Noise

Much of signal processing involves extracting signals of interest from noise. Without noise to combat, a radar receiver could detect an echo by simple energy thresholding. In a noiseless world an infinite amount of information could be transmitted through a communications channel every second. Were it not for noise, signal classification would be reduced to dictionary lookup. Yet signals in the real world are always noisy. Radar echoes are buried under noise, making their detection impossible without sophisticated processing. Modem signals rely on complex modulation and error correction schemes to approach the maximum rate attainable through noisy telephone lines. Due to noise, signal classification is still more an art than a science. Extracting a signal from noise can rely on knowledge of the clean signal and/or knowledge of the noise. Up to now we have learned to characterize clean signals; in this chapter we will study the characteristics of noise.

As discussed in Section 2.3, a stochastic signal cannot be precisely predicted, being bound only by its statistics. What do we mean by 'statistics'? It is jokingly said that *probability* is the science of turning random numbers into mathematical laws, while *statistics* is the art of turning mathematical laws into random numbers. The point of the joke is that most people take 'statistics' to mean a technique for analyzing empirical data that enables one to prove just about anything. In this book 'statistics' refers to something far more tangible, namely the parameters of probabilistic laws that govern a signal. Familiar statistics are the average or *mean value* and the *variance*.

In this chapter we will learn how noisy signals can be characterized and simulated. We will study a naive approach that considers noise to be merely a pathological example of signals not unlike those we have previously met. In particular, we will take the opportunity to examine the fascinating world of chaotic deterministic signals, which for all practical purposes are indistinguishable from stochastic signals but can be approached via periodic signals. Finally, we will briefly discuss the mathematical theory of truly stochastic signals.

5.1 Unpredictable Signals

'Pure noise' is the name we give to a quintessential stochastic signal, one that has only probabilistic elements and no deterministic ones. Put even more simply, pure noise is completely *random*; it obeys only probabilistic laws and can never be perfectly predicted. 'Plain' noise has a softer definition in that we allow signals with some deterministic characteristics, e.g. the sum of a pure noise and a deterministic signal. The ratio of the energy of the deterministic signal to that of the pure noise component is called the **S**ignal to Noise **R**atio (SNR), usually specified in dB. A signal with finite SNR is unpredictable to some degree. Our guesses regarding such noisy signals may be better than random, but we can quite never pin them down. An SNR of 0dB (SNR=1) means the signal and noise have equal energies.

There are four distinguishable ways for a signal to appear unpredictable: it may be pseudorandom, incompletely known, chaotic, or genuinely stochastic. The exact boundaries between these four may not always be clear, but there is progressively more known about the signal as we advance from the first to the third. Only the fourth option leads to true noise, but in practice it may be impossible to differentiate even between it and the other three.

A pseudorandom signal is completely deterministic, being generated by some completely defined algorithm. However, this algorithm is assumed to be unknown to us, and is conceivably quite complex. Being ignorant of the algorithm, the signal's behavior seems to us quite arbitrary, jumping capriciously between different values without rhyme or reason; but to the initiated the signal's behavior is entirely reasonable and predictable. If we may assume that there is no correlation between the unknown generating algorithm and systems with which the signal may interact, then for all intents and purposes a pseudorandom signal *is* noise. Pseudorandom signals will be treated in more detail in Section 5.4.

An incompletely known signal is also completely deterministic, being generated by a known algorithm that may depend on several parameters. The details of this algorithm and some, but not all, of these parameters are known to us, the others being *hidden variables*. Were we to know all these parameters the signal would be completely predictable, but our state of knowledge does not allow us to do so. In practice knowing the form and some of the parameters may not help us in the least, and the signal seems to us completely erratic and noise-like. In theory the signal itself is not erratic at all; it's simply a matter of our own ignorance!

A chaotic signal is also completely deterministic, being generated by a completely specified algorithm that may even be completely known to us. However, a chaotic signal seems noisy because of numeric sensitivity of this algorithm that causes us to rapidly lose information about the signal with the passage of time. Were all initial conditions to be specified to infinite precision, and all calculations to be performed with infinite accuracy, the signal would indeed be perfectly predictable; but *any* imprecision of knowledge or inaccuracy of computation will inevitably lead to complete loss of predictability after enough time has passed. Such *chaotic* signals will be treated in detail in Section 5.5.

A truly stochastic signal is one that is *not* generated by any deterministic algorithm at all. The time between successive clicks of a Geiger counter or the thermal noise measured across a resistor are typical examples. At a fundamental level, quantum mechanics tells us that nature abounds with such genuinely random signals. The philosophical and scientific consequences of this idea are profound [53]. The implications for DSP are also far-reaching, and will be discussed briefly in Section 5.6. However, a formal treatment of stochastic signals is beyond the scope of this book.

EXERCISES

- 5.1.1 The game of guessit is played by two or more people. First the players agree upon a lengthy list of functions of one variable t, each of which is also dependent on one or two parameters. The *inventor* picks function from the list and supplies parameters. Each *analyst* in turn can request a single value of the function and attempt to guess which function has been selected. What strategy should the inventor use to make the analysts' task more difficult? What tactics can the analysts use? Try playing guessit with some friends.
- 5.1.2 Generate a signal x with values in the interval [0...1] by starting at an arbitrary value in the interval and iterating $x_{n+1} = \lambda x_n(1-x_n)$ for $0 \le \lambda \le 4$. For what values of λ does this signal look random?
- 5.1.3 To which of the four types of unpredictable signal does each of the following most closely belong?
 - 1. Static noise on shortwave radio
 - 2. Sequence of heads (s=1) and tails (s=0) obtained by throwing a coin
 - 3. World population as a function of time
 - 4. Value of stock portfolio as a function of time
 - 5. Sequence produced by your compiler's random number generator
 - 6. Distance from earth to a given comet
 - 7. Position of a certain drop of water going down a waterfall
 - 8. Maximum daily temperature at your location
 - 9. The sequence of successive digits of π

5.2 A Naive View of Noise

No matter what its source, a noise-like signal is very different from the signals with which we have dealt so far. Although we can observe it as a function of time, its graph resembles *modern art* as compared to the classical lines of deterministic signals; and every time we observe and plot it we get a completely different graph. In Figures 5.1, 5.2, and 5.3 we plot distinct noise signals in the time domain. All the plots in each figure represent the same noise signal, and are called *realizations* of the underlying noise. No two realizations are precisely the same, yet there are noticeable similarities between realizations of the same noise, and different noise signals may be easily distinguishable by eye.

