x, y)$ is an element in its range, $Intensity^3$. The equality (1.6) asserts that for all values of t, x, y , the two sides are the same. On the left side of (1.6), the parentheses enclosing $Video(t)$ are not necessary; we could equally well write $Video(t)(x, y)$. However, the parentheses improve readability.

1.1.4 Signals representing physical attributes

The changes over time in the attributes of a physical object or device can be represented as functions of time or space.

Example 1.3: The position of an airplane can be expressed as

$$\text{Position: Time} \rightarrow \text{Reals}^3,$$

where

$$\forall t \in \text{Time}, \quad \text{Position}(t) = (x(t), y(t), z(t))$$

is its position in three-dimensional space at time t . The position and velocity of the airplane is a function

$$s: \text{Time} \rightarrow \text{Reals}^6, \quad (1.7)$$

where

$$s(t) = (x(t), y(t), z(t), v_x(t), v_y(t), v_z(t)) \quad (1.8)$$

gives its position and velocity at $t \in \text{Time}$.

The position of the pendulum shown on the left side of figure 1.10 is represented by the function

$$\theta: \text{Time} \rightarrow [-\pi, \pi],$$

where $\theta(t)$ is the angle with the vertical made by the pendulum at time t .

The position of the upper and lower arms of a robot arm depicted on the right side of figure 1.10 can be represented by the function

$$(\theta_u, \theta_l): \text{Time} \rightarrow [-\pi, \pi]^2,$$

where $\theta_u(t)$ is the angle at the shoulder made by the upper arm with the vertical at time t , and $\theta_l(t)$ is the angle made by the lower arm at the elbow at time t . Note that we can regard (θ_u, θ_l) as a single function with range as the product set $[-\pi, \pi]^2$ or as two functions, θ_u and θ_l , each with range $[-\pi, \pi]$. Similarly, we can regard s in (1.7) as a single function with range Reals^6 or as a collection of six functions, each with range Reals , as suggested by (1.8). \square

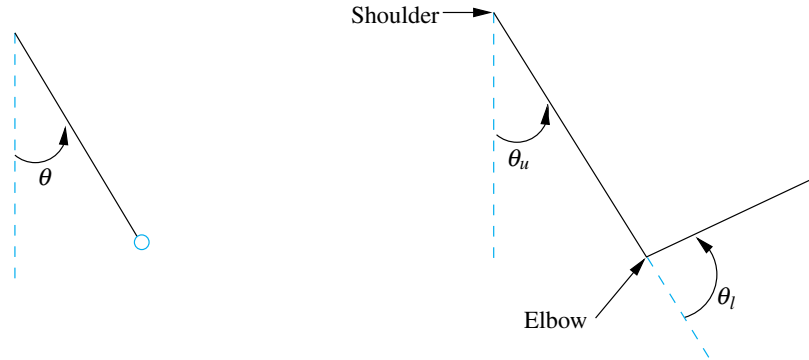


FIGURE 1.10: Position of a pendulum (left) and upper and lower arms of a robot (right).

Example 1.4: The spatial variation of temperature over some volume of space can be represented by a function

$$\text{AirTemp}: X \times Y \times Z \rightarrow \text{Reals},$$

where $X \times Y \times Z \subset \text{Reals}^3$ is the volume of interest, and $\text{AirTemp}(x, y, z)$ is the temperature at the point (x, y, z) . \square

1.1.5 Sequences

We have studied examples in which temporal or spatial information is represented by functions of a variable representing time or space. The domain of time or space may be continuous, as in *Voice* and *Image*, or discrete, as in *ComputerVoice* and *ComputerImage*.

In many situations, information is represented as **sequences** of symbols rather than as functions of time or space. These sequences occur in two ways: as a representation of **data** or as a representation of an **event stream**. Sequences, in fact, are special sorts of functions.

Examples of data represented by sequences are common. A file stored in a computer in binary form is a sequence of bits, or binary symbols—that is, a sequence of 0s and 1s. A text, like that in this book, is a sequence of words. A sheet of music represents a sequence of notes.

Example 1.5: Consider an N -bit-long binary file,

$$b_1, b_2, \dots, b_N,$$

where each $b_i \in \text{Binary} = \{0, 1\}$. We can regard this file as a function

$$\text{File}: \{1, 2, \dots, N\} \rightarrow \text{Binary},$$

with the assignment $\text{File}(n) = b_n$ for every $n \in \{1, \dots, N\}$.

Sometimes we can give a mathematical expression for a binary signal. For instance, the N -bit-long binary file *Alt*, consisting of an alternating sequence of 0s and 1s, is given as follows:

$$\forall n \in \{1, 2, \dots, N\}, \quad \text{Alt}(n) = \begin{cases} 0, & n \text{ even} \\ 1, & n \text{ odd} \end{cases}$$

If instead of *Binary* we take the range to be *EnglishWords*, then an N -word-long English text is a function

$$\text{EnglishText}: \{1, 2, \dots, N\} \rightarrow \text{EnglishWords}. \quad \square$$

In general, data sequences are functions of the form

$$\boxed{\text{Data}: \text{Indices} \rightarrow \text{Symbols}}, \quad (1.9)$$

where $\text{Indices} \subset \text{Naturals}$ —in which *Naturals* is the set of natural numbers—is an appropriate index set such as $\{1, 2, \dots, N\}$, and *Symbols* is an appropriate set of symbols such as *Binary* or *EnglishWords*.

One advantage of the representation (1.9) is that we can then interpret *Data* as a discrete-time signal, and so some of the techniques that we develop in later chapters for those signals will automatically apply to data sequences. However, the domain *Indices* in (1.9) does not represent uniformly spaced instances of time. All we can say is that if m and n are in *Indices* with $m < n$, then the m th symbol $\text{Data}(m)$ occurs in the data sequence *before* the n th symbol $\text{Data}(n)$, but we cannot say how much time elapses between the occurrence of those two symbols.

The second way in which sequences arise is as representations of event streams. An **event stream**, or **trace**, is a record or log of the significant events that occur in a system of interest. Here are some everyday examples.

Example 1.6: When you call someone by phone, the normal sequence of events is

LiftHandset, HearDialTone, DialDigits, HearTelephoneRing,
HearCalleeAnswer, . . . ,

but if the other phone is busy, the event trace is

LiftHandset, HearDialTone, DialDigits, HearBusyTone,

When you send a file to be printed, the normal trace of events is

CommandPrintFile, FilePrinting, PrintingComplete,

but if the printer has run out of paper, the trace might be

CommandPrintFile, FilePrinting, MessageOutOfPaper, InsertPaper, . . . □

Example 1.7: When you enter your car, the starting trace of events might be

StartEngine, SeatbeltSignOn, BuckleSeatbelt, SeatbeltSignOff, . . .

Thus, event streams are functions of the form

EventStream: Indices → *Symbols.* □

We show in chapter 3 that the behavior of finite state machines is best described in terms of event traces, and that systems that operate on event streams are often best described as finite-state machines.

1.1.6 *Discrete signals and sampling*

Voice and *PureTone* are said to be continuous-time signals because their domain *Time* is a continuous interval of the form $[\alpha, \beta] \subset \text{Reals}$. The domain of *Image*, similarly, is a continuous two-dimensional rectangle of the form $[a, b] \times [c, d] \subset \text{Reals}^2$. The signals *ComputerVoice* and *ComputerImage* have domains of time and space that are discrete sets. *Video* is also a discrete-time signal, but in principle it could be a function of a space continuum. We can define a function *ComputerVideo* in which all three sets that are composed to form the domain are discrete.

Discrete signals often arise from signals with continuous domains by **sampling**. We briefly explain the purpose of sampling here; a detailed discussion is taken up later. Continuous domains have an infinite number of elements. Even the domain $[0, 1] \subset \text{Time}$, representing a finite time interval, has an infinite number of elements. The signal assigns a value in its range to each of these infinitely many elements. Such a signal cannot be stored in a finite digital memory device such as a computer or CD-ROM. If we wish to store, say, *Voice*, we must approximate it by a signal with a finite domain.

A common way to approximate a function with a continuous domain, such as *Voice* and *Image*, by a function with a finite domain is by uniformly sampling its continuous domain.

