CHAPTER 14

An Introduction to Digital Signals and Systems

All of the systems we have discussed so far have involved real-world signals involving pressure changes (like speech sounds) or movements (like those of the middle ear bones) or changes in voltage (like the wave fed to a loudspeaker). As we will discuss further below, such signals are, as are all signals we directly experience, *analogue* signals. Perhaps surprisingly then, almost all the equipment we use to record, store, analyze and reproduce these signals is no longer analogue, but *digital*. In fact, as a result of technological advances in the last 20 or so years, digital signals and systems have almost entirely superseded analogue systems in most applications. Perhaps the best-known example of this is the fact that almost all music is now recorded and reproduced digitally.

As a result of these developments, anyone in the fields of speech and hearing (even those not involved in research) is guaranteed to meet digitally based equipment in the course of their work. For example, any work done with a computer involves digital signal processing although this may not always be apparent to the user. The intelligent use of such equipment necessitates at least a rudimentary understanding of its working principles. Unfortunately, it would take at least another whole book for a proper exploration of the properties of digital signals and systems! Our goals here are more limited—to acquaint you with the basics of this pervasive and rapidly expanding field, without going into too much detail. In any case, as you'll see, many of the principles that we have already developed apply in a straightforward way to digital signals and systems, although often in a slightly modified form.

Pros and cons of digital techniques

Before beginning, however, it is well to keep in mind the main advantages of digital techniques. First, regarding signals, is the possibility of making essentially perfect copies of an original.

As you may know, if you take an ordinary audio-tape recording (as made, for example, using the cassette recorder described in Chapter 4), and make a copy of it on another tape recorder, the copy, no matter how good the equipment, isn't quite the same as the original. If you copied that copy, things would be even worse. Copying the previous copy in a chain in this way will eventually lead to an absolutely useless reproduction. This is not so with digital signals—it is possible to make each copy completely equivalent to the original. This is a major attraction of compact discs which store signals digitally—the signals coming out of your CD system should be as good as those on the studio master. A related property is that digital signals are much more resistant to noise and degradation when transmitted (as in, for example, telephones and televisions).

The second primary advantage concerns systems. Digital systems can be extremely flexible and allow the routine use of processing schemes that would be impossible to perform with analogue systems.

What is an analogue signal?

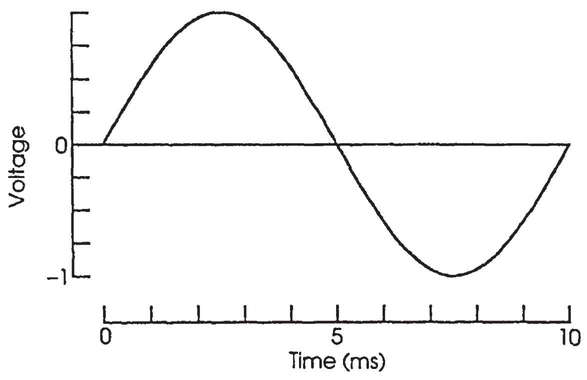
We've been bandying this word 'digital' about without much care, counting on you having a vague understanding of it. It's time now to define this term more precisely. The best way we can do that is by first giving a name to the kinds of signals that are *not* digital—*analogue* signals.

In fact, analogue signals are the only ones dealt with in this book so far. We didn't need to make this explicit, because there were no other kinds of signals to contrast them with. So, for example, speech sounds or musical tones, or the displacements of the middle ear bones, are all examples of analogue signals.

There are two crucial aspects of analogue signals which distinguish them from digital signals. First, they are continuous in time. This is another way of saying that at any particular moment of time that you tried to measure a signal, it would have *some* value. Between any two given moments, then, there are always an infinite number of instants at which the signal exists. Thus, you cannot write down all the values of analogue signals in a table (no matter how big the table or short the signal) because an infinite number of entries would be required. You should already be familiar with this limitation on the use of tables to express essentially continuous information from Chapter 6. There we showed why a graph could be better than a table in representing an amplitude response appropriately.

Not only are analogue signals continuous in *time* (normally plotted as the *x*-axis), but they are also continuous in *amplitude* (usually on the y-axis). This is another way of saying that, at a particular moment in time, the signal can have any amplitude—it is not restricted to some set of values. So, for example, an electrical signal coming from a microphone at a particular moment could have the value of 6.232071 V, or 1.21440633 V, or 0.1016173263926 V. The point is that whenever we make a measurement (here with a voltmeter), we round off to however many digits our equipment handles. But the signal itself could have any value, and so, in general, we would need an infinite number of digits to represent its true value.

Let's illustrate these two points by thinking about the difficulties of writing down (in a table) the values making up a single period of an analogue sine wave of frequency 100 Hz and peak amplitude 1 V:



Consider first the problem of writing down any single value with a fixed number of digits. Although this is easily done for, say, time values of 0, 5 and 10 ms, where we know that the amplitude of the sine wave is 0, what about at a time of 1.25 ms? As 1.25 ms is 1/8th of the total period of 10 ms, and there are 360° in each period, we need to take the sine of 360/8. This is equivalent to the sine of 45° (=360° × [1.25 ms/10 ms]) which a calculator says is 0.7071068. As good an approximation as this might be, it is not the exact value. The sine of 45° is actually $\sqrt{2}/2$, an irrational number which, as with all such numbers, takes an infinite number of digits to write down. Nor is this an unusual case. Many (in fact, most!) of the values of the sine wave are irrational.

Now, let's address the problem of the number of entries we would need to specify this waveform (ignoring the problem of the number of digits that we'd need). We'll take a sort of 'Zeno's paradox' approach. Certainly, we'd want to specify the endpoints of the waveform, 0 and 10 ms. Then we'd specify a point halfway between these two, at 5 ms. We could then go on to write down the value of the sinusoid at a time halfway between 0 and the last value picked. This would give us a series of values at 2.5 ms, 1.25 ms, 0.625 ms, 0.3125 ms, 0.15625 ms, and so on. The problem is that this series never ends, so no finite length table could describe the sinusoid completely.

To summarize, analogue signals are continuous in time and amplitude. They exist at every moment, and their amplitude at a particular moment can take on any value. This is why we normally use other kinds of analogue representations to represent them adequately—in our case, graphs. A graph of a signal is also continuous in time and amplitude.

What is a digital signal?

Although a graph can be a complete representation of an analogue *or* digital signal, a table can only be a complete representation of a digital signal. Here, for example, is a table that represents a particular digital signal:

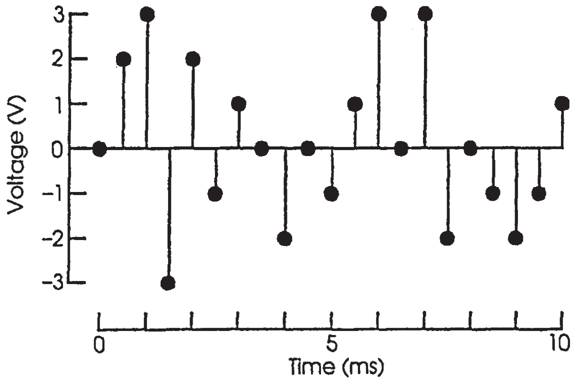
Time (ms)	Amplitude (V)
0.0	0
0.5	2
1.0	3
1.5	–3
2.0	2
2.5	–1
3.0	1
3.5	0
4.0	–2
4.5	0
5.0	–1
5.5	1
6.0	3
6.5	0
7.0	3
7.5	–2
8.0	0
8.5	–1
9.0	–2
9.5	–1
10.0	1

There are two properties of digital signals that make it possible to write them down in a table. First, they exist only at discrete, equally spaced moments of time. So, as here, we might have a value for a signal every ½ ms. *Between* these sample points the signal doesn't really exist—it only has values *at* the sample points. Any signal that has this property is known as a *discrete-time signal*, since it only exists at specific, discrete moments in time.

Second, the amplitude values that the signal can take on are limited to a set of discrete values, here a whole number of volts. This notion should not be new to you—we often restrict the possible values that numbers can take. For example, the interest payable on a loan in France would be rounded to the nearest Euro cent; in the United States or the United Kingdom to the nearest penny. When we measure a space for a shelf, we round to the nearest millimetre or 1/16th of an inch. A signal whose possible values are restricted in this way is said to be *quantized*, as two unequal values must differ by at least a fixed quantum (a single Euro cent, penny or millimetre in the examples above). This is quite different from analogue signals, for which there is no minimum difference between two unequal values.

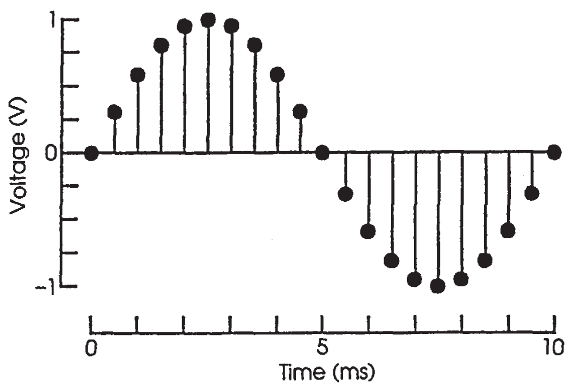
To summarize, there are two reasons why digital signals can be written down in a table and analogue signals cannot. In a given time period, there is only a finite number of sample points in a digital signal; in an analogue signal, there is an infinite number of values that the signal takes on. For a given amplitude range, there is only a finite number of digits that are needed to write down any particular value of a digital signal; the value of an analogue signal needs, in general, an infinite number.

Of course, we could also draw a graph of the digital signal tabulated above:



Note the distinct way in which a digital signal is represented here. In order to emphasize that it exists only at discrete moments, a small circle is drawn for each defined value of the waveform. From this circle is drawn a vertical line to the horizontal axis.