Were you to be presented with a new, previously unseen realization of one of the noise signals of the figures, and asked to which it belonged, you would probably have little difficulty in classifying it. How do you do it? How can we best characterize noise signals? It will not surprise you to learn that noise signals, like deterministic signals, have characteristics in the time domain and in the frequency domain.

In the time domain we are interested in the statistical attributes of individual signal values ν_n , such as the mean (average) $\langle \nu \rangle$, the variance or standard deviation, and the moments of higher orders. The set of *all* parameters that determine the probabilistic laws is called *sufficient statistics*. Sufficient statistics are not sufficient to enable us to precisely predict the signal's value at any point in time, but they constitute the most complete description of a stochastic signal that there is. Noise signals are called *stationary* when these statistics are not time-dependent. This implies that the probabilistic properties of the noise do not change with time; so if we measure the mean and variance now, or half an hour from now, we will get the same result.

We will almost always assume stationary noise signals to have zero mean, $\langle \nu \rangle = 0$. This is because noise v(t) of nonzero average can always be written

$$v(t) = \langle v \rangle + \nu(t)$$
 A D $v_n = \langle v \rangle + \nu_n$

where the constant $\langle v \rangle$ is of course a (deterministic) DC signal, and ν is noise with zero mean. There is no reason to apply complex techniques for stochastic signals to the completely deterministic DC portion. which can be handled by methods of the previous chapters.



Figure 5.1: A few realizations of a noise signal. The set of all such realizations is called the *ensemble*. Note that each realization is erratic, but although the different realizations are quite varied in detail, there *is* something similar about them.



Figure 5.2: A few realizations of another noise signal. Note the differences between this noise signal and the previous one. Although both have zero average and roughly the same standard deviation, the first is uniformly distributed while this signal is Gaussian distributed. A few values are off-scale and thus do not appear.



Figure 5.3: A few realizations of a third noise signal. Note the differences between this noise signal and the previous two. Although the signal is also zero average and of the same standard deviation, the first two signals were *white* while this signal has been low-pass filtered and contains less high-frequency energy.

NOISE

The most detailed information concerning the statistics of individual signal values is given by the complete probability distribution these values. Probability distributions are functions p(x) that tell us the probability of the signals taking on the value x. Digital signals can only take on a finite number of values, and thus (at least in principle) we can record the complete probability distribution as a table. To demonstrate this consider a noise signal that can take on only the values -1, 0, 1 and whose probability distribution is the following.

$$p(-1) = \frac{1}{4}$$
 $p(0) = \frac{1}{2}$ $p(+1) = \frac{1}{4}$ (5.1)

Note that the probabilities sum to one since each signal value *must* be either -1, 0, or +1. One signal with such a distribution may be

$$\dots 0, -1, +1, -1, -1, 0, +1, 0, 0, 0, +1, +1, 0, 0, -1, 0, \dots$$

while another could be

$$\dots 0, +1, 0, -1, +1, 0, -1, -1, 0, +1, 0, -1, 0, 0, +1, 0, \dots$$

as the reader may verify.

Given a long enough sample of a digital signal with unknown distribution, we can estimate its probability distribution by simply counting the number of times each value appears and at the end dividing by the number of signal values observed. For example, the noise signal

$$\dots -1, 0, +1, +1, 0, -1, 0, -1, 0, +1, +1, -1, +1, 0, -1, \dots$$

has a probability distribution close to $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$. The probability distribution of any digital signal must sum to unity (i.e., must be *normalized*)

$$\sum p(x_i) = 1 \tag{5.2}$$

where the sum is over all possible signal values.

We said before that the probability distribution contains the most detailed information available as to individual signal values. This implies that all single signal value statistics can be derived from it. For a digital signal we can express the mean as a sum over time,

$$\mu = \langle s_n \rangle = \frac{1}{N} \sum_{n=1}^N s_n \tag{5.3}$$

or we can sort the terms such that smaller s_n appear before larger ones.

This is in turn equivalent to summing each observed signal value s times the relative number of times it was observed p(s),

$$\mu = \sum_{s=-\infty}^{\infty} p(s)s \tag{5.4}$$

which is seen to be a simple sum of the probability distribution. The variance is defined to be the mean-squared deviation from the mean

$$\sigma^{2} = \left\langle (s_{n} - \mu)^{2} \right\rangle = \frac{1}{N} \sum_{n=1}^{N} (s_{n} - \mu)^{2}$$
(5.5)

which can also be written in terms of the probability distribution.