Example 1.8: If we sample a 10-second-long domain of *Voice*,

$$\text{Voice: } [0, 10] \rightarrow \text{Pressure},$$

10,000 times a second (i.e., at a frequency of 10 kHz), we get the signal

$$\text{SampledVoice: } \{0, 0.0001, 0.0002, \dots, 9.9998, 9.9999, 10\} \rightarrow \text{Pressure}, \quad (1.10)$$

with the assignment

$$\text{SampledVoice}(t) = \text{Voice}(t), \forall t \in \{0, 0.0001, 0.0002, \dots, 9.9999, 10\}. \quad (1.11)$$

Notice from (1.10) that uniform sampling means picking a uniformly spaced subset of points of the continuous domain $[0, 10]$. \square

In the example, the **sampling interval**, or **sampling period**, is 0.0001 second, which corresponds to a **sampling frequency**, or **sampling rate**, of 10,000 Hz. Because the continuous domain is 10 seconds long, the domain of *SampledVoice* has 100,000 points. A sampling frequency of 5,000 Hz would yield the domain $\{0, 0.0002, \dots, 9.9998, 10\}$, which has half as many points. The sampled domain is finite, and its elements are discrete values of time.

Notice also from (1.11) that the pressure assigned by *SampledVoice* to each time in its domain is the same as that assigned by *Voice* to the same time; that is, *SampledVoice* is indeed obtained by sampling the *Voice* signal at discrete values of time.

Figure 1.11 shows an exponential function *Exp*: $[-1, 1] \rightarrow \text{Reals}$ defined by

$$\text{Exp}(x) = e^x.$$

SampledExp is obtained by sampling with a sampling interval of 0.2. Thus, its domain is

$$\{-1, -0.8, \dots, 0.8, 1.0\}.$$

The continuous domain of *Image* given by (1.1), which describes a grayscale image on an 8.5×11 inch sheet of paper, is the rectangle $[0, 11] \times [0, 8.5]$, representing the space of the page. In this case, too, a common way to approximate *Image* by a signal with a finite domain is to sample the rectangle. Uniform sampling with a **spatial resolution** of, say, 100 dots per inch in each dimension

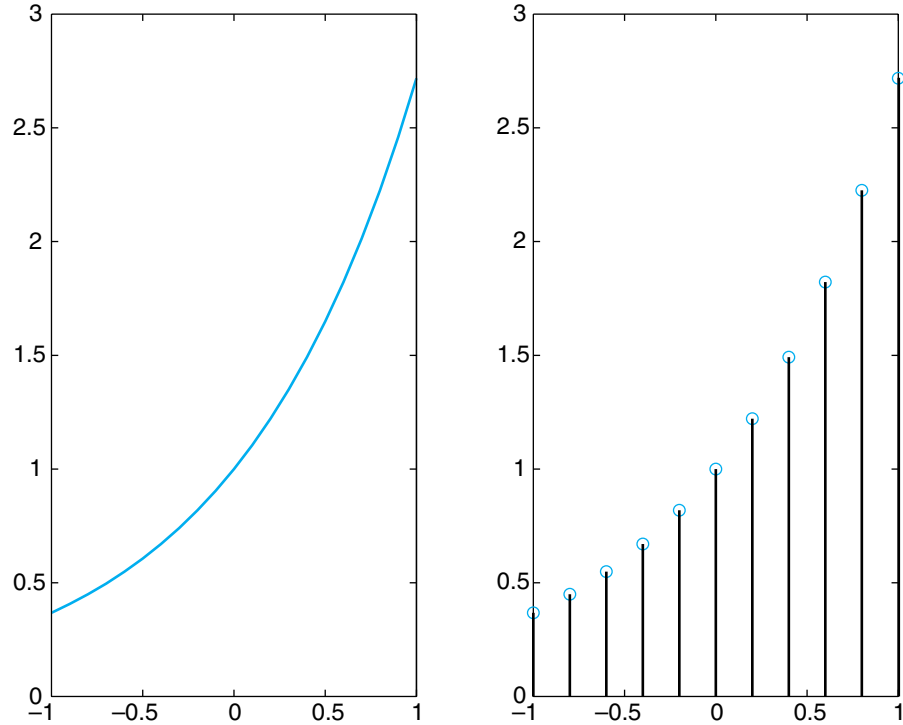


FIGURE 1.11: The exponential functions Exp (left) and $SampledExp$ (right), obtained by sampling with a sampling interval of 0.2.

yields the finite domain $D = \{0, 0.01, \dots, 8.49, 8.5\} \times \{0, 0.01, \dots, 10.99, 11.0\}$. Therefore, the sampled grayscale picture is

$$SampledImage: D \rightarrow [0, B_{max}]$$

with

$$SampledImage(x, y) = Image(x, y), \forall (x, y) \in D.$$

As mentioned before, each sample of the image is called a pixel, and the size of the image is often given in pixels. The size of a computer screen display, for example, may be 600×800 or $768 \times 1,024$ pixels.

Sampling and approximation

Let f be a continuous-time function, and let $Sampledf$ be the discrete-time function obtained by sampling f . Suppose we are given $Sampledf$, as, for example, in the left graph of figure 1.12. Can we reconstruct or recover f from $Sampledf$? This question lies at the heart of digital storage and communication technologies. The general answer to this question tells us, for example, what audio quality we can

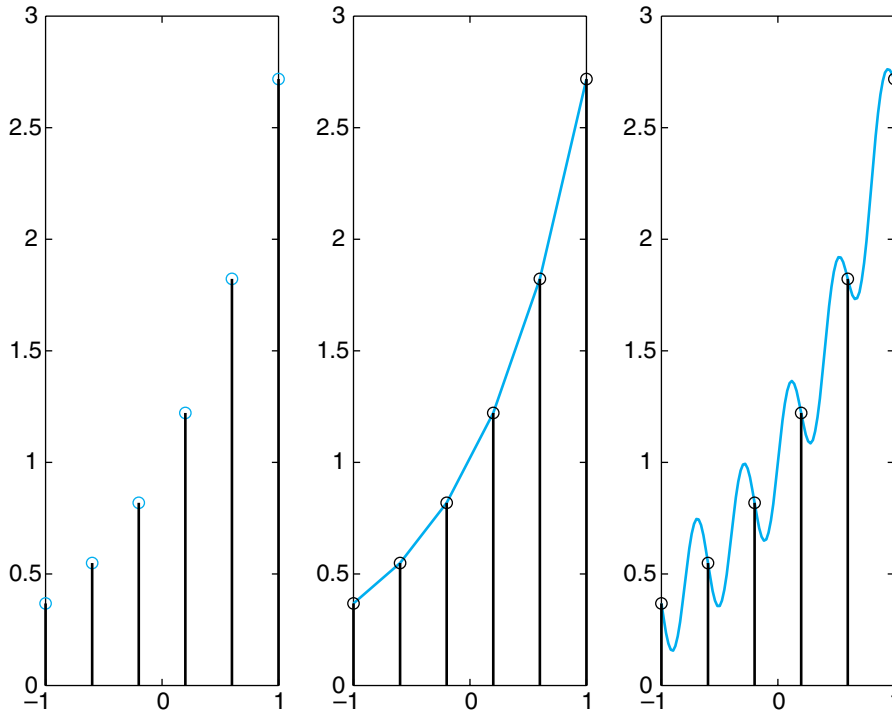


FIGURE 1.12: The discrete-time signal (left) is obtained by sampling the continuous-time signal (middle or right).

obtain from a given discrete representation of a sound. The format for a compact disc is based on the answer to this question. We discuss it in much detail in later chapters.

For the moment, note that the short answer to the question above is no. For example, we cannot tell whether the discrete-time function in the left graph of figure 1.12 was obtained by sampling the continuous-time function in the middle graph or the function in the right graph. Indeed, there are infinitely many such functions, and a choice must be made. One option is to connect the sampled values by straight line segments, as shown in the middle graph. Another choice is shown in the right graph. The choice made by a compact disc player is different from both of these, as explored further in chapter 11.

Similarly, an image like *Image* cannot be uniquely recovered from its sampled version *SampledImage*. Several different choices are commonly used.

Digital signals and quantization

Even though *SampledVoice* in example 1.8 has a finite domain, we may nonetheless be unable to store it in a finite amount of memory. To see why, suppose that the range *Pressure* of the function *SampledVoice* is the continuous interval $[a, b]$.