This is only one example of an arbitrary waveform, so let's give a further illustration with a digital version of a signal you are more familiar with—a single period of a digital 100-Hz sine wave. We'll assume an inter-sample time (or *sampling period*) of 0.5 ms and rounding of each amplitude value to three significant digits:



Although we've drawn this as a graph for you to see that there is still a sinusoidal wave shape, the information can also be represented in a table:

Time (ms)	Amplitude (V)
0.0	0.000
0.5	0.309
1.0	0.588
1.5	0.809
2.0	0.951
2.5	1.000
3.0	0.951
3.5	0.809
4.0	0.588
4.5	0.309
5.0	0.000
5.5	−0.309
6.0	−0.588
6.5	−0.809
7.0	−0.951
7.5	−1.000
8.0	−0.951
8.5	−0.809
9.0	−0.588
9.5	−0.309
10.0	−0.000

Because the signal only exists at discrete moments of time, there isn't the problem of infinitely long tables that occurs with analogue signals. Because the values are quantized to a certain number of digits, there isn't the problem of amplitude values that need an infinite number of digits to be written down.

Thinking about the differences between graphs (normally analogue representations) and tables (necessarily digital representations) can give some insight into the relative advantages and disadvantages of digital signals. Imagine that you have a table of numbers you need a copy of. With proper care, it is relatively easy (though laborious) to copy the set of numbers exactly. Tracing a graph, on the other hand, although relatively speedy, would not be nearly as accurate.

Digital systems

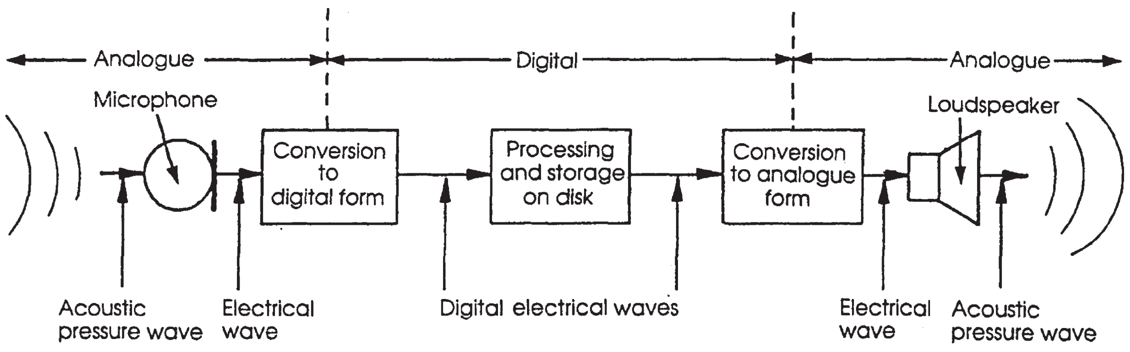
Writing down digital signals in the form of a table, is of course, not the only way they can be represented. But in this form, as a simple list of numbers, they can be easily entered into and hence stored and operated on by a digital computer. Much of the development of digital signal techniques, in fact, was due to the desire to apply the flexibility of general-purpose computers to signal-processing problems.

In this mode, then, computers are used as *digital systems*, the name given to any system that has digital signals for inputs and outputs. In the same way, all the systems we have investigated until now, with analogue signals as input and output, are known as *analogue systems*.

Not all digital systems are general purpose, however, although they are all computers of a sort. For reasons of speed and economy, more and more use is being made of hardware specially constructed for a specific task—a type of computer with a fixed program.

No matter what particular form the digital system takes, special or general purpose, it's relatively rare for a complete system to have digital signals both at the input and output. In other words, we rarely encounter completely digital systems, especially for any task concerned with speech and hearing. We can neither speak digitally nor listen to digital sounds directly. At least one end of the process (and very often both) must be an analogue signal. Typically, practical systems are a series of systems, some digital, some analogue and some which are the go-betweens.

As an example, let's look in a very broad way at the steps involved between a recording of a solo piano player and your listening to a compact disc of it at home:



The starting point is, of course, the sounds created by the pianist as she or he strikes the keyboard. This analogue acoustic signal is first transduced into an electrical signal by a microphone. Although the information is now in a different form (electrical instead of acoustic), it is still analogue. In order to make use of digital techniques of storage and processing, it's necessary to transform the analogue electrical wave into digital form—known as performing an *analogue-to-digital* (usually abbreviated as A-to-D) conversion. Now the information is stored as a series of numbers, and so can be copied and stored with little or no degradation. The acoustic information, represented as a long list of numbers, is coded and pressed into the surface of the compact disc.

In order to listen to the disc, all these steps must be performed in reverse. The compact disc player reads the series of numbers from the disc, but in digital form they are of little use. The digital electrical signal must first be transformed into analogue form by *digital-to-analogue* (usually abbreviated as D-to-A) conversion, before being amplified and then transduced, 'for your listening pleasure', into sound again by a loudspeaker.

This chain of events is fairly common in many systems which are based on digital techniques: (1) the transduction of information into analogue electrical form followed by A-to-D conversion, (2) the processing and/or storage of that information by a digital system and (3) the reconversion into analogue form by D-to-A conversion and final transduction to sound.

Although purely digital systems are fairly rare in speech and hearing, there are many instances in which only D-to-A or A-to-D are performed. For example, in a digital speech synthesizer, the signals originate in a digital form but are converted to analogue signals for output. Computer-based speech analysis systems begin with the analogue speech signal but present the results in digital form.

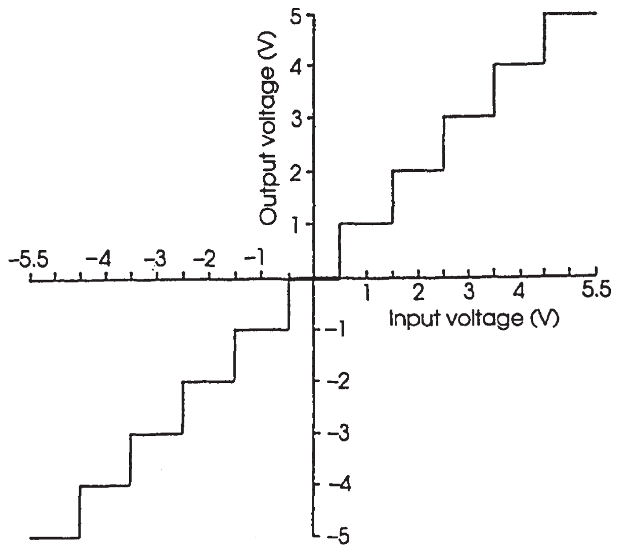
Because we almost always want to do some sort of conversion between analogue and digital signals, it is of utmost importance to know the limitations of these processes. In other words, we want to be sure that in converting from A-to-D form (or vice versa) we don't lose information or add any noise or distortion. We'll see that, as long as some explicit and easily understandable rules are followed, we can convert between the two kinds of representations freely, losing little information. Let's begin with the process of converting an analogue signal into digital form. As digital signals differ from analogue signals in two ways, we consider the process of transformation to consist of two stages: *quantization* followed by *sampling.*

Quantization

Let's suppose that we want to use a digital system to process the 100-Hz analogue sinusoid we looked at above. For convenience, we'll make the peak amplitude 5.5 V instead of 1 V.

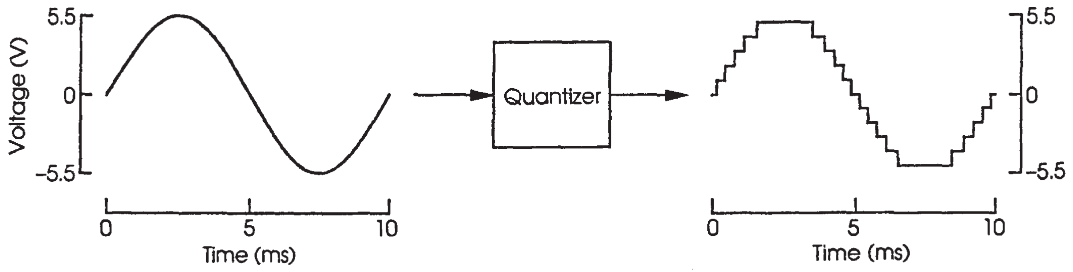
The first step that's necessary is to quantize the signal so that it can only take on a fixed set of amplitude values. In order to do this, we must define two parameters: (1) the maximum voltage levels of the signal to be represented and (2) the number of different levels that are possible. Since the sinusoid goes between +5.5 V and −5.5 V, this will be our defined range. To show the effects of quantization clearly, we'll begin with a small number of possible levels—say, 11 running from −5.5 to +5.5.

How, then, do we transform the amplitude values of the analogue signal, which are continuous, into this digital discrete scale? One way is to use a rule based on rounding to the nearest digit. We simply take the analogue amplitude value and round it to the nearest number of volts. The easiest way to represent this information is in the form of an *input–output* function, similar to those we used in Chapter 11 to portray rectification. The *x*-axis is the instantaneous amplitude value of the input analogue signal, while the *y*-axis gives the instantaneous value of the quantized output:



From this figure, for example, you can see that input voltages between −0.5 and 0.5 V are quantized as 0, while input voltages between −4.5 and −5.5 V are quantized as −5.