$$\sigma^2 = \sum_{x=-\infty}^{\infty} p(x)(x-\mu)^2 \tag{5.6}$$

Analog signals have a nondenumerably infinite number of possible signal values, and so a table of probabilities cannot be constructed. In such cases we may resort to using histograms, which is similar to digitizing the analog signal. We quantize the real axis into bins of width δx , and similar to the digital case we count the number of times signal values fall into each bin. If the histogram is too rough we can choose a smaller bin-width δx . In the limit of infinitesimal bin-width we obtain the continuous probability distribution p(x), from which all finite width histograms can be recovered by integration. Since the probability distribution does not change appreciably for close values, doubling small enough bin-widths should almost precisely double the number of values falling into each of the respective bins. Put another way, the probability of the signal value x falling into the histogram bin of width δx centered on x_0 is $p(x_0)\delta x$, assuming δx is small enough. For larger bin-widths integration is required, the probability of the signal value being between x_1 and x_2 being $\int_{x_1}^{x_2} p(x) dx$. Since every signal value must be some real number, the entire distribution must be normalized.

$$\int_{-\infty}^{\infty} p(x) \, dx = 1 \tag{5.7}$$

In analogy with the digital case, the mean and variance are given by the following.

$$\mu = \int_{-\infty}^{\infty} p(x) x \, dx \qquad \sigma^2 = \int_{-\infty}^{\infty} p(x) (x - \mu)^2 dx \qquad (5.8)$$



Figure 5.4: Four different probability distributions. (A) represents the uniform distribution. (B) depicts an exponential distribution. (C) is the bell-shaped Gaussian (or normal) distribution. (D) is a representative bimodal distribution, actually the mixture of two Gaussians with different means.

From its very definition, the probability distribution of a random signal must be nonnegative and have an integral of one. There are a large number of such functions! For example, signal values may be uniformly distributed over some range, or exponentially distributed, or have a Gaussian (normal) distribution with some mean and variance, or be multimodal. Uniformly distributed signals only take on values in a certain range, and all of these values are equally probable, even those close to the edges. In Figure 5.4.A we depict graphically the uniform distribution. Gaussian distribution means that all signal values are possible, but that there is a most probable value (called the mean μ) and that the probability decreases as we deviate from the mean forming a bell-shaped curve with some characteristic width (the standard deviation σ). Mathematically,

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}}$$
(5.9)

is the famous Gaussian function, depicted in Figure 5.4.C. It is well known that when many students take an exam, their grades tend to be distributed in just this way. The rather strange constant before the exponent ensures that the Gaussian is normalized.

The frequency domain characteristics of random signals are completely distinct from the single-time signal value characteristics we have discussed so far. This may seem remarkable at first, since in DSP we become accustomed to time and frequency being two ways of looking at one reality. However, the dissimilarity is quite simple to comprehend. Consider a digital signal

$$s_1, s_2, s_3, \ldots s_N$$

with some signal value distribution and a new signal obtained by arbitrarily replicating each signal value

so that each value appears twice in a row. The new signal obviously has the same single-sample statistics as the original one, but its frequencies have been halved! Alternatively, consider permuting the order of signal values; this once again obviously results in an identical probability distribution, but quite different frequency characteristics! A signal's frequency statistics are determined by the relationship between signal values at various relative positions, and thus contains information different from the signal value statistics.

We will often talk of white noise. White noise is similar to white light in that its spectrum is flat (constant, independent of frequency). Having all possible frequencies allows the signal to change very rapidly, indeed even knowing the entire past history of a white noise signal does not contribute anything to prediction of its future. We thus call a discrete time signal s_n white noise if observation of $\{s_n\}_{n=-\infty}^{k-1}$ does not allow us to say anything useful about the value of s_k other than what the single-signal value statistics tell us.

Of course not all noise is white; when the noise signal's spectrum is concentrated in part of the frequency axis we call it colored noise. Colored noise can be made by passing white noise through a band-pass filter, a device that selectively enhances Fourier components in a certain range and rejects others. As we decrease the bandwidth of the filter, the signal more and more resembles a sine wave at the filter's central frequency, and thus becomes more and more predictable.

Since they are independent, time and frequency domain characteristics can be combined in arbitrary ways. For example, white noise may happen to be normally distributed, in which case we speak of Gaussian white noise. However, white noise may be distributed in many other ways, for instance, uniformly, or even limited to a finite number of values. This is possible because the time domain characteristics emanate from the individual signal values, while the frequency domain attributes take into account the relation between values at specific times.

Our naive description of noise is now complete. Noise is just like any other signal—it has well defined time domain and frequency domain properties. Although we have not previously seen a flat spectrum like that of white noise, nothing prevents a deterministic signal from having that spectrum; and colored noise has narrower spectra, more similar to those with which we are familiar. The time domain characterization of noise *is* different from that of regular signals—rather than specifying how to create the signal, we must content ourselves with giving the signal's statistics. From our naive point of view we can think of all noise signals as being pseudorandom or incompletely known; we suppose that if we had more information we *could* describe the 'noise signal' in the time domain just as we describe other signals.

The reader probably realizes from our use of the word *naive* in describing this characterization of noise, that this isn't the entire story. It turns out that stochastic signals don't even *have* a spectrum in the usual sense of the word, and that more sophisticated probabilistic apparatus is required for the description of the time domain properties as well. We will take up these topics in Section 5.6. However, our naive theory is powerful enough to allow us to solve many practical problems. The next section deals with one of the first successful applications of noise removal, the processing of radar returns.

EXERCISES

- 5.2.1 Write a program to generate digital noise signals with probability distribution (5.1). Estimate the probability distribution using 10, 100, 1000, and 10,000 samples. What is the error of the estimation?
- 5.2.2 Equation (5.6) for the variance require two passes through the signal values; the first for computation of μ and the second for σ^2 . Find a single-pass algorithm.
- 5.2.3 Using the random number generator supplied with your compiler write a zero-mean and unity variance noise generator. Make a histogram of the values it produces. Is it uniform? Calculate the empirical mean and standard deviation. How close to the desired values are they?
- 5.2.4 Using the noise generator of the previous exercise, generate pairs of random numbers and plot them as x, y points in the plane. Do you see any patterns? Try skipping L values between the x and y.
- 5.2.5 The noise generator you built above depends mainly on the most significant bits of the standard random number generator. Write a noise generator that depends on the least significant bits. Is this better or worse?
- 5.2.6 You are required to build the sample value histogram of a signal that only takes on values in a limited range, based on N samples. If you use too few bins you might miss relevant features, while too many bins will lead to a noisy histogram. What is the 'right' number of bins, assuming the probability distribution is approximately flat? What is the error for 10,000 samples in 100 bins?
- 5.2.7 What are the average, variance, and standard deviation of a Gaussian signal? What are the sufficient statistics? In what way is a Gaussian noise signal the simplest type of noise?