To represent every value in $[a, b]$ requires infinite **precision**. In a computer, in which data are represented digitally as finite collections of bits, such precision would require an infinite number of bits for just one sample. But a finite digital memory has a finite word length in which we can store only a finite number of values. For instance, if a word is eight bits long, it can have 2^8 , or 256, different values. So we must approximate each number in the range $[a, b]$ by one of 256 values. The most common approximation method is to **quantize** the signal. A common approach is to choose 256 uniformly spaced values in the range $[a, b]$ and to approximate each value in $[a, b]$ by the one of the 256 values that is closest. An alternative approximation, called **truncation**, is to choose the largest of the 256 values that is less than or equal to the desired value.

Example 1.9: Figure 1.13 shows a *PureTone* signal, a *SampledPureTone* obtained after sampling, and a quantized *DigitalPureTone* obtained by means of four-bit, or 16-level, truncation. *PureTone* has continuous domain and continuous range, whereas *SampledPureTone* (depicted with circles) has discrete

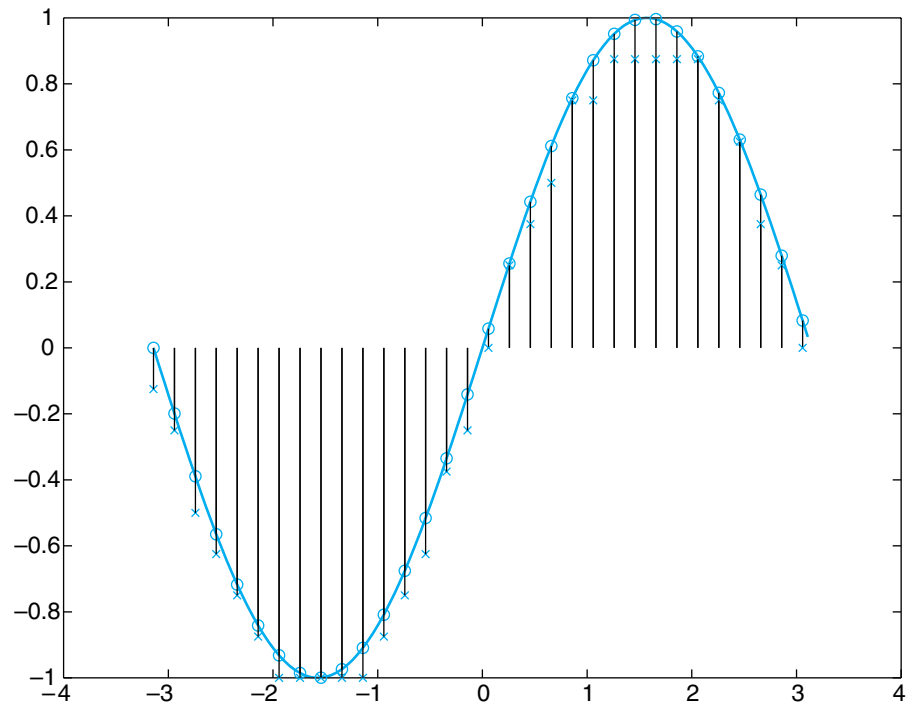


FIGURE 1.13: *PureTone* (continuous curve), *SampledPureTone* (circles), and *DigitalPureTone* signals (\times s).

domain and continuous range, and *DigitalPureTone* (depicted with \times s) has discrete domain and discrete range. Only the last of these can be precisely represented on a computer. \square

It is customary to call a signal with continuous domain and continuous range, such as *PureTone*, an **analog signal** and to call a signal with discrete domain and range, such as *DigitalPureTone*, a **digital signal**.

Example 1.10: In digital telephony, voice is sampled every 125 microseconds (μsec) or at a sampling frequency of 8,000 Hz. Each sample is quantized into an eight-bit word, or 256 levels. This yields an overall rate of $8,000 \times 8 = 64,000$ bits per second. The worldwide digital telephony network is therefore composed primarily of channels capable of carrying 64,000 bits per second or multiples of this (so that multiple telephone channels can be carried together). In cellular phones, voice samples are further compressed to bit rates of 8,000 to 32,000 bits per second. \square

1.2 Systems

Systems are functions that transform signals. There are many reasons for transforming signals. A signal carries information. A transformed signal may carry the same information in a different way. For example, in a live concert, music is represented as sound. A recording system may convert that sound into a pattern of magnetic fields on a magnetic tape. The original signal, the sound, is difficult to preserve for posterity. The magnetic tape has a more persistent representation of the same information. Thus, **storage** is one of the tasks accomplished by systems.

A system may transform a signal into a form that is more convenient for **transmission**. Sound signals cannot be carried by the Internet. There is simply no physical mechanism in the Internet for transporting rapid variations in air pressure. The Internet provides instead a mechanism for transporting sequences of bits. A system must convert a sound signal into a sequence of bits. Such a system is called an **encoder** or **coder**. At the far end, of course, a **decoder** is needed to convert the sequence back into sound. When a coder and a decoder are combined into the same physical device, the device is often called a **codec**.

A system may transform a signal to hide its information so that snoopers do not have access to it. This is called **encryption**. For encryption to be useful, matching **decryption** is needed.

A system may **enhance** a signal by emphasizing some of the information it carries and deemphasizing some other information. For example, an **audio equalizer** may compensate for poor room acoustics by reducing the magnitude of certain low frequencies that happen to resonate in the room. In transmission, signals are often degraded by **noise** or distorted by physical effects in the transmission medium. A system may attempt to reduce the noise or reverse the



distortion. When the signal is carrying digital information over a physical channel, the extraction of the digital information from the degraded signal is called **detection**.

Systems are also designed to **control** physical processes such as the heating in a room, the ignition in an automobile engine, and the flight of an aircraft. The state of the physical process (room temperature, cylinder pressure, aircraft speed) is sensed. The sensed signal is processed to generate signals that drive actuators, such as motors or switches. Engineers design a system called the **controller** that, on the basis of the processed sensor signal, determines the signals that control the physical process (turn the heater on or off, adjust the ignition timing, change the aircraft flaps) so that the process has the desired behavior (room temperature adjusts to the desired setting, engine delivers more torque, aircraft descends smoothly).

Systems are also designed for **translation** from one format to another. For example, a command sequence from a musician may be transformed into musical sounds. Or the detection of risk of collision in an aircraft might be translated into control signals that trigger evasive maneuvers.

1.2.1 *Systems as functions*

Consider a system S that transforms input signal x into output signal y . The system is a function, so $y = S(x)$. Suppose $x: D \rightarrow R$ is a signal with domain D and range R . For example, x might be a sound, $x: \text{Reals} \rightarrow \text{Pressure}$. The domain of S is the set X of all such sounds, which we write

$$X = [D \rightarrow R] = \{x \mid x: D \rightarrow R\}. \quad (1.12)$$

This notation reads “ X , also written $[D \rightarrow R]$, is the set of all x such that x is a function from D to R .” This set is called a **signal space** or **function space**. A signal or function space is a set of all functions with a given domain and range.

Example 1.11: The set of all sound segments with duration $[0, 1]$ and range *Pressure* is written

$$[[0, 1] \rightarrow \text{Pressure}].$$

Notice that square brackets are used for both a subset of reals, as in $[0, 1]$, and for a function space, as in $[D \rightarrow R]$, although the meanings of these two notations are obviously very different.

The set *ImageSet* considered in section 1.1.3 is the function space

$$\text{ImageSet} = [\text{DiscreteVerticalSpace} \times \text{HorizontalSpace} \rightarrow \text{Intensity}^3].$$

Because this is a set, we can define functions that use it as a domain or range, as we have done earlier with

$$\text{Video: } \text{FrameTimes} \rightarrow \text{ImageSet}.$$

Similarly, the set of all binary files of length N is

$$\text{BinaryFiles} = [\text{Indices} \rightarrow \text{Binary}],$$

where $\text{Indices} = \{1, \dots, N\}$. \square

A system S is a function mapping a signal space into a signal space,

$$S: [D \rightarrow R] \rightarrow [D' \rightarrow R'].$$

Systems are therefore much like signals, except that both their domain and their range are signal spaces. Thus, if $x \in [D \rightarrow R]$ and $y = S(x)$, then it must be that $y \in [D' \rightarrow R']$. Furthermore, if z is an element of D' , $z \in D'$, then

$$y(z) = S(x)(z) = (S(x))(z) \in R'.$$

The parentheses around $S(x)$ in $(S(x))(z)$ are not necessary but may improve readability.