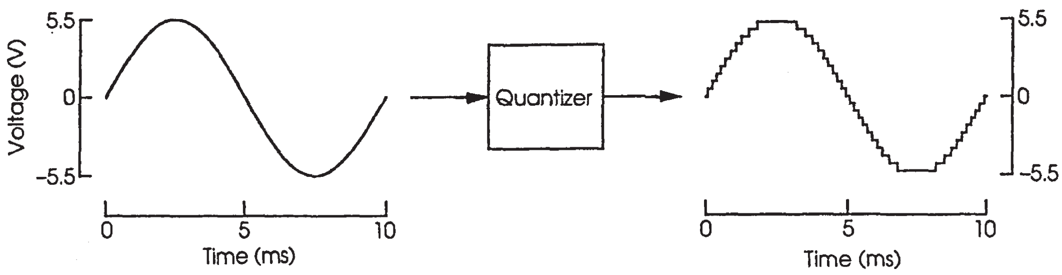
Let's put our 100-Hz sinusoid through this quantizer, and see what comes out:



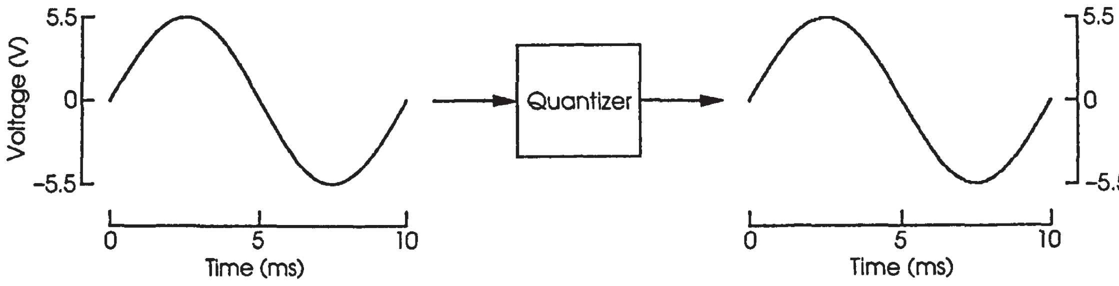
The quantized signal is a sort of sine wave with steps. The amplitude values it takes are, of course, limited to the 11 possible that we defined.

Rarely does a quantizer use as few steps as this, however. If we think of the quantized signal as an approximation to the original analogue signal, then we can get better approximations by increasing the number of available levels.

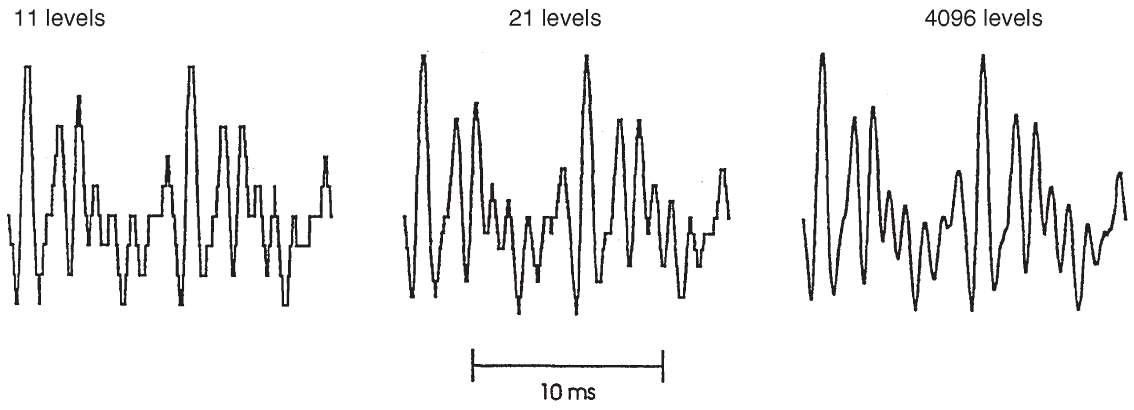
This should be apparent by considering how the resolution (in amplitude) of the quantizer varies with the number of possible levels. Here, 11 levels were used to represent a total voltage range of 11 V (−5.5 to 5.5 V); hence, each level represented a range of levels 1-V wide in the original analogue signal. If 21 levels were possible, each level would only need to represent about 0.52 V, and the quantized sinusoid would look more similar to the analogue original:



Taking a big jump upwards to, say, 111 levels (about 0.1 V per level), the steps in the quantized waveform are barely visible:



Of course, any signal can be quantized, not only sinusoids. Here, for example, are the waveforms of two periods of the vowel /ɑ/, quantised to 11, 21 and 4096 levels.



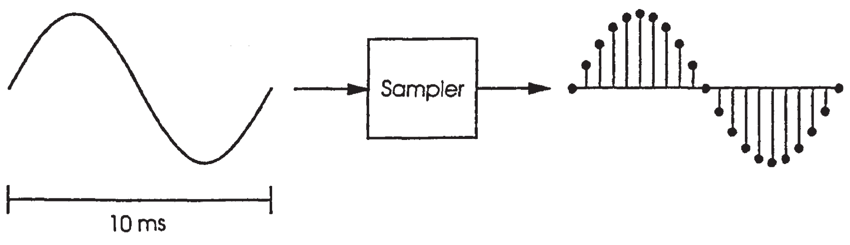
Real-life digital systems almost always use a number of levels that can be expressed as an integer power of two (that is, as 2^n, where n is a whole number) because almost all digital hardware (computers included) express numbers internally in the base 2 or *binary* system. In such a system, it's necessary to specify the maximum number of *bits* or binary digits that are available to represent each amplitude value. A 1-bit system can express two levels, and each extra 'bit' doubles the number of available levels. (In a similar way, a base 10 system has 10 available levels for each digit, and each added digit increases by a factor of 10 the number of available levels.) So, an 8-bit system has 256 levels and a 12-bit system 4096. In fact, this is the way digital systems are normally described—as an *8-bit system*, not a system with 256 quantization levels (although the two mean the same thing).

For high-quality audio applications (as in the compact disc), 14–16 bits are used, but many tasks can be done with as few as 8 or 10 bits. As always, different situations demand different degrees of accuracy. The crucial point is that signals can often be quantized finely enough (that is, with enough levels) to be treated as if *any* amplitude value were possible. In other words, the quantized signal can be considered such a good approximation to the original that the effects of quantization can be ignored. Thus, most of the theoretical developments in digital signal processing (though by no means all) assume an unquantized signal. In the rest of our development, we too shall ignore quantization effects.

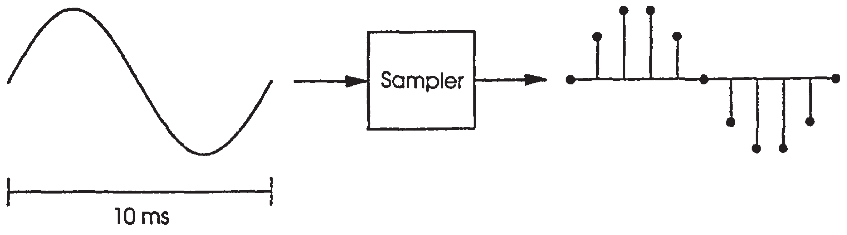
Sampling

We're now halfway to a digital signal. Although we can restrict the amplitude values that an analogue signal takes on with a quantizer, the quantized signal still exists at every moment in time. What we need to do now is restrict to discrete moments the times at which the signal can exist—a process known as *sampling.* Sampling simply measures the amplitude value of the signal at equally spaced moments in time. The most important parameter to define in a sampler is its *sampling frequency* (or equivalently, *sampling rate*)—the number of times per second that the value of the continuous time signal is recorded. The reciprocal of the sampling frequency, which is just the time between samples, is known as the *sampling period.*

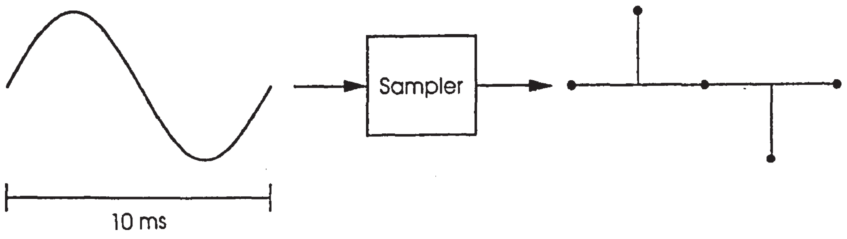
Here is an example that we've seen before—a quantized 100-Hz sinusoid being sampled every 0.5 ms, or, equivalently with a sampling frequency of 2 kHz [12 bits were used in the quantization (4096 levels) so you won't see the steps]:



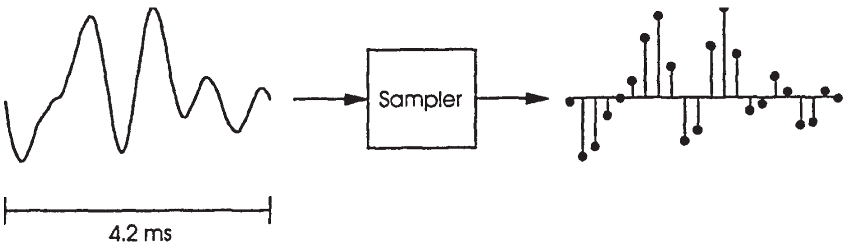
We could sample this waveform at a slower rate, say 1 kHz:



Or even more slowly, at 400 Hz



At the end of sampling (which we have assumed would follow quantization), we have our digital signal ready for input to a digital system. Of course, it's possible to sample any signal, not just sinusoids. Here, for example, is a 5-kHz sampled version of a short stretch of the quantized vowel /ɑ/ shown above (p. 319):



How fast does sampling have to be? The sampling theorem

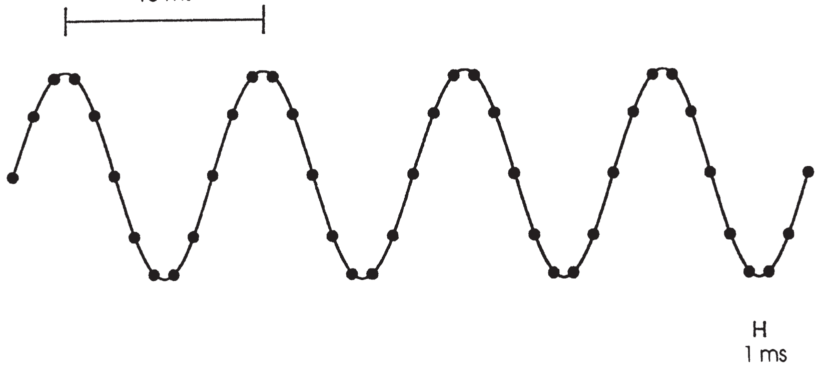
Although we can now convert a continuous time signal into a discrete time one (via sampling), we still have to address the problem of how well the information in the original signal is preserved in the sampled version of it. One thing should already be apparent from examining the results above of sampling the same 100-Hz sinusoid at three different sampling rates: the faster that sampling occurs, the more the sampled signal looks like the original. Does this mean that we should always sample as fast as we possibly can? The answer is no, for two reasons.