5.3 Noise Reduction by Averaging

Radar is an acronym for radio detection and ranging. The basic principle of range finding using radar was first patented in 1935 by Robert Watson-Watt, but practical implementations were perfected by American and British scientists during World War II. Although modern radars are complex signal processing systems, the principles of the basic pulse radar are simple to explain. The radar transmitter periodically sends out a powerful electromagnetic pulse of short time duration; the time between pulses is called the Pulse Repetition Interval (PRI). The pulse leaves the transmitter at the speed of light c and impinges upon various objects, whereupon minute fractions of the original signal energy are reflected back to the radar receiver. The round-trip time between the transmission of the pulse and the reception of the returned echo can thus be used to determine the distance from the radar to the object

$$r = \frac{1}{2}cT \tag{5.10}$$

where the speed of light c is conveniently expressed as about 300 meters per microsecond.

The radar receiver is responsible for detecting the presence of an echo and measuring its Time Of Arrival (TOA). The time between the TOA and the previous pulse transmission is called the *lag* which, assuming no ambiguity is possible, should equal the aforementioned round-trip time T. In order to avoid ambiguity the lag should be less than the PRI. Radar receivers must be extremely sensitive in order to detect the minute amounts of energy reflected by the objects to be detected. To avoid damaging its circuitry, the radar receiver is blanked during pulse transmission; and in order to keep the blanking time (and thus distance to the closest detectable target) minimal we try to transmit narrow pulse widths. This limits the amount of energy that may be transmitted, further decreasing the strength of the echo. Unfortunately, large amounts of natural and man-made noise are picked up as well, and the desired reflections may be partially or completely masked. In order to enhance the echo detection various methods have been developed to distinguish between the desired reflection signal and the noise. In general such a method may exploit characteristics of the signal, characteristics of the noise, or both. In this section we show how to utilize the knowledge we have acquired regarding the attributes of noise; the known PRI being the only signal-related information exploited. In Section 9.6 we will see how to improve on our results, notably by embedding easily detectable patterns into the pulses.

172 NOISE

We can view the received signal as being the sum of a deterministic periodic signal x_n and an additive zero-mean noise signal ν_n

$$y_n = x_n + \nu_n$$

and our task is to recover x_n to the best of our abilities. The periodicity (with period equal to the PRI) of the desired signal derives from the supposition that the target is stationary or moving sufficiently slowly, and it enables us to observe the same echo signal many times. For sufficiently strong echoes we can simply isolate the echoes and measure the TOA for each pulse transmitted. Then we need only subtract successive TOAs to find the lag. However, this approach is not optimal, and doesn't work at all when the echoes are hidden deep in the noise. We are thus led to seek a stronger technique, one that exploits more knowledge regarding the noise.

The only quantitative statement made about the additive noise ν_n was that it had zero mean. From one PRI to the next the desired signal x_n remains unchanged, but the received signal y_n is seems completely different from x_n , as depicted in Figure 5.5. Sometimes y_n is greater than x_n , but (due to the noise having zero mean) just as frequently it will be less. Mathematically, using the linearity of the expectation operator, we can derive $\langle y_n \rangle = \langle x_n + \nu_n \rangle = \langle x_n \rangle + \langle \nu_n \rangle = x_n$.



Figure 5.5: A pulsed radar signal contaminated by additive zero-mean noise. Note that from pulse to pulse the noise is different, but the pulse shape stays the same. Thus the uncontaminated signal can be reconstructed by pulse-to-pulse integration.

Hence, although in general the observed y_n is not the desired x_n , its average *is*. We can thus average the observed signals and obtain a much cleaner estimate of x_n . Such averaging over successive pulses is called radar return *integration*. With each new pulse transmitted, the true echo signal becomes stronger and stronger, while the noise cancels out and grows weaker and weaker. Even if the echo was initially completely buried in the noise, after sufficient averaging it will stand out clearly. Once detected, the lag measurement can be made directly on the average signal.

A similar operation can be performed for all periodic phenomena. When the desired underlying signal is periodic, each period observed supplies independent observations, and averaging increases the SNR. Another special case is slowly varying signals. Assuming the additive noise to be white, or at least containing significant spectral components at frequencies above those of x_n , we can average over adjacent values. The time domain interpretation of this operation is clear—since x_n varies more slowly than the noise, adjacent values are close together and tend to reinforce, while the higher-frequency noise tends to average out. The frequency domain interpretation is based on recognizing the averaging as being equivalent to a low-pass filter, which attenuates the high-frequency noise energy, while only minimally distorting the low-frequency signal. So once again just the zero mean assumption is sufficient to enable us to increase the SNR.

These averaging techniques can be understood using our naive theory, but take on deeper meaning in the more sophisticated treatment of noise. For example, we assumed that we could perform the averaging either in time or over separate experiments. This seemingly innocent assumption is known as the *ergodic hypothesis* and turns out to be completely nontrivial. We will return to these issues in Section 5.6.