1.2.2 Telecommunications systems

We give some examples of systems that occur in or interact with the global telecommunications network. This network is unquestionably one of the most remarkable accomplishments of humankind. It is astonishingly complex, composed of hundreds of distinct corporations, and linking billions of people. We often think of it in terms of its basic service, plain old telephone service (**POTS**). POTS is a voice service, but the telephone network is in fact a global, high-speed digital network that carries not just voice but also video, images, and computer data, including much of the traffic in the Internet.

Figure 1.14 depicts a small portion of the global telecommunications network. In POTS, represented at the upper right, a **twisted pair** of copper wires connects a central office to a home telephone. The twisted pair is called the **local loop** or **subscriber line**. At the central office, the twisted pair is connected to a **line card**, which usually converts the signal from the telephone immediately into digital form. The line card, in turn, is connected to a **switch**, which routes incoming and outgoing telephone connections.

The digital representation of a voice signal, a sequence of bits, is routed through the telephone network. It is usually combined with other bit sequences,

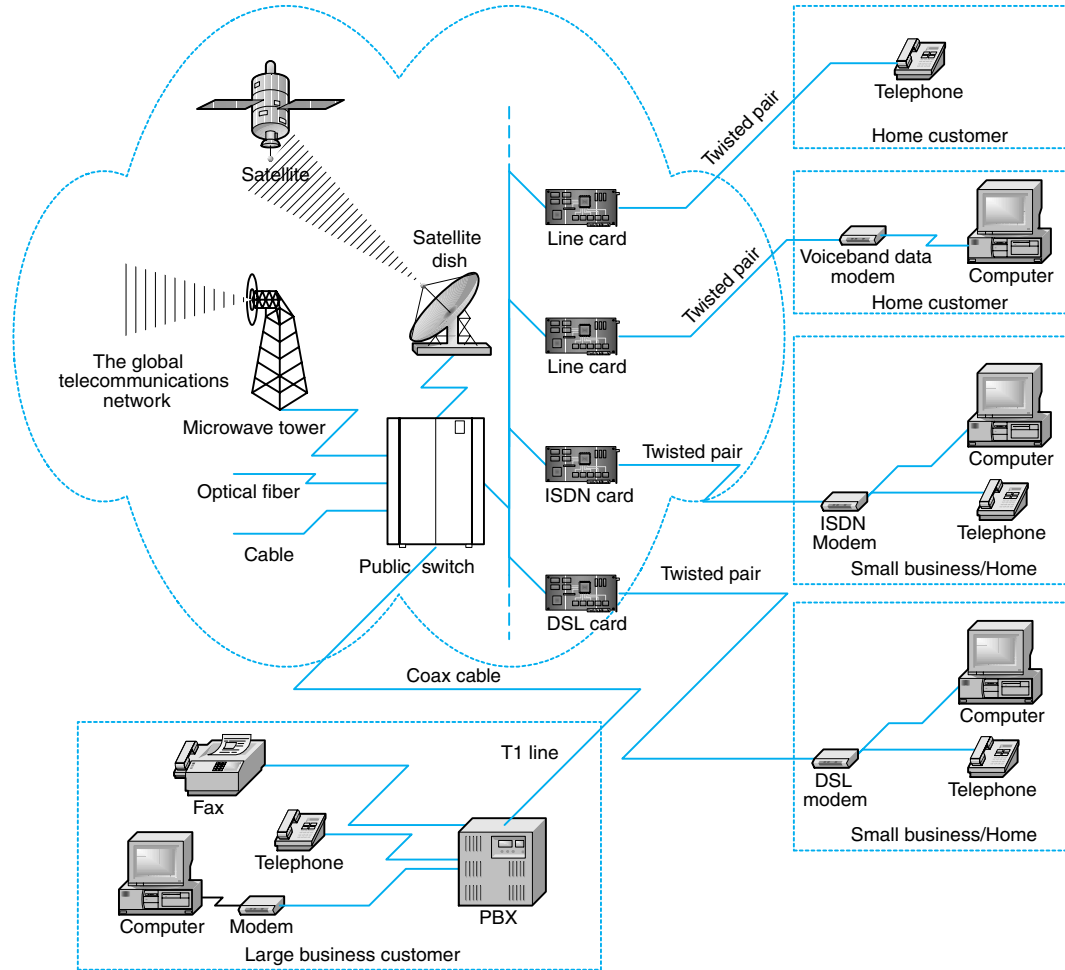


FIGURE 1.14: A portion of the global telecommunications network.

which are other voices or computer data, and sent over high-speed links implemented with optical fiber, microwave radio, coaxial cable, or satellites.

Of course, a telephone conversation usually involves two parties, so the network delivers to the same line card a digital sequence representing the far-end speaker's voice. That digital sequence is decoded and delivered to the telephone via the twisted pair. The line card therefore includes a codec.

The telephone itself, of course, is a system. It transforms the electrical signal that propagates down the twisted pair into a sound signal and transforms a local sound signal into an electrical signal that can propagate down the twisted pair.

POTS can be abstracted as shown in figure 1.15. The entire network is reduced to a model that accepts an electrical representation of a voice signal

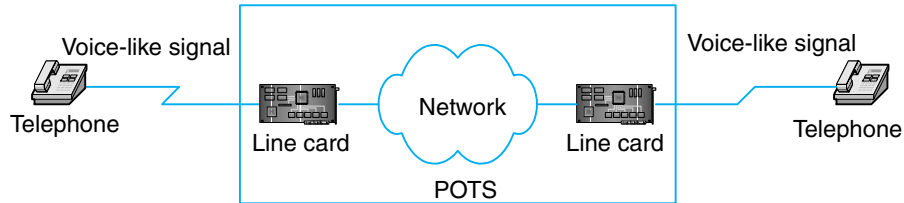


FIGURE 1.15: Abstraction of plain old telephone service (POTS).

and transports it to a remote telephone. In this abstraction, the digital nature of the telephone network is irrelevant. The system simply transports (and degrades somewhat) a voice signal.

Wireless communication

PROBING FURTHER

The telephone network has been freeing itself of its dependence on wires. **Cellular telephones**, which came into widespread use in the 1990s, use radio waves to connect a small, hand-held telephone to a nearby **base station**. The base station connects directly to the telephone network.

There are three major challenges in the design of cellular networks. First, radio spectrum is scarce. Frequencies are allocated by regulatory bodies, often constrained by international treaties. Finding frequencies for new technologies is difficult. Thus, wireless communication devices have to be efficient in their use of the available frequencies. Second, the power available to drive a cellular phone is limited. Cellular phones must operate for reasonably long periods of time with the aid of only small batteries that fit easily within the handset. Although battery technology has been improving, the low power that these batteries can deliver severely limits the range of a cellular phone (how far it can be from a base station) and the processing complexity (the microprocessors in a cellular phone consume considerable power). Third, networking is complicated. In order to be able to route telephone calls to a cellular phone, the network needs to know where the phone is (or, more specifically, which base station is closest). Moreover, the network needs to support phone calls in moving vehicles, which implies that a phone may move out of range of one base station and into the range of another during the course of a telephone call. The network must **hand off** the call seamlessly.

Although “radio telephones” have existed for a long time, particularly for maritime applications in which wireline telephony is impossible, it was the cellular concept that made it possible to offer radio telephony to large numbers of users. The concept is simple. Radio waves propagating along the surface of the earth lose power approximately proportionally to the inverse of the fourth power of distance. In other words, if at distance d meters from a transmitter you receive w watts of radio power, then at distance $2d$ you will receive approximately $w/2^4 = w/16$ watts of radio power. This fourth-power propagation loss was traditionally considered to be only a hindrance to wireless communication. It had to be overcome by greatly boosting the transmitted power. The cellular concept turns this

continued on next page

**PROBING
FURTHER**

hindrance into an advantage: Because the loss is so high, beyond a certain distance the same frequencies can be reused without significant interference. Thus, the service area is divided into cells. A second benefit of the cellular concept is that, at least in urban areas, a cellular phone is never far from a base station. Thus, it does not need to transmit a high-power radio signal to reach a base station. This makes it possible to operate on a small battery.