First, there are the practical problems. Fast sampling rates obviously lead to more data points per second than slow sampling rates. For a mode of storage with fixed capacity (like a compact disc or MP3 player), halving the sampling rate would allow twice as much music. Also, there are technical problems in making fast sampling devices and the solutions to them tend to be expensive.

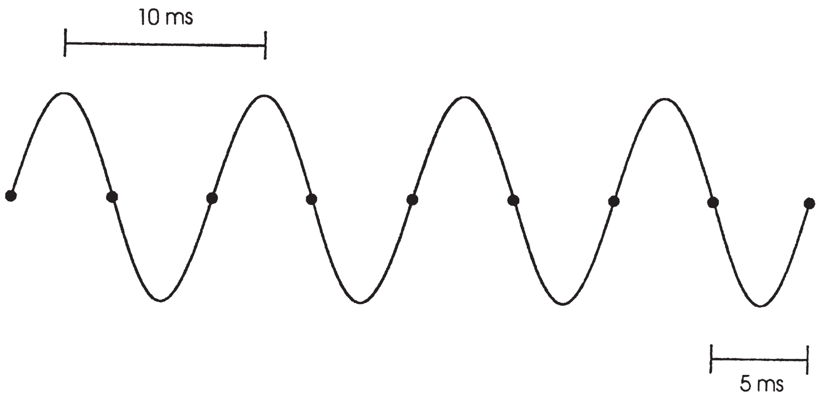
Second and much more importantly, there is a result, known as the *sampling theorem*, which gives the minimum sampling rate which allows the original signal to be reconstructed perfectly. It turns out that going faster than this doesn't help. The limitations on sampling, then, are quite different to those on quantization. The more levels of quantization, the better, because the original signal is better and better approximated (even though the practical advantage may be small beyond a certain point). There is no point at which *all* the information is retained. Sampling a sinusoid at an increasing rate does not ensure better approximation. Below the sampling rate set by the sampling theorem, the information at the input to the sampler is not preserved in the output—above this rate it is.

The rule given by the sampling theorem is simple—as long as the sampling rate is more than twice the frequency of the sinusoid, no information is lost. This rate is known as the *Nyquist rate.* We can get some intuitive understanding of this result by looking at what happens if we sample a sinusoid more slowly than the Nyquist rate.

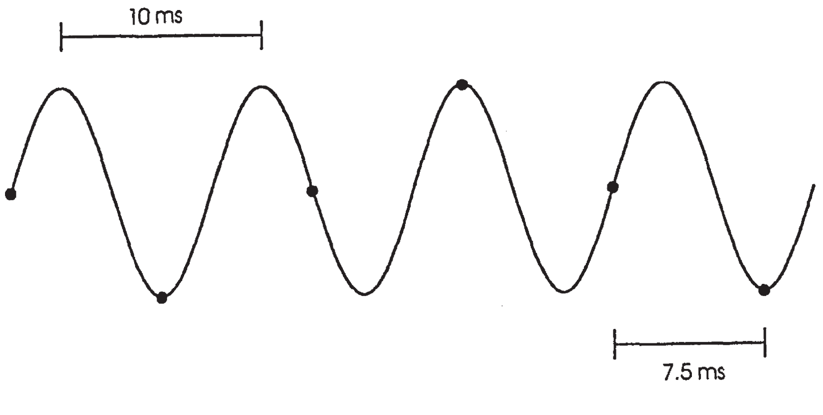
Let's go back to our 100-Hz sinusoid. The sampling theorem says that a sampling rate greater than 200 Hz (the Nyquist rate) will allow a reconstruction of the continuous time signal from its sampled version. We've already seen the results for rates rather faster than this above, but let's draw the digital waveform obtained by sampling at 1 kHz in a slightly different way, with the sample points as solid circles superimposed on the original waveform:



What happens if we now sample at exactly the Nyquist rate, 200 Hz?

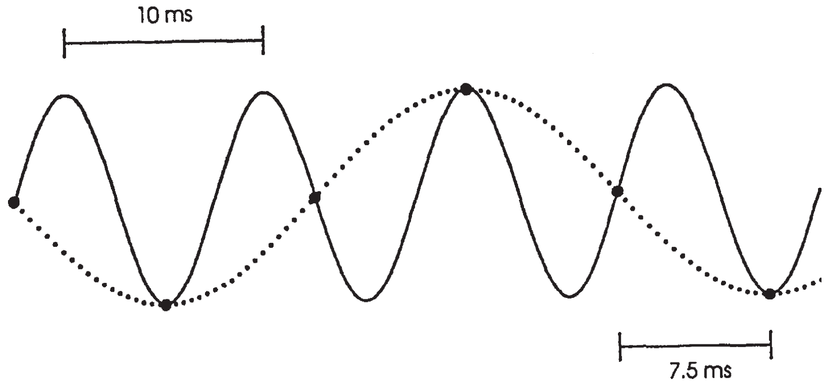


It should be obvious that this sampling rate is not fast enough. All the samples have a value of 0. In other words, the output of a 200-Hz sampler to a 100-Hz sinusoid is indistinguishable from no signal at all! Another way to think of this is as a direct current (DC) level, or a sinusoid of 0 Hz. Therefore, it is as if the 200-Hz sinusoid has been transformed into a sinusoid of 0 Hz. If we sample more slowly than the Nyquist rate of 200 Hz, even stranger things can happen. Here is the same 100-Hz sinusoid sampled now at a rate of about 133.3 Hz (with an exact sampling period of 7.5 ms):



The interesting thing here is that, at the sample points, the 100-Hz sinusoid is completely equivalent to a sinusoid of a much lower frequency, 33.3 Hz. This can be seen if we draw in the

33.3-Hz sinusoid as well, as a dotted line:



Thus, inputs of 100 Hz and 33.3 Hz are indistinguishable at the output of the sampler. Again, it is as if the 100-Hz sinusoid has been transformed into one of a lower frequency.

This transformation of sinusoids into different frequencies is known as *aliasing*. In everyday parlance, an alias is an identity you take on in place of your genuine identity. Here, a 100-Hz sinusoid has taken on the identity of one at 33.3 Hz.

It can be shown that, for a given sampling rate, all sinusoids will be represented at the output of the sampler as sinusoids of a frequency somewhere between DC (0 Hz) and half the sampling rate (known as the *Nyquist frequency*). So, sinusoids of any frequency at the input of a 200-Hz sampler will be represented at the output as sinusoids of frequencies between 0 and 100 Hz. Only for input sinusoids of frequencies below 100 Hz (that is, less than the Nyquist frequency) will the output match the input. At frequencies higher than this, input sinusoids will be aliased into lower frequencies. This is why the sampling theorem requires a sampling rate at least twice the frequency of the sinusoid to preserve the information in it.

Sampling complex signals

Having now determined the sampling rate necessary for sinusoidal signals, we can easily generalize this result to complex signals. In order to reconstruct any analogue signal from the output of a sampler, the sampling must be done at a rate more than twice as high as the highest frequency component in the signal.

Often, the sampling rate is fixed. It is important then to ensure that there are no frequency components present in the analogue signal above the Nyquist frequency; otherwise they will alias to lower frequencies. This is usually done by low-pass filtering the signal (known as *anti-aliasing* filtering). Of course, this means that any energy in the frequency region above the Nyquist frequency is lost, but this is preferable to the information being represented wrongly. Usually, the Nyquist frequency is chosen so that the components filtered out are of relatively low level, and their loss of little importance. In a compact disc, for example, the sampling frequency is 44.1 kHz in order to preserve information up to about 20 kHz, because that is more-or-less the highest frequency sinusoid that people can hear. For many applications in speech and hearing, sampling rates as low as 10 or 20 kHz are sufficient. Evoked responses measured from the brain can be accurately represented at even lower sampling rates, sometimes as low as 500 Hz.

Processing the digital signal

At this point, if the quantization and sampling have been done properly, we have an appropriate digital signal ready for processing of some kind. We will not, in fact, detail any particular digital system just yet, but you should have some idea of the types of processing that are possible.

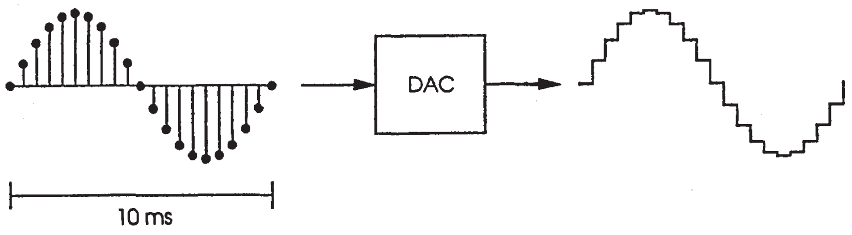
One popular use of digital systems is to mimic analogue systems. It's possible to create digital filters of all types (e.g. high- or low-pass), as we will see later in this chapter, and these can be much more flexible than their equivalent analogue counterparts. For example, it's relatively easy to design digital filters with such desirable characteristics as linear phase responses.

There is also much digital processing that would be extremely difficult, if not impossible, to implement using analogue electronics. Two techniques common in speech processing, *cepstral* and *linear-predictive analyses*, are of this sort.

A crucial point is that many of the digital systems in use may be considered to be LTI (although they are normally termed linear *shift*-invariant systems, with identical meaning). Hence, all the concepts that we've developed throughout this book have their counterparts in the digital world, although often in modified form.