EXERCISES

- 5.3.1 Generate M random ± 1 values and sum them up. The average answer will obviously be zero, but what is the standard deviation? Repeat for several different M and find the dependence on M.
- 5.3.2 In this exercise we will try to recover a constant signal corrupted by strong additive noise. Choose a number x between -1 and +1. Generate M random numbers uniformly distributed between -1 and +1 and add them to the chosen number, $s_n = x + \nu_n$. Now try to recover the chosen number by averaging over M values $\hat{x} = \sum_{n=1}^{M} s_n$ and observe the error of this procedure $x \hat{x}$. Perform this many times to determine the *average error*. How does the average error depend on M?

174 NOISE

5.3.3 Generate M sets of 1024 points of a sinusoidal signal corrupted by additive zero-mean noise,

$$s_n = \sin(\Omega t) + g\nu_n$$

where ν_n is uniform in the range [-1...+1]. Average s_n over the M sets to reduce the noise. Use $\Omega = 0.01$, g = 0.1, 1, 10 and M = 10, 100, 1000. How does the residual noise decrease as a function of M?

5.3.4 Using the same signal as in the previous exercise, replace each s_n value by the average

$$s_{n-L} + s_{n-L+1} + \ldots + s_{n-1} + s_n + s_{n+1} + \ldots + s_{n+L-1} + s_{n+L}$$

How well does this work compared to the previous exercise? Try $\Omega = 0.001$ and $\Omega = 0.1$. What can you say about time averaging?

5.4 Pseudorandom Signals

Although noise is often a nuisance we wish weren't there, we frequently need to generate some of our own. One prevalent motive for this is the building of simulators. After designing a new signal processing algorithm we must check its performance in the presence of noise before deploying it in the real world. The normal procedure (see Section 17.7) requires the building of a simulator that inexpensively provides an unlimited supply of input signals over which we exercise complete control. We can create completely clean signals, or ones with some noise, or a great deal of noise. We can then observe the degradation of our algorithm, and specify ranges of SNR over which it should work well.

We may also desire to generate noise in the actual signal processing algorithm. Some algorithms actually require noise to work! Some produce output with annoying features, which may be masked by adding a small amount of noise. Some are simply more interesting with probabilistic elements than without.

In this section we will discuss methods for generating random numbers using deterministic algorithms. These algorithms will enable us to use our familiar computer environment, rather than having to input truly probabilistic values from some special hardware. You undoubtably already have a function that returns random values in your system library; but it's often best to know how to do this yourself. Perhaps you checked your random number generator in the exercises of the previous section and found that it is not as good as you need. Or perhaps you are designing an embedded application that runs without the benefit of support libraries, and need an efficient noise generator of your own. Or maybe you are given the job of writing just such a library for some new DSP processor.

Before embarking on our exposition of random number generators there is a myth we must dispel. There is no such thing as a random number! If there is no such thing, then why are we trying to generate them? We aren't. What we are trying to generate are *random sequences* or, in DSP terminology, *random digital signals*. Each particular signal value, once generated, is perfectly well known. It's just that the connection *between* the signal values at different times is nontrivial and best described in probabilistic terms. Ideally one should not be able to guess the next value that the generator will produce based on the previous values (unless one knows the algorithm). Unfortunately, the term 'random number generator' has become so entrenched in popular computer science jargon that it would be futile to try to call it something else. You can safely use this term if you remember that these generators are not to be used to generate a single 'random' value; their proper use is always through generating large numbers of values.

There are several relatively good algorithms for generating random sequences of numbers, the most popular of which is the *linear recursion method*, originally suggested by D.H. Lehmer in 1951. This algorithm employs the integer recursion

$$x_{n+1} = (ax_n + b) \bmod m$$
 (5.11)

starting from some quite nonrandom initial integer x_0 . The integer parameters a, b, and m must be properly chosen for the scheme to work, for instance, by taking large m, and requiring b and m to be relatively prime, and a to be a large 'unusual' number. Real-valued random signals may be obtained by dividing all the integer values by some constant. Thus to create random real-valued signals in the range [0...1) one would probably simply use $\nu_n = \frac{x_n}{m}$, yielding quantized values with spacing $\frac{1}{m}$. Subtracting $\frac{1}{2}$ from this yields noise approximately symmetric around the zero.

The signals generated by equation (5.11) are necessarily periodic. This is because the present signal value completely determines the entire future, and since there are only a finite number of integer values, eventually some value must reoccur. Since apparent periodicity is certainly a bad feature for supposedly random signals, we wish the signal's period to be as long (and thus as unnoticeable) as possible. The longest period possible for the linear recursion method is thus the largest integer we can represent on our computer (often called MAXINT). 176

Long period is not enough. Taking a = 1, b = 1, and m = MAXINT gives us the sequence 1, 2, 3...MAXINT, which indeed only repeats after MAX-INT values, but hardly seems random. This is the reason we suggested that a be relatively large; this allows successive values to be widely separated. Keeping b and m relatively prime makes successive values as unrelated as possible. There is a lot more to say about optimal selection of these parameters, but instead of saying it we refer the reader to the extensive literature.