**PROBING
FURTHER***LEO telephony*

Ideally, a cellular phone, with its one phone number, could be called anywhere in the world, wherever it happens to be, without the caller's needing to know where it is. The technological and organizational infrastructure is evolving to make this possible. When a phone "roams" out of its primary service area, it negotiates with the local service provider in a new area for service. If that service provider has a business agreement with the customer's main service provider, then it provides service. This requires complex networking so that telephone calls to the customer are routed to the correct locale.

However, digital cellular service, particularly in the United States, remains spotty; many rural areas are not served. Providing such service by installing base stations is expensive. Moreover, maritime service away from coastlines is technically impossible with cellular technology.

One candidate technology for truly global telephony services is based on low-earth-orbit (**LEO**) satellites. One such project (now bankrupt) is the Iridium project, spearheaded by Motorola, and so named because in the initial conception, there would be 77 satellites. The Iridium atom has 77 electrons. The idea is that enough satellites are put into orbit so that one is always near enough to communicate with a hand-held telephone. When the orbit is low enough so that a hand-held telephone can reach the satellite (a few hundred kilometers above the surface of the earth), the satellites move by fairly quickly. As a consequence, during the course of a telephone conversation, the connection may have to be handed off from one satellite to another. In addition, in order to be able to serve enough users simultaneously, each satellite has to reuse frequencies according to the cellular concept. To do that, it focuses multiple beams on the surface of the earth, using multielement antenna arrays.

As of this writing, this approach has not yet proved economically viable. The investment already has been huge, with at least one high-profile bankruptcy to date, and so the risks are high. Better networking of terrestrial cellular services may provide formidable competition, particularly as service improves to rural areas. The LEO approach, however, has one advantage that terrestrial services cannot hope to match anytime soon: truly worldwide service. The satellites provide service essentially everywhere, even in remote wilderness areas and at sea.

Dual-tone, multifrequency

Even in POTS, not all the information transported is voice. At a minimum, the telephone needs to be able to convey to the central office a telephone number in order to establish a connection. A telephone number is not a voice signal. It is intrinsically discrete. Because the system is designed to carry voice signals, one option is to convert the telephone number into a voice-like signal. A system is needed with the structure shown in figure 1.16. The block labeled dual-tone multifrequency (DTMF) is a system that transforms a sequence of numbers (coming from the keypad on the left) into a voice-like signal.

The DTMF standard provides precisely such a mechanism. As indicated at the left in figure 1.16, when the customer pushes one of the buttons on the telephone keypad, a sound that is the sum of two sinusoidal signals is generated. The frequencies of the two sinusoids are given by the row and column of the key. For example, a “0” is represented as a sum of two sinusoids with frequencies 941 Hz and 1,336 Hz. The waveform for such a sound is shown in figure 1.17. The line card in the central office measures these frequencies to determine which digit was dialed.

Modems

Because POTS is ubiquitous, it is attractive to find a way for it to carry computer data. Like the numbers on a keypad, computer data are intrinsically discrete. Computer data are represented by bit sequences, which are functions of the form

$$\text{BitSequence: Indices} \rightarrow \text{Binary},$$

where $\text{Indices} \subset \text{Naturals}$, the natural numbers, and $\text{Binary} = \{0, 1\}$. Like keypad numbers, in order for a bit sequence to traverse a POTS phone line, it has to be transformed into something that resembles a voice signal. Furthermore, a

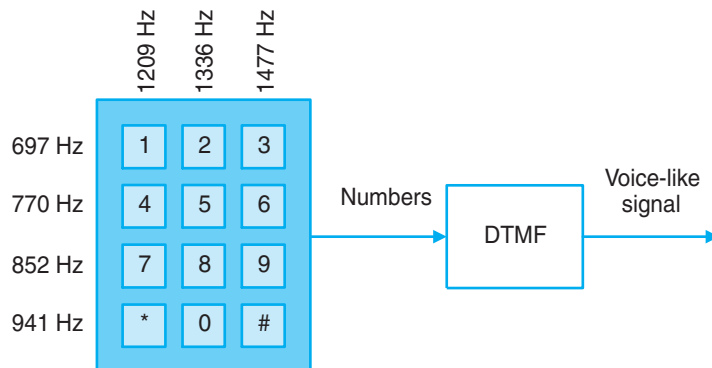


FIGURE 1.16: DTMF system converts numbers from a keypad into a voice-like signal.

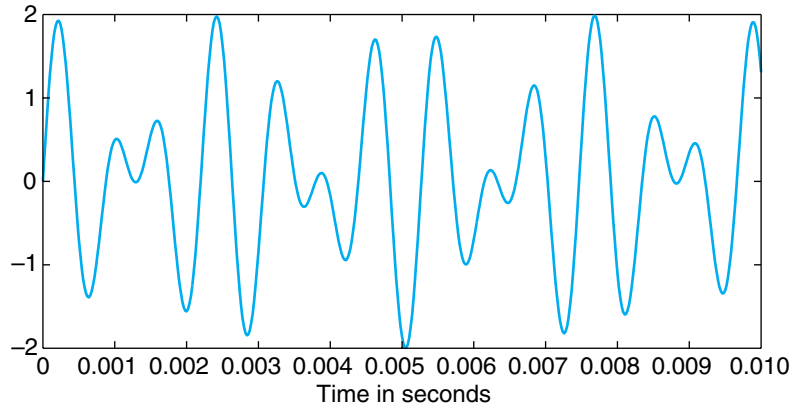


FIGURE 1.17: Waveform representing the “0” key in DTMF system.

system is needed to transform the voice-like signal back into a bit sequence. A system that does that is called a **voiceband data modem**, shown just below the upper right in figure 1.14. The word **modem** is a contraction of “modulator” and “demodulator.” A typical arrangement is shown in figure 1.18.

The voice-like signal created by modern modems does not sound like the discrete tones of DTMF; rather it sounds more like hissing. This is a direct conse-

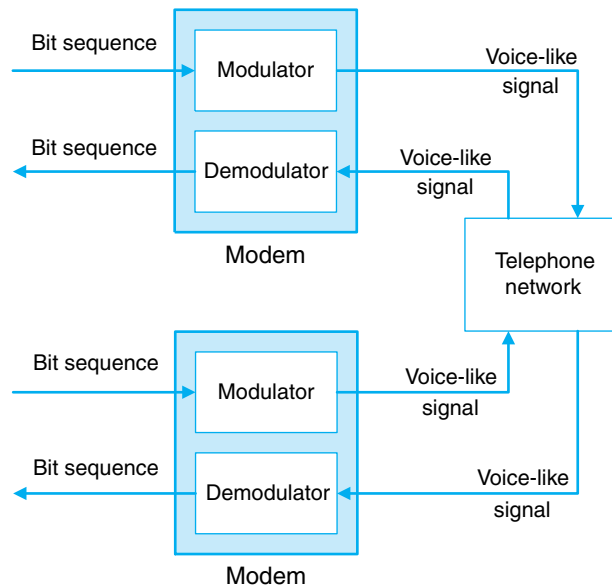


FIGURE 1.18: Voiceband data modems.

quence of the fact that modern modems carry much more data per second than DTMF can (up to 54,000 bits per second rather than just a few digits per second).

Most line cards involved in POTS convert the voice signal into a digital bit stream at the rate of 64,000 bits per second. This bit stream is then transported by the digital network. A voiceband data modem gains access to the digital network rather indirectly, by first constructing a voice-like signal to send to the line card. This gives the voiceband data modem its universality. It works anywhere because the telephone network is designed to carry voice, and it is making the digital data masquerade as voice.

Digital networks

The first widely available service that gave direct access to the global digital telephone network was the integrated services digital network (**ISDN**). The ISDN service required that a different line card be installed at the central office; it was therefore not as universally available as POTS. In fact, after it was developed in the early 1980s, it took nearly 10 years in the United States for ISDN to become widely installed.

The configuration for ISDN is shown below the voiceband data modem in figure 1.14. It requires a special modem on the customer side as well as a special line card in the central office. ISDN typically provides two channels at rates of 64,000 bits per second plus a third control channel with a rate of 16,000 bits per second. One of the 64 kilobytes per second (kbps) channels can be used for voice while, simultaneously, the other two channels are used for data.

A more modern service is the digital subscriber line (**DSL**). As shown at the lower right in figure 1.14, the configuration is similar to that of ISDN. Specialized modems and line cards are required. Asymmetric DSL (**ADSL**) is a variant that provides an asymmetric bit rate, whereby the rate in the direction from the central office to the customer is much higher than the rate from the customer to the central office. This asymmetry reflects the reality of most Internet applications, in which relatively few data flow from the client and torrents of data (including images and video) flow from the server.