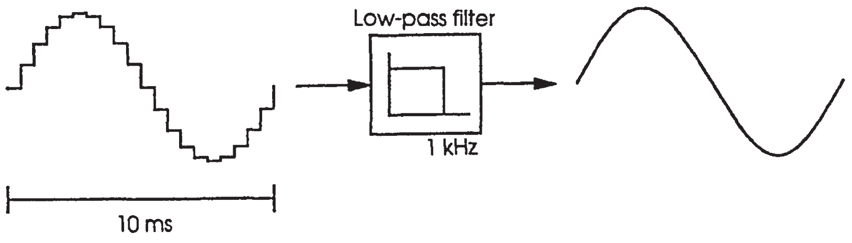
Reconverting back to analogue form

Having now processed the signal, we want to get it back into analogue form. The first step is to take the amplitude values and convert them from numbers back into voltages with a digital-to-analogue converter (or DAC, pronounced 'dak'). This is done at a sampling rate identical to the one initially performed, and essentially reverses the operation of the sampler—we are back to the quantized continuous time signal. Here's our digital 100-Hz sinusoid (sampled at 2 kHz), before and after D-to-A conversion:



Clearly, this output doesn't look the same as the original analogue sinusoid—nor would it sound the same. Analogue sinusoids don't have steps in them as this 'sinusoid' does. These steps are *not* the result of quantization. Although the signal is indeed still quantized, the quantization is fine enough to be ignored. Even with no quantization, such steps would still occur, as they arise from the operation of sampling. Analogue signals can change continuously since they exist at all moments of time; digital signals reconverted into analogue form must show steps, as changes can only occur at discrete moments of time.

As we've discussed before, fast changes like this in a signal represent high frequencies. But we've already shown that only frequencies up to the Nyquist frequency can be represented. Hence, we can low-pass filter this stepped signal (at half the sampling rate) to remove these meaningless high frequencies, while at the same time not throwing away any correct information. This will, in effect, smooth out the steps and leave us with the original sinusoid:



This low-pass filtering step (unlike the anti-aliasing filter discussed previously) is mandatory. If the original analogue signal is already limited in frequency content to less than half the sampling frequency (as in our 100-Hz sinusoid sampled at 2 kHz), low-pass anti-alias filtering prior to sampling would not be necessary (because there is no energy in the signal above 1 kHz). It would still be necessary after D-to-A conversion, though, in order to eliminate the 'steps' in the waveform.

A digital amplifier

Although you should now have a pretty clear idea of what a digital signal is like, digital signal processing may still seem rather a mystery. In fact, it is probably much easier to understand explicitly what digital rather than analogue systems do, because the basic operations often involve nothing more than multiplication and addition. But before we can describe any specific examples, we need to introduce some special terms and expressions for dealing with digital systems. The easiest way to do this is to take a particular system—a digital version of the amplifier introduced in Chapter 4 (p. 47). The amplifier has as its output a signal that is twice its input (roughly a 6 dB gain). This can be represented as a digital system in the following way:

$$y[n] = 2 \cdot x[n] \tag{1}$$

$x[]$ represents the input signal, $y[]$ represents the output signal, and the operator '$\cdot$' represents multiplication. Above, in our tabulation of a digital sinusoid, we used genuine values of time, spaced every ½ s. In fact, it is more typical to simply number the samples of a digital signal, which effectively means that the numbers specifying the time axis do not need to be stored (with the sampling rate noted separately). Therefore, the letter 'n' in the bracket of $x[n]$ indicates that the nth digital input sample is being referred to. Similarly $y[n]$ represents the nth output sample. 'n' would typically start at 0, because when you ask a computer scientist to count to 5, they say '0, 1, 2, 3, 4, 5'! Also, we conventionally think of waveforms as starting at time 0, so calling the first possible sample the 0th sample is not so farfetched.

When processing a digital signal then, n is incremented so that it steps through the entire set of samples from the first to the last (assuming a total of N values). This could be represented explicitly by specifying '$n = 0, N-1$' after any formula (like the one above) but this is rarely done.

Let's get back to our particular example. Equation (1) above indicates that the system takes the first digital input sample (at time 0) and multiplies it by 2 to obtain the first output sample. It then takes the second input sample and multiplies it by 2 again to give the next output sample and so on. These steps can be represented as follows:

$$y[0] = 2 \cdot x[0]$$

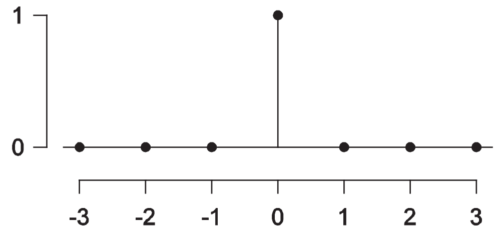
$$y[1] = 2 \cdot x[1]$$

$$\ldots$$

$$y[N-1] = 2 \cdot x[N-1]$$

Just as for analogue systems, a digital system can be described by a frequency response, or equivalently, an impulse response.

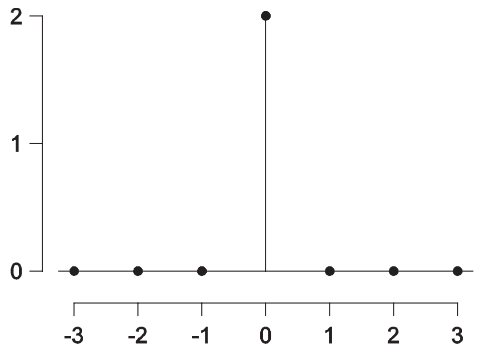
Let's determine the impulse response first. A digital impulse is very similar to an analogue one, but even simpler to understand. Just like an analogue impulse, a digital impulse only exists at one moment in time. Because digital signals are theoretical notions anyway, you shouldn't have any difficulty imagining such a signal, whereas it's hard to think about an analogue impulse because it is infinitely narrow. A digital impulse also has a finite amplitude of 1, so you need not imagine something with infinite amplitude. Here's what a digital impulse looks like:



It should be obvious to you what the impulse response of this digital amplifier must be. Remember, all you need do is put an impulse (a single sample value of 1 at time 0) into the digital amplifier. For all values of n not equal to 0, $x[n] = 0$, so the output of the filter will also be 0. Only for $n = 0$ need you do the calculation:

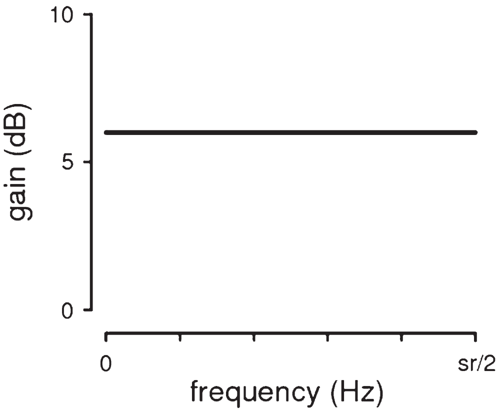
$$y[0] = 2 \cdot x[0] = 2 \cdot 1 = 2$$

Therefore, the impulse response would look like this:



Just as you might expect, the amplitude response of this digital filter could be calculated from the spectrum of the impulse response using a Fourier Transform. But a standard Fourier transform only works on analogue signals, so you would need a modified transform known as the *discrete Fourier transform* (DFT). A DFT takes a digital signal and determines what set of digital sinusoids (frequencies, amplitudes and phases) needs to be added together to synthesize that particular signal. You may also have heard of an FFT—a *fast Fourier transform*. An FFT gives exactly the same result as a DFT, but in a cleverer way, which makes the computations much faster. Therefore, computers always use FFTs to make their calculations.

However, no such calculation is really necessary for such a simple system. Clearly, any sinusoid passed through this system will have its amplitude doubled, so the amplitude response will have a gain of 2 (or 6 dB) for all frequencies. But remember that the range of frequencies will be limited to half the sampling rate (sr/2), which we have not explicitly specified in this example:



Take special note that the amplitude response of a digital *system* is continuous, existing at all frequencies (at least over a restricted frequency range) even though a digital *signal* only exists at discrete moments. This should make some sense—for any given sample rate, digital sinusoids can exist at any frequency, not just a specified set.

This digital amplifier is about the simplest system one can imagine that does something useful, and a digital impulse is the simplest useful signal. Let's now consider a situation in which digital sinusoids of various frequencies are passed through a very basic low-pass filter.

A simple digital low-pass filter

The system we'll consider is slightly more complex than the amplifier:

$$y[n] = \frac{x[n] + x[n-1]}{2} \tag{2}$$

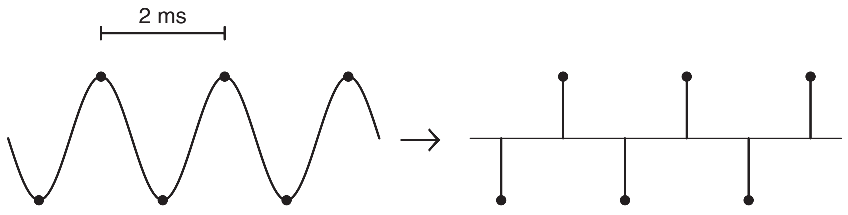
Unlike the digital amplifier, this system has two x's on the right-hand side. The first x is followed by n, which indicates the current input value, and the second is followed by $n{-}1$, which means the *preceding* input value. For example, if the current sample is at $n = 2$, then $n{-}1 = 1$ is the previous sample. Hence this formula indicates the current and the preceding inputs should be added and divided by two to obtain the output—which is to say, averaged.

Equation (2) makes it clear that we are calculating an average, but it will be generally more useful for us to rewrite this equation in a more conventional form as:

$$y[n] = 0.5 \cdot x[n] + 0.5 \cdot x[n-1] \tag{3}$$

The values that are multiplied against each of the input samples (here equal to 0.5) are known as the *coefficients* of the $x[n]$ and $x[n{-}1]$ terms. In fact, they are not essential for the low-pass characteristic of this filter, but simply change its overall gain. They are included here to introduce you to the idea of filter coefficients which we'll discuss further later.