The implementation of equation (5.11) is quite problematic due to the possibility of overflow. Normally we desire m to be close to MAXINT, but then x may be quite large as well and ax + b would surely overflow. Choosing m to be small enough to prohibit overflow would be overly restrictive, severely limiting period length. In assembly language programming this may sometimes be circumvented by temporarily allocating a larger register, but this option is not available to the writer of a portable or high-level language routine. The constraints can be overcome by restructuring the computation at the expense of slightly increased complexity (in the following / represents integer division without remainder).

```
Given integers m, a, b, x

Precompute:

q \leftarrow m/a

r \leftarrow m-a*q

l \leftarrow m-b

Loop:

k \leftarrow x/q

x \leftarrow a*(x-q*k)-r*k-l

if x < 0 then x \leftarrow x+m
```

By the way, if what you want is random *bits* then it's not a good idea to generate random integers and extract the LSB. This is because a sequence of integers can appear quite random, even when its LSB is considerably less so. Luckily there are good methods for directly generating random bits. The most popular is the Linear Feedback Shift Register (LFSR), which is somewhat similar to linear recursion. A shift register is a collection of bits that can be shifted one bit to the right, thus outputting and discarding the LSB and making room for a new MSB. Linear feedback means that the new bit to be input is built by xoring together some of the bits in the shift register. Starting off with some bits in the shift register, we generate a sequence of bits by shifting to the right one bit at a time. Since the state of the shift register uniquely determines the future of the sequence, the sequence eventually become periodic. If the shift register ever has all zeros it becomes stuck in this state, and so this must be avoided at all costs.

One of the first random number generators was suggested by John von Neumann back in 1946. His method starts with some D digit integer. Squaring this integer produces an integer with 2D digits from which the next integer in the sequence is obtained by extracting the middle D digits. This recursion produces a periodic sequence of D digit integers, but this sequence will be considerably less random than one generated by a properly selected linear recursion generator.

Another random number generator does not require a multiplication, but does need more memory

$$x_{n+1} = (x_{n-j} + x_{n-k}) \bmod m$$

where j, k, and m need to be carefully chosen. Of course we need a buffer of length $\max(j, k)$, and must somehow initialize it.

Even if our random number generator turns out to be of inferior performance, there are ways to repair it. The most popular method is to use several different suboptimal generators and to combine their outputs in some way. For example, given three generators with different periods that output b bit integers, we can add the outputs or xor together their respective bits (an operation that is usually fast) and obtain a much better sequence. Given only two generators we can 'whiten' one by placing its values into a FIFO buffer and output a value from the buffer chosen by the second generator. This can even be accomplished by using a single suboptimal generator for both purposes. For example, assume that each call to 'random' returns a new pseudorandom real number between 0 and 1; then

```
Allocate buffer of length N
for i \leftarrow 1 to n
buffer<sub>i</sub> \leftarrow random
Loop:
k \leftarrow \texttt{floor}(n \texttt{random}) + 1
output buffer<sub>k</sub>
buffer<sub>k</sub> \leftarrow random
```

is more random, since it whitens short time correlations.

The algorithms we have discussed so far return uniformly distributed pseudorandom numbers. In practice we frequently require pseudorandom numbers with other distributions, most frequently Gaussian. There are two popular ways of generating Gaussian noise given a source of uniformly distributed noise. The first relies on the 'law of large numbers' (see Appendix A.13) that states that the sum of a large number of independent random numbers, whatever their original distribution, will tend to be Gaussianly distributed. To exploit this law requires generating and adding N (even 12 is often considered large enough) uniform random numbers. Of course the maximum value that can be obtained is N times the maximum value of the uniform generator, so in reality the Gaussian is somewhat truncated, but the true distribution is extremely small there anyway. Often of more concern is the computational burden of computing N uniform random numbers per Gaussian random required.

The second method commonly used to generate Gaussianly distributed numbers, sometimes called the Box-Muller algorithm after its inventors, is best understood in steps. First pick at random a point inside the unit circle, $x + iy = re^{i\theta}$. If we selected the point such that x and y are independent (other than the constraint that the point be inside the circle) then r and θ will be as well. Now θ is uniformly distributed between 0 and 2π ; how is r distributed? It is obvious that larger radii are more probable since the circumference increases with radius; in fact it is quite obvious that the probability of having a radius between zero and r increases as r^2 . We now create a new point in the plane u + iv, whose angle is θ but with radius ρ that obeys $r^2 = e^{-\rho^2/2}$. The probability of such a point having radius less than R is the same as the probability that the original squared radius r^2 is greater than $e^{-R^2/2}$. From this it follows that u and v are Gaussianly distributed.

How do we select a point inside a circle with all points being equally probable? The easiest way is to randomly pick a point inside the square that circumscribes the unit circle, and to discard points outside the circle. Picking a point inside a square involves independently generating two uniformly distributed random numbers x and y. Since u and v are also independent, for every two uniform random numbers that correspond to a point inside the circle we can compute two Gaussianly distributed ones.

Thus we arrive at the following efficient algorithm:

```
generate two uniform random numbers between -1 and +1, x and y
r^2 \leftarrow x^2 + y^2
if r^2 > 1 return to the beginning
\rho^2 \leftarrow -2\ln r^2, c \leftarrow \frac{\rho}{r}
u \leftarrow cx and v \leftarrow cy
```

EXERCISES

- 5.4.1 Not only isn't there such a thing as a random number, there really is no such thing as a random sequence of finite length. For example, all sequences of ten digits are equally probable, namely one chance in 10^{10} . Yet we feel viscerally that sequences such as $\{1, 1, 1, 1, 1, 1, 1, 1\}$ or $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ are less random than say $\{1, 9, 3, 6, 3, 4, 5, 8, 2\}$. Can you explain this feeling?
- 5.4.2 You can test a random function using the following graphical test. Generate successive values r_1, r_2, \ldots and make a scatter plot consisting of points (r_k, r_{k-1}) . If the resulting picture has structure (e.g., noticeable lines) the random sequence has short-term correlations. If the plot looks reasonably homogeneous repeat the procedure but plot (r_k, r_{k-m}) instead. Test the integer recursions (equation (5.11)) defined by a=10, b=5, m=50; a=15625, b=0, m=65536; and the generator supplied with your programming environment.
- 5.4.3 Take inferior random generators from the previous exercise and whiten them using the algorithm given in the text. Perform the graphical test once again.
- 5.4.4 Code a Gaussian noise generator based on the law of large numbers and check its distribution.
- 5.4.5 Some people use this algorithm to generate Gaussianly distributed numbers: generate two uniform random numbers, x and y, between 0 and +1 $a = \sqrt{-2 \ln x}, \phi = 2\pi y$ $u \leftarrow a \sin(\phi)$ and $v \leftarrow a \cos(\phi)$ Is this algorithm correct? What are the advantages and disadvantages relative to the algorithm given in the text?
- 5.4.6 Other people use the following algorithm: generate two uniform random numbers, x and y, between 0 and +1 $u = \sqrt{-2 \ln x} \sin(2\pi y)$ $v = \sqrt{-2 \ln x} \cos(2\pi y)$ Show that this method is mathematically equivalent to the method given