Modems are used for many other channels besides the voiceband channel. Digital transmission over radio, for example, requires that the bit sequence be transformed into a radio signal that conforms with regulatory constraints on that radio channel. Digital transmission over electrical cable requires transforming the bit sequence into a form that propagates well over such cable and that does not radiate too much radiofrequency interference. Digital transmission over optical fiber requires transforming the bit sequence into a light signal, whose intensity is usually modulated at very high rates. At each stage in the telephone network, therefore, a voice signal has a different physical form with properties that are well suited to the medium through which the signal propagates. For example, voice, which in the form of sound travels well only over short distances, is converted to an electrical signal that carries well over copper wires for a few kilometers. Copper wires, however, are not as

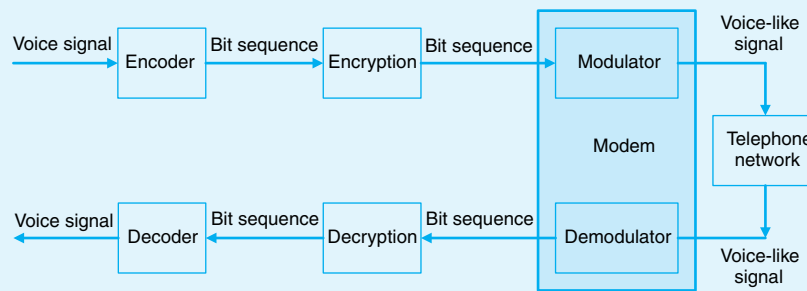
well suited for long distances as optical fiber. Most long-distance communication channels today use optical fiber, although satellites still have certain advantages.

PROBING FURTHER

Encrypted speech

Pairs of modems are used at opposite ends of a telephone connection, each with a transmitter and a receiver, to achieve bidirectional (**full duplex**) communication. Once such modems are in place, and once they have been connected via the telephone network, they function as a bidirectional “bit pipe.” That bit pipe is then usable by other systems, such as a computer.

One of the strangest uses is to transmit digitally represented and encrypted voice signals. Here is a depiction of this relatively complicated arrangement:



What is actually sent through the telephone network sounds like hissing, which by itself provides a modicum of privacy. Casual eavesdroppers are unable to understand the encoded speech. However, this configuration also provides protection against sophisticated listeners. A listener who is able to extract the bit sequence from this sound is nonetheless unable to reconstruct the voice signal because the bit sequence is encrypted.

Only one end is shown. The encoder and decoder, which convert voice signals to and from bit sequences, are fairly sophisticated systems, as are the encryption and decryption systems. The fact that such an approach is cost effective has more to do with economics and sociology than with technology.

Signal degradation

A voice received via the telephone network is different from the original in several ways. These differences can be modeled by a system that degrades the voice signal.

First, there is a loss of information because of sampling and quantization in the encoder, as discussed in section 1.1.6. Moreover, the media that carry the signal, such as the twisted pair, are not perfect; they distort the signal. One cause of

distortion is addition of **noise** to the signal. Noise, by definition, is any undesired component in the signal. Noise in the telephone network is sometimes audible as background hissing, or as **crosstalk**—that is, leakage from other telephone channels into your own. Another degradation results because the medium attenuates the signal, and this attenuation depends on the signal frequency. The line card, in particular, usually contains a **bandlimiting filter** that discards the high-frequency components of the signal. This is why telephone channels do not transport music well. Finally, the signal propagates over a physical medium at a finite speed, bounded by the speed of light, and so there is a delay between the time you say something and the time when the person at the other end hears what you say. Light travels through 1 kilometer (km) of optical fiber in approximately $5 \mu\text{s}$, and so the 5,000 km between Berkeley and New York causes a delay of about 25 milliseconds (ms); this delay, however, is not easily perceptible.*

Communications engineering is concerned with how to minimize the degradation for all kinds of communication systems, including radio, television, cellular phones, and computer networks (such as the Internet).

1.2.3 Audio storage and retrieval

We have seen how audio signals can be represented as sequences of numbers. Digital audio storage and retrieval are all about finding a physical and persistent representation for these numbers. These numbers can be converted into a single sequence of bits (binary digits) and then “printed” onto some physical medium from which they can later be read back. The transformation of sound into its persistent representation can be modeled as a system, as can the reverse or playback process.

Example 1.12: In the case of compact discs (CDs), the physical medium is a layer of aluminum on a platter into which tiny pits are etched. In the playback device, a laser aimed at the platter uses an interference pattern to determine whether a pit exists at a particular point in the platter. These pits thus naturally represent binary digits, inasmuch as they can exist in two states (present or not present).

*A phone conversation relayed by satellite has a much larger delay. Most satellites traditionally used in the telecommunications network are **geosynchronous**, which means that they hover at the same point over the surface of the earth. To do that, they have to orbit at a height of 22,300 miles, or about 35,900 km. A radio signal takes about 120 ms to traverse that distance; because a signal has to go up and back, there is an end-to-end delay of at least 240 ms (which does not include delays in the electronics). If you are using this channel for a telephone conversation, the round-trip time from when you say something to when you get a reaction is a minimum of 480 ms. This delay can be quite annoying, impeding your ability to converse until you get used to it. If you use Internet telephony, the delays are even longer, and can be irregular, depending on how congested the Internet is when you call.

Whereas a voiceband data modem converts bit sequences into voice-like signals, a musical recording studio does the reverse, creating a representation of the sound that is a bit sequence:

RecordingStudio: Sounds \rightarrow BitStreams.

There is a great deal of engineering in the details, however. For instance, CDs are vulnerable to surface defects, which may arise in manufacturing or from hands-on use. These defects may obscure some of the pits or fool the reading laser into detecting a pit where there is none. To guard against this, a very clever error-correcting code called a Reed-Solomon code is used. The coding process can be viewed as a function,

Encoding: BitStreams \rightarrow RedundantBitStreams,

where *RedundantBitStreams* \subset *BitStreams* is the set of all possible encoded bit sequences. These bit sequences are redundant, in that they contain more bits than are necessary to represent the original bit sequence. The extra bits are used to detect errors and, sometimes, to correct them. Of course, if the surface of the CD is too badly damaged, even this clever scheme fails, and the audio data are not recoverable.

CDs also contain **metadata**, which is extra information about the audio signal. This information allows the CD player to identify the start of a musical number, its length, and sometimes the title and the artist.

The CD format can also be used to contain purely digital data. Such a CD is called a **CD-ROM** (read-only memory). It is called this because, like a computer memory, it contains digital information. Unlike the data in a computer memory, however, that information cannot be modified.

Digital video discs (DVDs) take this concept much further, including much more metadata. They may eventually replace CDs. They are entirely compatible, in that they can contain exactly the same audio data that a CD can. DVD players can play CDs; however, CD players cannot play DVDs. DVDs can also contain digital video information and, in fact, any other digital data.

Digital audio tape (DAT) is also a competitor of CDs but has not captured much of a market. \square

1.2.4 *Modem negotiation*

A very different kind of system is the one that manages the establishment of a connection between two voiceband data modems. These two modems are at physically different locations, are probably produced by different manufacturers, and possibly use different communication standards. Both modems convert bit streams to and from voice-like signals, but other than that, they do not have much in common.

When a connection is established through the telephone network, the answering modem emits a tone that announces, “I am a modem.” The initiating modem listens for this tone and, if it fails to detect it, assumes that no connection can be established and hangs up. If it does detect the tone, it answers with a voice-like signal that announces, “I am a modem that can communicate according to ITU standard x ,” x being one of the many modem standards published by the **International Telecommunication Union (ITU)**.

The answering modem may or may not recognize the signal from the initiating modem. The initiating modem, for example, may be a newer modem that operates under a standard that was established after the answering modem was manufactured. If the answering modem does recognize the signal, it responds with a signal that says, “Good, I too can communicate using standard x , so let’s get started.” Otherwise, it remains silent. If the initiating modem fails to get a response, it tries another signal, announcing, “I am a modem that can communicate according to ITU standard y ,” y being typically now an older (and slower) standard. This process continues until the two modems agree on a standard.