For an input signal, we'll begin with a 500-Hz sinusoid sampled at 1000 times per second (1 kHz). The sampling theorem indicates that this sampling rate is the lowest rate that will accurately represent the 500-Hz sinusoid. The process of sampling results in each period of the sinusoid being sampled at two equally spaced points in time:



As you can see, the sinusoid varies in peak amplitude between +1 and −1. The first sample is at the trough (−1) and the second point at the peak (+1). If we specified the sequence of values in a list, the values of the sampled signal would be:

$$\ldots -1, +1, -1, +1, -1, +1 \ldots$$

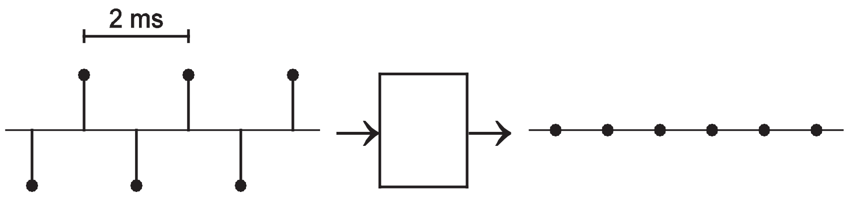
Let's now see what the system of equation (3) does to this signal. We'll start at the second sample in the sequence ($n = 1$) so that we've got a previous sample to average it with ($n = 0$). It is easy to see that the two input samples we need to calculate $y[1]$ are $x[1] = +1$ and $x[0] = -1$. Thus:

$$y[1] = (0.5 \cdot +1) + (0.5 \cdot -1) = +0.5 - 0.5 = 0$$

Stepping to the next (third) sample in the sequence ($n = 2$), now $x[2] = -1$ and $x[1] = +1$, so:

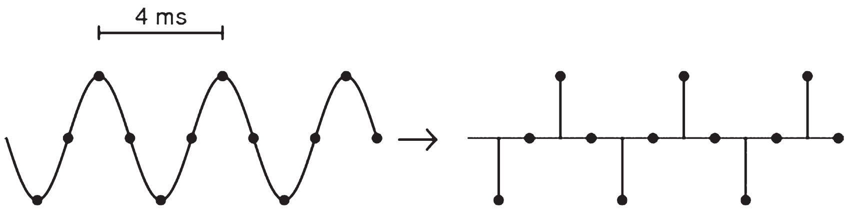
$$y[2] = (0.5 \cdot -1) + (0.5 \cdot +1) = -0.5 + 0.5 = 0$$

The output is zero again. This is true whatever adjacent pair of samples is used. Thus, for this sequence alternating between +1 and −1 (a sampled version of a 500-Hz sinusoid), all output amplitudes are 0. Effectively this particular input signal is removed, or *filtered out* by this simple operation.



What would this system do with a lower frequency sinusoid, say at 250 Hz? By now you should be able to work out that at the 1000-Hz sampling rate, the signal will repeat after four samples (one period of this sinusoid) and that starting at the same phase as for the 500-Hz sinusoid, the sample values will be:

$$\ldots -1, 0, +1, 0, -1, 0, +1, 0, \ldots$$



The output values for $y[n] = 0.5 \cdot x[n] + 0.5 \cdot x[n-1]$ need calculating for four pairs of values before they repeat:
For the first pair, when $n = 1$:

$$x[n-1] = -1, x[n] = 0, \text{ hence } y[1] = -0.5$$

For the second pair, when $n = 2$:

$$x[n-1] = 0, x[n] = +1, \text{ hence } y[2] = +0.5$$

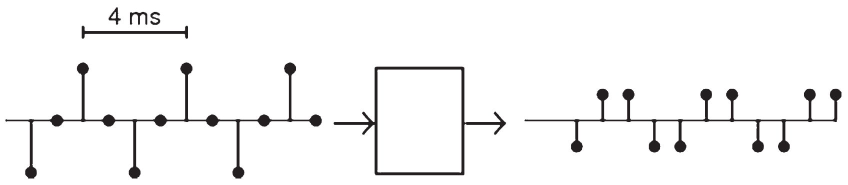
For the third pair, when $n = 3$:

$$x[n-1] = +1, x[n] = 0, \text{ hence } y[3] = +0.5$$

For the fourth pair, when $n = 4$:

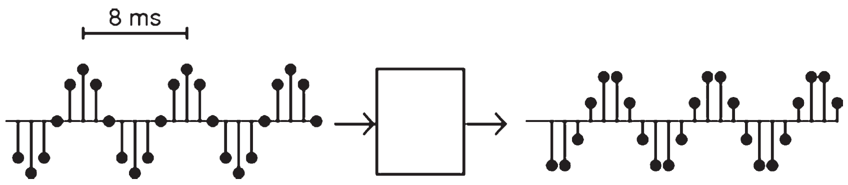
$$x[n-1] = 0, x[n] = -1, \text{ hence } y[4] = -0.5$$

This set of values then repeats, because the input signal is periodic, leading to:

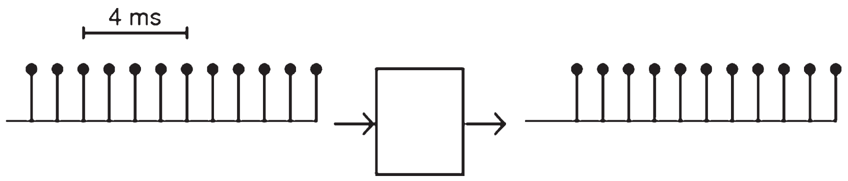


The system produces some output for this signal (that is, it is not filtered out completely) but it is attenuated to some extent. You may also note that a *phase shift* has been introduced, because the digital output wave looks different from the input wave, even though they are both digital sinusoids at the same frequency. This is especially apparent because the relatively low sampling rate leads to a rather sparse (but fully sufficient) representation.

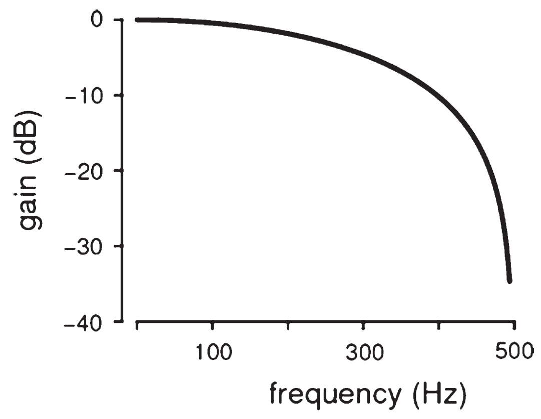
Since the 500-Hz signal is filtered out completely but the 250-Hz signal is not, the system can be described as a crude form of low-pass filter. This can be more readily seen if we use an even lower frequency sinusoid (125 Hz) and examine the input and output (you'll be asked to do the calculations in the exercises):



Finally, we will put a signal through our system which has a frequency of 0 Hz (DC). If we let its initial value be +1, then all subsequent values would also be +1, because by definition, this wave cannot vary in amplitude. So the response of $y[n] = 0.5 \cdot x[n] + 0.5 \cdot x[n-1]$ to all inputs is also +1.

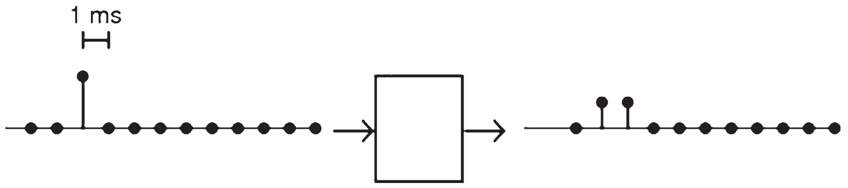


As usual, we want to summarize these measurements in an amplitude response, which shows the extent to which sinusoids of any frequency are passed through the system. This is clearly a low-pass filter:



Note that the gain at 500 Hz is not actually drawn on the graph. The system completely filters out this frequency, leading to a gain of 0 on linear scales, and hence one which cannot be expressed in dB.

It is also interesting to examine the response of this system to an impulse:



This impulse response is very short, only having non-zero values for two samples. Any filter with an impulse response that only consists of a finite number of values (no matter how long), is known as a *finite impulse response* (FIR) filter. Keep this in mind for later.

A simple digital high-pass filter

As you know, the other important type of filter is high pass. The following formula specifies a simple type of high-pass filter:

$$y[n] = 0.5 \cdot x[n] - 0.5 \cdot x[n-1] \tag{4}$$

Let's put the same 500-Hz sinusoid that we used above (sampled at 1 kHz) into this system and determine its output. You'll remember that the sampled signal alternated between −1 and +1.

We start at the second sample in the sequence ($n = 1$) to ensure we've got a previous sample to calculate with ($n = 0$) as we did before. The two input samples we need to obtain $y[1]$ are

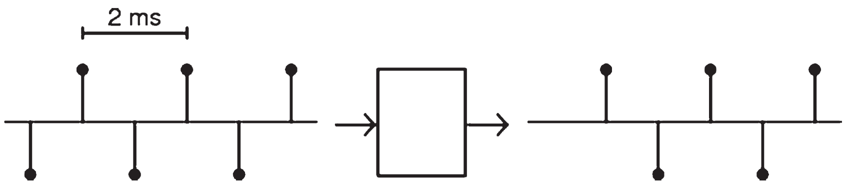
$x[1] = +1$ and $x[0] = -1$. Doing the mathematics (remembering that subtracting a negative value means adding that value) gives:

$$y[1] = (0.5 \cdot +1) - (0.5 \cdot -1) = +0.5 - (-0.5) = +1$$

For the next (third) sample in the sequence ($n = 2$), the input samples are now $x[2] = -1$ and $x[1] = +1$, so:

$$y[2] = (0.5 \cdot -1) - (0.5 \cdot +1) = -0.5 - (0.5) = -1$$

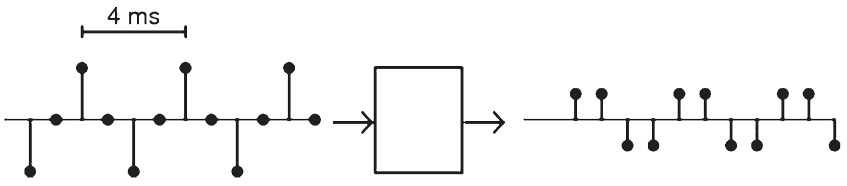
The numbers for all subsequent n, $n-1$ pairs are either [+1, −1] or [−1, +1], which give +1 and −1 as the result. Consequently, the original input signal is passed by this system unchanged:



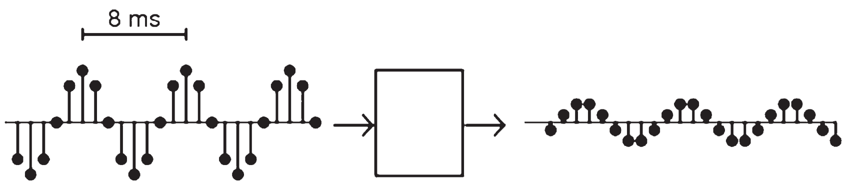
Next we'll put the 250-Hz sinusoid through the system. The values of the 250-Hz signal repeat every fourth term (. . . , −1, 0, +1, 0, −1, 0, +1, 0, . . .). Thus, the output values need calculating for four pairs of input values before they repeat, as shown at the left of the following table. The output values ($y[n]$) obtained by substituting these values into equation (4) are shown in the right column:

$x[n]$	$0.5 \cdot x[n]$	$x[n-1]$	$0.5 \cdot x[n-1]$	$y[n]$
$x[1] = 0$	0	$x[0] = -1$	−0.5	$y[1] = +0.5$
$x[2] = +1$	+0.5	$x[1] = 0$	0	$y[2] = +0.5$
$x[3] = 0$	0	$x[2] = +1$	+0.5	$y[3] = -0.5$
$x[4] = -1$	−0.5	$x[3] = 0$	0	$y[4] = -0.5$

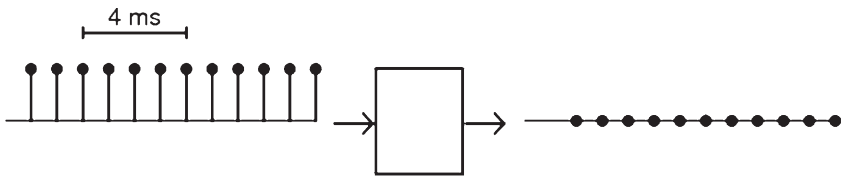
Thus, the 250-Hz input leads to the following output:



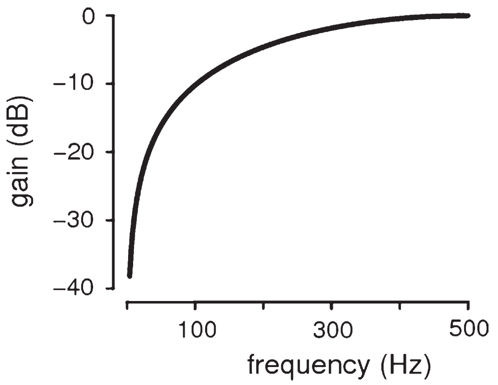
You should be able to do the calculations yourself now, for the 125-Hz sinusoid:



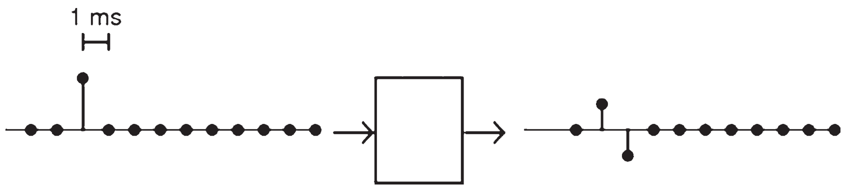
Finally, consider the particularly interesting case of the lowest frequency that can exist—DC. Assuming again that all values of this signal are +1, the response of $y[n] = 0.5 \cdot x[n] - 0.5 \cdot x[n{-}1]$ will always be zero. This is true whatever particular value the DC signal has. In other words, this system has completely filtered out the lowest frequency signal:



The complete amplitude response is shown in the next figure. This shows that this system has a high-pass characteristic with output levels going down as frequency decreases until it reaches zero at DC. (Again, of course, the gain of zero cannot be represented on dB scales.) You should find this a convincing example of a high-pass filter:



Consider, too, its impulse response, which we know to contain all the information about its properties:



Like the low-pass system, the impulse response is very short, again only having non-zero values for two samples. This is another example of an FIR system.

A simple infinite impulse response system

We have now shown three simple digital systems, all of which had impulse responses which were finite (in fact, quite short). As you might have guessed by our emphasis of this fact, not all digital systems are like this. FIR filters operate on a weighted sum of past inputs. Therefore, if the input becomes zero and stays at that value, then eventually the output will become zero as well. Therefore, the impulse response lasts only a finite time. Contrasted to this are *infinite impulse response* (IIR) filters for which the impulse response extends to infinity. Perhaps surprisingly, it is very easy to make a digital system that is IIR. All the system needs do is to reuse the *output* values of the system in its calculations, in addition to the input values. Here's an example:

$$y(n) = 0.5 \cdot x[n] + 0.5 \cdot y[n-1] \tag{5}$$

The important difference between this system and the three digital FIR systems we have considered earlier is that y terms appear on the left *and* right side of the equation. In this system, the current output ($y[n]$) is based on half the current input ($0.5 \cdot x[n]$) and half the previous output ($0.5 \cdot y[n-1]$). The ($0.5 \cdot y[n-1]$) term gives the system input from the past and, for this reason, IIR systems are sometimes referred to as systems with *memory*. In addition, because this implies information is being recycled, these are also known as *recursive* systems.

We can examine the response of this system just as we did for the FIR filters. Again we will use the 500-Hz signal as input. We'll start the system with current input $x[1] = +1$ and assume the previous output is zero ($y[0] = 0$). Putting these values into equation (5) gives:

$$y[1] = (0.5 \cdot 1) + (0.5 \cdot 0) = +0.5$$

The next input is -1 ($x[2] = -1$) and we have a past value of output ($y[1] = +0.5$). Using these values in equation (5) gives:

$$y[2] = 0.5 \cdot (-1) + (0.5 \cdot 0.5) = -0.25$$

The next two values are as follows:

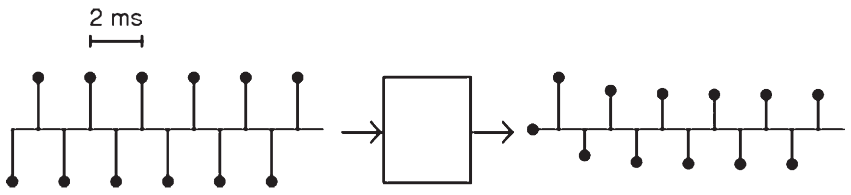
$$y[3] = (0.5 \cdot 1) + (0.5 \cdot -0.25) = 0.375$$

$$y[4] = (0.5 \cdot (-1) + (0.5 \cdot 0.375) = -0.3125$$

For a 500-Hz sinusoid, the FIR filters discussed earlier repeated their output after four samples. However, this system has a memory so the outputs do not repeat exactly. This is apparent when we calculate the output $y[5]$:

$$y[5] = (0.5 \cdot 1) + (0.5 \cdot -0.3125) = 0.34375$$

For this system, $y[1\,] = 0.5$ and $y[5] = 0.34375$ (that is, the values do not repeat). This is because of what is known as a *start-up transient*, due to the fact that the input wave started at a particular moment in time, and was zero before that. Eventually, the output does settle down to being periodic as you can see from the output wave here:

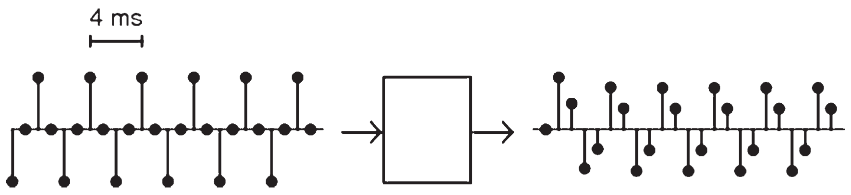


You'll note that more cycles of the input wave have been pictured, in order to allow the start-up transient to die away in the output. Also the y-axis of the output signal has been multiplied by a factor of 2, here and in the following three figures, so as to make the output waveforms easier to see.

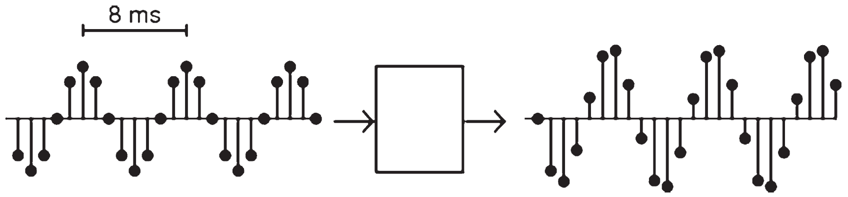
Next we put the 250-Hz sinusoid into the system. Again we assume the output value to be zero before the signal started so $y[1] = 0$. The output values ($y[n]$) obtained by substituting the appropriate values into equation (5) are shown in the right column:

$x[n]$	$0.5 \cdot x[n]$	$y[n-1]$	$0.5 \cdot y[n-1]$	$y[n]$
$x[1] = 0$	0	$y[0] = 0$	0	$y[1] = 0$
$x[2] = +1$	+0.5	$y[1] = 0$	0	$y[2] = +0.5$
$x[3] = 0$	0	$y[2] = +0.5$	+0.25	$y[3] = +0.25$
$x[4] = -1$	−0.5	$y[3] = +0.25$	0.125	$y[4] = -0.375$

Here's what the input and output signals look like. Note how the output level (once it has reached equilibrium) has increased a bit from that obtained with an input frequency of 500 Hz:



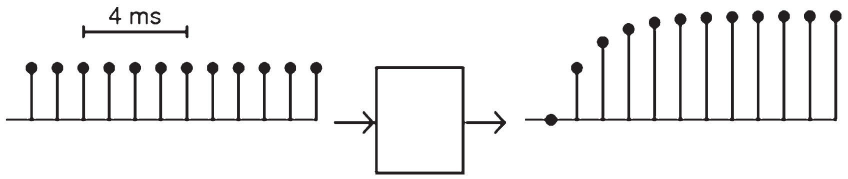
The output amplitude increases again for a 125-Hz sinusoid (you will be asked to do these calculations in the exercises). Only three cycles are shown here because the start-up transient is very short:



Putting a DC signal through our system and letting the first output value be $y[0] = 0$ (as previously) and all input values be $+1$, the response can be calculated as follows:

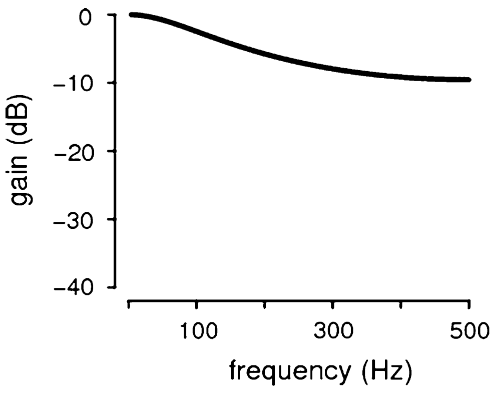
$x[n]$	$0.5 \cdot x[n]$	$y[n-1]$	$0.5 \cdot y[n-1]$	$y[n]$
$x[1] = +1$	$+0.5$	$y[0] = 0$	0	$y[1] = +0.5$
$x[2] = +1$	$+0.5$	$y[1] = +0.5$	$+0.25$	$y[2] = +0.75$
$x[3] = +1$	$+0.5$	$y[2] = +0.75$	$+0.375$	$y[3] = +0.875$
$x[4] = +1$	$+0.5$	$y[3] = +0.875$	$+0.4375$	$y[4] = +0.9375$

which looks like this:



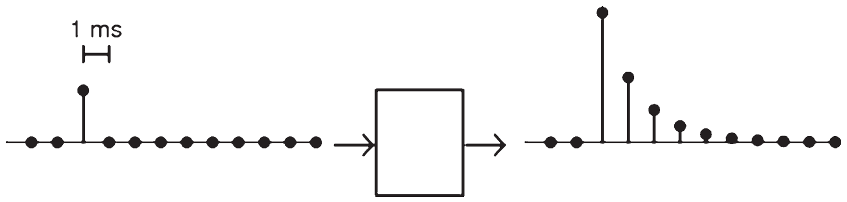
The start-up transient is especially prominent in this case, as the filter rises gradually in the output amplitude over its first five or six values. It then reaches an equilibrium level larger than any other input frequency we have tried.

Given that the output amplitude from this system has been increasing as the input frequency has been progressively lowered, it should not surprise you to find that this is an example of a low-pass filter:



This low-pass filter does not vary so much in its effect over frequency as our previous filter, with a maximum attenuation of only about 10 dB.

Let's look too at the impulse response of this system. Here the the *y*-axis of the output signal as been multiplied by a factor of 4 so as to make the decay of the waveform visible for longer:



You can see that the impulse response goes on for an extended period of time, in theory, forever. In fact, the quantisation of values represented digitally will mean that the number will become so small, it will round to zero, as it appears to have done here. Theoretical developments typically assume, as we mentioned above, that quantisation effects can be ignored. So we still talk about an IIR even if this will not happen in a case like this in any real digital system.

FIR and IIR systems

A last word about FIR and IIR systems. FIR systems have a number of useful properties that lead them to be preferred over IIR systems for certain applications. One is that they can easily be designed to have a linear phase response by making the impulse response symmetric. Linear phase responses are sometimes desirable as they delay all sinusoidal components by the same amount so they tend to preserve waveform shapes (see p. 103 for further discussion of this). Additionally, FIR systems are simpler to understand and design and are always *stable* (which is to say, they never end up veering off into infinite output values no matter what the input—IIR filters can readily do this unless carefully designed). The main disadvantage of FIR systems is that they are computationally less efficient than IIR ones. In other words, more multiplications and additions are necessary in order to get a similar frequency response. This, of course, takes more time and computing power.

Concluding remarks

This completes our discussion of digital signals and systems, although we have obviously just scratched the surface. Given the ubiquity of digital signal processing, it may not surprise you to learn that we have already been using these techniques throughout the book, even though it was never mentioned! Perhaps the best examples of this can be found in Chapter 11, where all but a few of the spectrograms were made on a computer, hence required the use of digital signals and systems. Even our FM sweep was generated digitally.

Exercises

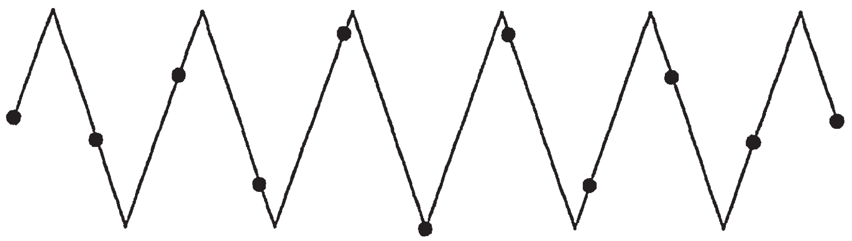
1. A signal containing no energy above 5 kHz is sampled at 10 kHz and stored on a computer. If this is to be output through a DAC at 10 kHz, sketch the amplitude response of a realistic output filter that would attenuate by at least 12 dB all components in the signal above 4.5 kHz.

2. What sampling rate would you recommend if you were recording digitally (a) a 100-Hz sinusoid, (b) a 200-Hz sinusoid, (c) a 100-Hz square wave, (d) a 200-Hz square wave, (e) a 100-Hz triangle wave and (f) a 200-Hz triangle wave? What factors govern your choice?

3. If you wanted to construct a digital metronome which ticked twice a second, what output rate would you use and how many bits on the DAC would be appropriate? (Assume that it is only important that the tick occur regularly, with the form of the pulse giving the tick being irrelevant.)

4. Design and sketch an input–output function for a 1-bit DAC that would be appropriate for quantizing sinusoids with peak-to-peak amplitudes of 1 V. Sketch a sinusoid before and after quantizing. Do the same for a 2- and 3-bit quantizer. Comment on the three quantized sinusoids.

5. A signal generator was set up to produce a triangular signal which was sampled at the points indicated. Was the sampling performed correctly? If not, say in what way the process went wrong and suggest remedial measures.



6. Cine film is recorded at the rate of 24 images per second. If you watch a cowboy film, the wheels on the wagons appear

to move slower than we know they would. Write an account of why this occurs. (You may include diagrams if you think this helps.)

7. Here are the values for one period of the 125-Hz digital sinusoid that was used in the examples of simple digital systems:

−0.71	−1.00	−0.71	0.00	0.71	1.00	0.71	0.00

Using these, do the calculations for two complete cycles of the input to check the output waves shown on pages 332, 335 and 338.

8. The following complex periodic signals are to be filtered, sampled, stored in a computer, and then played out again:

(a) 100-Hz fundamental and 19 higher harmonics at constant amplitude.
(b) 100-Hz fundamental and seven higher harmonics at constant amplitude.
(c) 100 Hz fundamental and 19 higher harmonics dropping at 12 dB/octave.
(d) 100-Hz fundamental and seven higher harmonics dropping at 12 dB/octave.
(e) 100-Hz fundamental and 19 higher harmonics dropping at 6 dB/octave.
(f) 100-Hz fundamental and seven higher harmonics dropping at 6 dB/octave.

The sampling rates, filter cutoffs and filter slopes of the filters used prior to sampling these signals (and in reconversion to analogue form) are tabulated below. (i) For each filter specification indicate which signals have been filtered properly so that the output signal accurately represents at least some range of the spectrum of the input signal. (ii) For the first signal, summarize the relationship between the filter cutoffs and the sampling rate in octaves. (iii) For this same signal, sketch roughly the amplitude spectrum of the input signal, amplitude response of the filter and the spectrum of the output. Summarize the relationship between the input spectra and filter characteristics in words. Summarize the major differences between the input and output signals.

Sampling rate	Filter cutoff	Filter slope frequency
1 kHz	250 Hz	Infinitely steep
1 kHz	250 Hz	6 dB/octave
1 kHz	250 Hz	3 dB/octave
1 kHz	–	No filtering
1 kHz	500 Hz	Infinitely steep
1 kHz	500 Hz	6 dB/octave
1 kHz	500 Hz	3 dB/octave
1 kHz	–	No filtering
1 kHz	1 kHz	Infinitely steep
1 kHz	1 kHz	6 dB/octave
1 kHz	1 kHz	3 dB/octave
1 kHz	–	No filtering
1 kHz	2 kHz	Infinitely steep
1 kHz	2 kHz	6 dB/octave
1 kHz	2 kHz	3 dB/octave
1 kHz	–	No filtering
2 kHz	1 kHz	Infinitely steep
2 kHz	1 kHz	6 dB/octave
2 kHz	1 kHz	3 dB/octave
2 kHz	–	No filtering
2 kHz	2 kHz	Infinitely steep
2 kHz	2 kHz	6 dB/octave
2 kHz	2 kHz	3 dB/octave
2 kHz	–	No filtering
2 kHz	4 kHz	Infinitely steep
2 kHz	4 kHz	6 dB/octave
2 kHz	4 kHz	3 dB/octave
2 kHz	–	No filtering
4 kHz	1 kHz	Infinitely steep
4 kHz	1 kHz	6 dB/octave
4 kHz	1 kHz	3 dB/octave
4 kHz	–	No filtering
4 kHz	4 kHz	Infinitely steep
4 kHz	4 kHz	6 dB/octave
4 kHz	4 kHz	3 dB/octave
4 kHz	–	No filtering
4 kHz	8 kHz	Infinitely steep
4 kHz	8 kHz	6 dB/octave
4 kHz	8 kHz	3 dB/octave
4 kHz	–	No filtering
4 kHz	16 kHz	Infinitely steep
4 kHz	16 kHz	6 dB/octave
4 kHz	16 kHz	3 dB/octave
4 kHz	–	No filtering