show that this method is mathematically equivalent to the method given in the text. In addition to requiring calls to sine and cosine functions, this method is numerically inferior to the one given in the text. Why?

- 5.4.7 Complete the proof of the second algorithm for generating Gaussianly distributed random numbers.
- 5.4.8 How can we generate random numbers with an arbitrary distribution given a uniform generator?
- 5.4.9 Show that after an initial transient LFSR sequences are always periodic. What is the maximal period of the sequence from a shift register of length K? Find a maximal length LFSR sequence of length 15.

5.5 Chaotic Signals

Completely specified deterministic signals, that is, signals generated by completely specified deterministic algorithms, can still appear to be entirely random and chaotic. The word 'chaos' comes from the Greek $\xi\alpha\sigma\sigma$, the most ancient of the gods, and refers to the confused primordial state before the creation. The study of chaotic signals is quite the reverse; what can be fruitfully examined is the route taken from orderly (often periodic) behavior to the chaotic. Most of this section will be devoted to the study of the transition from periodic to chaotic behavior in the simplest possible setting.

How can deterministic signals exhibit chaotic behavior? Turbulence of rapidly flowing liquids is one of the prototypes of chaos; although the equations of fluid dynamics are well known, we cannot predict the exact behavior of twisting currents and whirlpools. When the flow is slow the behavior *is* understandable, so we can start with a slowly flowing liquid and gradually increase the flow until chaos sets in. Similarly, the future value of investments may become unpredictable when interest rates are high and the market volatile, but such prediction is straightforward under more subdued conditions. One can forecast the weather for the next day or two when conditions are relatively stable, but prediction becomes impossible over longer periods of time.

There is a simple mathematical explanation for the appearance of chaos in a deterministic setting. Linear equations (whether algebraic, differential, or difference) have the characteristic that small changes in the input lead to bounded changes in output. Nonlinear equations do not necessarily have this attribute. In fact it is known that for nonlinear equations with three or more free parameters there always are values of these parameters for which infinitesimally small changes in initial conditions lead to drastic changes of behavior. Even one or two parameter nonlinear equations may become oversensitive. Such equations are said to exhibit chaotic behavior since our knowledge of the initial conditions is never sufficient to allow us to predict the output far enough from the starting point. For example, we may be able to predict tomorrow's weather based on today's, but the fundamental equations are so sensitive to changes in the temperature and air pressure distributions that we have no chance of accurately predicting the weather next week.

Perhaps the simplest example of knowledge loss is the shift and truncate recursion

$$x_{n+1} = \operatorname{Trunc}\left(10x_n\right) \tag{5.12}$$

which shifts the signal value's decimal point to the right, and then removes the integer part. The first few values starting with $x_0 = \pi - 3$ are

 $0.1415926535\ldots, 0.4159265358\ldots, 0.1592653589\ldots, 0.5926535897\ldots, 0.9265358979\ldots, 0.2653589793\ldots$

which seem to oscillate wildly over the unit interval. Had we chosen x_0 slightly different from $\pi - 3$, the deviation of the resulting x_n from the above values would exponentially increase; for example, with a difference of 10^{-5} all similarity is lost after only five iterations.

The weather prediction example is similar. It turns out that the equations relating air pressure, temperature, wind velocity, etc. are highly nonlinear, even for rather simplistic models of atmospheric conditions. Weather prediction relies on running such models, with appropriate initial weather conditions, on large computers and observing the resulting weather conditions. The initial specification is rather coarsely defined, since only gross features such as average air temperature and pressure are known. This specification leads to specific predictions of the weather as a function of time. However, slight changes in the specification of the initial weather conditions lead to rather different predictions, the differences becoming more and more significant as time goes on. This is the reason that the weather can be predicted well for the short term, but not weeks in advance. Lorenz, who discovered the instability of weather prediction models in the early 1960s, called this the 'butterfly effect'; a butterfly flapping its wings in Peking will affect the weather in New York a month later!

How can we hope to study such nonlinear equations? Isn't chaos by definition incomprehensible and thus unresearchable? The trick is to study *routes to chaos*; we start at values of parameters for which the nonlinear equations are *not* chaotic, and then to vary the parameters in order to approach the chaotic region. Before entering the chaotic region, the output signal, although increasingly bizarre, *can* be profitably investigated. In this section we will study Feigenbaum's route to chaos. This route is easy to study since it occurs in a simple one-parameter setting, arguably the simplest nonlinear equation possible. It also seems to model well many interesting physical situations, including some of the examples mentioned above.