Once agreement is reached, the modems need to make measurements of the telephone channel to compensate for its distortion. They do this by sending each other preagreed signals called **training signals**, defined by the standard. The training signal is distorted by the channel, and, because the receiving modem knows the signal, it can measure the distortion. It uses this measurement to set up a device called an **adaptive equalizer**. Once both modems have completed their setup, they begin to send data to one another.

As systems go, modem negotiation is fairly complex. It involves both event sequences and voice-like signals. The voice like signals need to be analyzed in fairly sophisticated ways, sometimes producing events in the event sequences. It takes this entire book to analyze all parts of this system. The handling of the event sequences is treated through the use of finite-state machines, and the handling of the voice-like signals is treated through the use of frequency-domain concepts and filtering.

1.2.5 Feedback control systems

Feedback control systems are composite systems in which a **plant** embodies a physical process whose behavior is guided by a control signal. A plant may be, for example, a mechanical device, such as the power train of a car; a chemical process; or an aircraft with certain inertial and aerodynamic properties. Sensors attached to the plant produce signals that are fed to the controller, which then generates the control signal. This arrangement, whereby the plant feeds the controller and the controller feeds the plant, is a complicated sort of composite system called a **feedback control system**. It has extremely interesting properties, which we explore in much more depth in subsequent chapters.

In this section, we construct a model of a feedback control system by using the syntax of block diagrams. The model consists of several interconnected

components. We identify the input and output signals of each component and how the components are interconnected, and we argue on the basis of common-sense physics how the overall system will behave.

Example 1.13: Consider a forced-air heating system, which heats a room in a home or office to a desired temperature. Our first task is to identify the individual components of the heating system. These are

- a furnace/blower unit (which we will simply call the heater) that heats air and blows the hot air through vents into a room,
- a temperature sensor that measures the temperature in a room, and
- the control system that compares the specified desired temperature with the sensed temperature and turns the furnace/blower unit on or off, depending on whether the sensed temperature is below or above the demanded temperature.

The interconnection of these components is shown in figure 1.19.

Our second task is to specify the input and output signals of each component system (the domain and range of the function), ensuring the input–output matching conditions. The heater produces hot air if it is turned on. Therefore, its input signal is simply a function of time that takes one of two values, *On* or *Off*. We call input to the heater (a signal) *OnOff*,

$$\text{OnOff}: \text{Time} \rightarrow \{\text{On}, \text{Off}\},$$

and we take $\text{Time} = \text{Reals}_+$, the nonnegative real numbers. Thus, the input signal space is

$$\text{OnOffProfiles} = [\text{Reals}_+ \rightarrow \{\text{On}, \text{Off}\}].$$

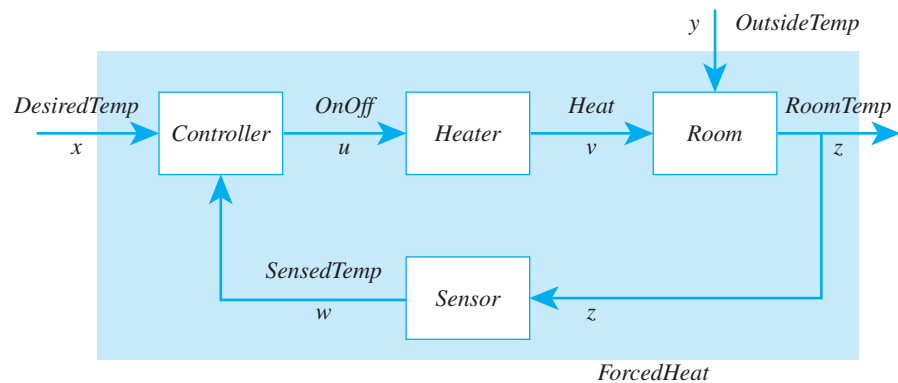


FIGURE 1.19: The interconnected components of a forced-air heating system.

(Recall that the notation $[D \rightarrow R]$ defines a function space, as explained in section 1.2.1.) When the heater is turned on, it produces heat at some rate that depends on the capacity of the furnace and blower. We measure this heating rate in British thermal units (BTUs) per hour. Thus, the output signal of the heater, which we name *Heat*, is of the form

$$\text{Heat}: \text{Reals}_+ \rightarrow \{0, B_c\},$$

where B_c is the heater capacity measured in BTUs per hour. If we name the output signal space *HeatProfiles*, then

$$\text{HeatProfiles} = [\text{Reals}_+ \rightarrow \{0, B_c\}].$$

Thus the *Heater* system is described by a function,

$$\text{Heater}: \text{OnOffProfiles} \rightarrow \text{HeatProfiles}. \quad (1.13)$$

Commonsense physics tells us that when the heater is turned on, the room will begin to warm up, and when the heater is turned off, the room temperature will fall until it reaches the outside temperature. The room temperature therefore depends on both the heat delivered by the heater and the outside temperature. Thus the input signal to the room is the pair (*Heat*, *OutsideTemp*). We can take *OutsideTemp* to be of the form

$$\text{OutsideTemp}: \text{Reals}_+ \rightarrow [\min, \max],$$

where $[\min, \max]$ is the range of possible outside temperatures, measured, say, in degrees Celsius. The output signal of the room is of course the room temperature,

$$\text{RoomTemp}: \text{Reals}_+ \rightarrow [\min, \max].$$

If we denote

$$\text{OutsideTempProfiles} = [\text{Reals}_+ \rightarrow [\min, \max]]$$

and

$$\text{RoomTempProfiles} = [\text{Reals}_+ \rightarrow [\min, \max]],$$

then the behavior of the *Room* system is described by the function

$$\text{Room}: \text{HeatProfiles} \times \text{OutsideTempProfiles} \rightarrow \text{RoomTempProfiles}. \quad (1.14)$$

In a similar manner, the *Sensor* system is described by the function

$$\text{Sensor: RoomTempProfiles} \rightarrow \text{SensedTempProfiles}, \quad (1.15)$$

with input signal space *RoomTempProfiles* and output signal space

$$\text{SensedTempProfiles} = [\text{Reals}_+ \rightarrow [\min, \max]].$$

The *Controller* is described by the function

$$\begin{aligned} \text{Controller: DesiredTempProfile} \times \\ \text{SensedTempProfile} \rightarrow \text{OnOffProfile}, \end{aligned} \quad (1.16)$$

where

$$\text{DesiredTempProfiles} = [\text{Reals}_+ \rightarrow [\min, \max]].$$

We have constructed a model in which every output drives a compatible input.

The overall forced-air heating system (the shaded part of figure 1.19) has a pair of input signals (desired temperature and outside temperature) and one output signal (room temperature). Therefore, it is described by the function

$$\begin{aligned} \text{ForcedHeat: DesiredTempProfiles} \times \\ \text{OutsideTempProfiles} \rightarrow \text{RoomTempProfiles}. \end{aligned}$$

If we are given the input signal value x of desired temperature and the value y of outside temperature, we can compute the value $z = \text{ForcedHeat}(x, y)$ by solving the following four simultaneous equations:

$$\begin{aligned} u &= \text{Controller}(x, w) \\ v &= \text{Heater}(u) \\ z &= \text{Room}(y, v) \\ w &= \text{Sensor}(z) \end{aligned} \quad (1.17)$$

Given x and y , we must solve these four equations to determine the four unknown functions u , v , w , and z of which u , v , and w are the internal signals and z is the output signal. Of course, to solve these simultaneous equations, we need to specify the four system functions. So far we have simply given names to those functions and identified their domain and

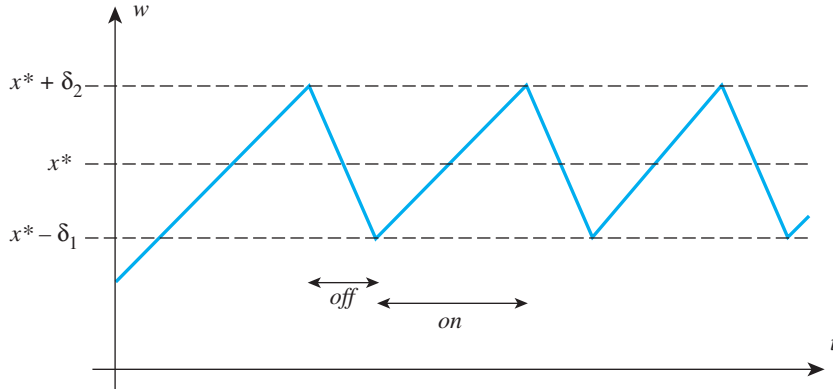


FIGURE 1.20: With a thermostatic controller, the room temperature fluctuates around the desired temperature setting, x^* .

range. To complete the specification, we must describe how those functions assign output signals to input signals.

If the sensor is functioning properly, we expect *Sensor's* output signal to be the room temperature; that is, for all z and for all $t \in \text{Reals}_+$,

$$w(t) = \text{Sensor}(z)(t) = z(t).$$

A thermostatic controller has a simple behavior: It turns the heater on if the sensed temperature falls below the desired temperature by a certain amount, say δ_1 , and it turns the heater off if the sensed temperature rises above the desired temperature by, say, δ_2 . Thus, for all x, w and for all $t \in \text{Reals}_+$,

$$u(t) = \text{Controller}(x, w)(t) = \begin{cases} \text{On}, & \text{if } w(t) - x(t) \leq -\delta_1 \\ \text{Off}, & \text{if } w(t) - x(t) \geq \delta_2. \end{cases}$$

Suppose, finally, that the desired temperature is set to some constant—say, x^* ; therefore, for all $t \in \text{Reals}_+$,

$$x(t) = x^*.$$

We can expect the behavior depicted in figure 1.20. When $x^* - w(t)$ drops below $-\delta_1$, the controller turns on the heater, the room temperature increases until $x^* - w(t)$ rises above δ_2 , and then the controller turns off the heater. Thus, the room temperature fluctuates around the desired temperature, x^* . \square

1.3 Summary



Signals are functions that represent information. We studied examples of three classes of signals. The first class consists of functions of discrete or continuous time and space that occur in human perception and eletromechanical sensors. The second class consists of functions of time and space representing attributes of physical objects or devices. The third class consists of sequences of symbols representing data or the occurrences of events. In each class, the domain and the range can be defined precisely.

Systems are functions that transform signals. We studied telecommunication systems, in which a network that was originally designed for carrying voice signals is used for many other kinds of signals today. One way to accomplish this is to design systems such as modems that transform signals so that they masquerade as voice-like signals. We also examined system models for signal degradation and for storage of signals. We studied systems that are concerned primarily with discrete events and command sequences, and we examined a feedback control system. Both the telephone system and the forced-air heating system were described with the use of block diagrams as interconnections of simpler component systems. In all cases, systems were given as functions in which the domain and the range are function spaces, or sets of functions.

EXERCISES

Each problem is annotated with the letter **E** (exercise), **T** (requires some thought), or **C** (requires some conceptualization). Problems labeled **E** are usually mechanical; those labeled **T** require a plan of attack; and those labeled **C** usually have more than one defensible answer.

- E** 1. The function $x : \text{Reals} \rightarrow \text{Reals}$ given by

$$\forall t \in \text{Reals}, \quad x(t) = \sin(2\pi \times 440t)$$

is a mathematical example of a signal in the signal space $[\text{Reals} \rightarrow \text{Reals}]$. Give a mathematical example of a signal x in each of the following signal spaces.

- $[\text{Integers} \rightarrow \text{Reals}]$
- $[\text{Reals} \rightarrow \text{Reals}^2]$
- $[\{0, 1, \dots, 600\} \times \{0, 1, \dots, 400\} \rightarrow \{0, 1, \dots, 255\}]$
- Describe a practical application for the signal space $[\{0, 1, \dots, 600\} \times \{0, 1, \dots, 400\} \rightarrow \{0, 1, \dots, 255\}]$; that is, what might a function in this space represent?

- C** 2. For each of the continuous-time signals below, represent the signal in the form of $f: X \rightarrow Y$ and as a sketch like figure 1.1. Carefully identify the range and domain in each case.
- The voltage across the terminals of a car battery.
 - The closing prices on each day of a share of a company.
 - The position of a moving vehicle on a straight one-lane road of length L .
 - The simultaneous position of two moving vehicles on the same straight one-lane road of length L .
 - The sound heard in both of your ears.
- E** 3. In digital telephony, voice is sampled at a rate of 8,000 samples/second; therefore, the sampling period is $1/8,000 = 125$ microseconds. What are the sampling period and the sampling frequency of sound in a compact disc (CD)?
- E** 4. Figure 1.4 displays the plots of two sinusoidal signals and their sum. Sketch by hand the plots of the four functions *Step*, *Triangle*, *Sum*, and *Diff*, all with domain $[-1, 1]$ and range *Reals*, defined by, $\forall t \in [-1, 1]$,

$$\text{Triangle}(t) = 1 - |t|,$$

$$\text{Step}(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0, \end{cases}$$

$$\text{Sum}(t) = \text{Triangle}(t) + \text{Step}(t),$$

$$\text{Diff}(t) = \text{Triangle}(t) - \text{Step}(t).$$

- C** 5. The following examples of spatial information can be represented as a signal in the form of $f: X \rightarrow Y$. Specify a reasonable choice for the range and domain in each case.
- An image impressed on photographic paper.
 - An image stored in computer memory.
 - The height of points on the surface of the earth.
 - The location of the chairs in a room.
 - The household voltage in Europe, which has frequency 50 Hz and is 210 volts, RMS.
- C** 6. The image called *Albers* consists of an eight-inch yellow square in the center of a white 12-inch square background. Express *Albers* as a function by choosing the domain, range, and function assignment.

- E** 7. How many bits are there in a $768 \times 1,024$ pixel image in which each pixel is represented as a 16-bit word? How long would it take to download this image over a 28-kbps voiceband modem, a 384-kbps DSL modem, and a 10 Megabits per second (Mbps) Ethernet local area network?
- C** 8. Represent these examples as data or event sequences. Specify reasonable choices for the range and domain in each case.
- The result of 100 tosses of a coin.
 - The sequence of button presses inside an elevator.
 - The sequence of main events in a soda vending machine.
 - Your response to a motorist who is asking directions.
 - A play-by-play account of a game of chess.
- C** 9. Formulate the following items of information as functions. Specify reasonable choices for the domain and range in each case.
- The population of U.S. cities.
 - The white pages in a phone book (careful: the white pages may list two identical names and may list the same phone number under two different names).
 - The birth dates of students in class.
 - The broadcast frequencies of AM radio stations.
 - The broadcast frequencies of FM radio stations (look at your radio dial or at the web page:
<http://www.eecs.berkeley.edu/~eal/eecs20/sidebars/radio/index.html>).
- E** 10. Use MATLAB to plot the graph of the following continuous-time functions defined over $[-1, 1]$, and on the same plot display 11 uniformly spaced samples (0.2 seconds apart) of these functions. Are these samples good representations of the waveforms?
- $f: [-1, 1] \rightarrow \text{Reals}$, where for all $x \in [-1, 1]$, $f(x) = e^{-x} \sin(10\pi x)$.
 - Chirp*: $[-1, 1] \rightarrow \text{Reals}$, where for all $t \in [-1, 1]$, $\text{Chirp}(t) = \cos(10\pi t^2)$.
- E** 11. Suppose the pendulum of figure 1.10 is rotating counterclockwise at a speed of one revolution per second over the five-second interval $[0, 5]$. Sketch a plot of the resulting function: $\theta: [0, 5] \rightarrow [-\pi, \pi]$. Assume $\theta(0) = 0$. Also specify this function mathematically. Your plot is discontinuous, but the pendulum's motion is continuous. Explain this apparent inconsistency.

- T 12. There is a large difference between the sets X , Y , and $[X \rightarrow Y]$. This exercise explores some of that difference.
- (a) Suppose $X = \{a, b, c\}$ and $Y = \{0, 1\}$. List all the functions from X to Y —that is, all the elements of $[X \rightarrow Y]$. Note that part of the problem here is to figure out *how* to list all the functions.
 - (b) If X has m elements and Y has n elements, how many elements does $[X \rightarrow Y]$ have?
 - (c) Suppose

$$\begin{aligned} \text{ColorMapImages} = & [\text{DiscreteVerticalSpace} \times \text{DiscreteHorizontalSpace} \\ & \rightarrow \text{ColorMapIndexes}]. \end{aligned}$$

Suppose the domain of each image in this set has 6,000 pixels and the range has 256 values. How many distinct images are there? Give an approximate answer in the form of 10^n . Hint: $a^b = 10^{b \log_{10}(a)}$.