We'll introduce Feigenbaum's route with a simple example, that of fish in a closed pond. Let us denote by x the present fish population divided by the maximum possible population (thus $0 \le x \le 1$). We observe the population every day at the same hour, thus obtaining a digital signal x_n . How does x_n vary with time? Assuming a constant food supply and a small initial number



Figure 5.6: The logistics recursion relates the new signal value x_{n+1} to the old one x_n by a inverted parabola. As such it is the simplest nonlinear recursion relation. It can also be used to approximate any recursion with a single smooth maximum.

of fish, we expect an initial exponential increase in population,

$$x_{n+1} = r \, x_n$$

but once the number of fish becomes large, we anticipate an opposite tendency due to overpopulation causing insufficient food and space, and possibly spread of disease. It makes sense to model this latter tendency by a $1-x_n$ term, since this leads to pressure for population decrease that is negligible for small populations, and increasingly significant as population increases. Thus we predict

$$x_{n+1} = r \, x_n \, (1 - x_n) \tag{5.13}$$

which is often called the *logistics equation*. This equation is quadratic (see Figure 5.6) and thus nonlinear. It has a single free parameter r (which is related to the amount we feed the fish daily), which obey $0 \le r \le 4$ in order for the signal x to remain in the required range $0 \le x \le 1$. Although a nonlinear equation with one free parameter is not guaranteed to be chaotic, we will see that there *are* values of r for which small changes in x_0 will lead to dramatic changes in x_n for large n. This means that when we overfeed there will be large unpredictable fluctuations in fish population from day to day.

You may object to studying in depth an equation derived from such a fishy example. In that case consider a socialistic economy wherein the state wishes to close the socioeconomic gap between the poor and the wealthy. It is decided to accomplish this by requiring everyone to deposit their money in a state-controlled bank that pays lower interest rates to the wealthy. Let y_n be the amount invested as a function of time, y_{max} the maximum wealth allowed by law, and *i* the applicable interest. The usual financial formulas tell us $y_{n+1} = (1+i)y_n$, but here *i* must be a decreasing function of *y*, which we take to be $i = i_0(1 - \frac{y_n}{y_{max}})$. Substitution leads to

$$y_{n+1} = \left(1 + i_0 \left(1 - \frac{y_n}{y_{max}}\right)\right) y_n$$

which by a simple change of variables becomes the logistics equation (5.13).

We could continue to give examples that lead to the same equation. It is so ubiquitous simply because it is the simplest nonlinear recursion relation for a single bounded signal that contains a single free parameter. Any time we obtain a quadratic relationship we can transform it into the logistics equation by ensuring that the variable is constrained to the unit interval; indeed any time we have any nonlinear recursion with a single smooth maximum we can approximate it by the logistics equation in the vicinity of the maximum.

Now that we are convinced that such a study is worthwhile, let us embark upon it. We expect the signal x_n to eventually approach some limiting value, i.e., that the number of fish or the amount of money would approach a constant for long enough times. This is indeed the case for small enough rvalues. To find this value as a function of r we need to find a *fixed point* of the recursion, that is, a signal value that once attained forever remains unchanged. Since $x_{n+1} = f(x_n)$ must equal x_n , we conclude that a fixed point x_1 must obey the following equation.

$$x_1 = f(x_1) = r \, x_1 \, (1 - x_1) \tag{5.14}$$

Zero is obviously a fixed point of the logistics equation since $x_n = 0$ implies $x_{n+1} = rx_n(1 - x_n) = 0$ as well. When you have no fish at all, none are born, and an empty bank account doesn't grow. Are there any nontrivial fixed points? Solving equation (5.14) we find the nonzero fixed points are given by $x_1 = p_r \equiv 1 - \frac{1}{r}$.

For this simplest of recursions we could algebraically find the fixed points with little trouble. For more complex cases we may fall back to a graphical method for finding them. In the graphical method you first plot the recursion function $x_{n+1} = f(x_n)$ (with x_n on the x axis and x_{n+1} on the y axis). Then you overlay the identity line $x_{n+1} = x_n$. Fixed points must correspond to intersections of the recursion plot with the identity line. In our case the recursion is an inverted parabola, and we look for its intersections with the



Figure 5.7: Graphical method of finding fixed points of the logistics equation. From bottom to top the inverted parabolas correspond to r = 0, 1, 2, 3, 4. We see that for r < 1 the parabola intersects the identity line only at x = 0, while for larger r there is an additional point of intersection.

45° line (Figure 5.7). It is easy to see that for the parameter region $0 \le r \le 1$ the only possible fixed point is $x_0 = 0$, but for r > 1 the new fixed point p_r appears. For $r \gtrsim 1$ the new fixed point p_r is close to the old one (zero), gradually moving away with increasing r.

So we have found that the steady state behavior of the recursion is really very simple. For r < 1 we are underfeeding our fish, or the interest is negative, and so our fish or money disappear. An example of this behavior is displayed in Figure 5.8.A. For 1 < r < 3 the number or fish or amount of money approaches a constant value as can be seen in Figure 5.8.B. However, we are in for quite a shock when we plot the behavior of our fish or money for r > 3 (Figures 5.8.C and 5.8.D)! In the first case the signal oscillates and in the second it seems to fluctuate chaotically, with no possibility of prediction. In the chaotic case starting at a slightly different initial point produces a completely different signal after enough time has elapsed! We don't yet understand these phenomena since we progressed along the route to chaos too quickly, so let's backtrack and increase r more slowly.

The most important feature of the behavior of the signal for small r is the existence of the fixed point. Fixed points, although perhaps interesting, are not truly significant unless they are *attractive*. An attractive fixed point is one that not only replicates itself under the recursion, but *draws in* neighboring values as well. For $r \leq 1$ the zero fixed point is attractive—no matter where we start we rapidly approach x = 0; but for 1 < r < 3 the new fixed



Figure 5.8: Signal produced by recursion of the logistics equation for different values of r. In (A) we have r = 0.9 and the signal decays to zero. In (B) we have r = 1.9 and the signal approaches a constant value. (C) depicts r = 3.1 and the signal oscillates between two values. In (D) we have r = 4 with two slightly different initial states; the signal is noise-like and irreproducible.

point 'draws in' all signals that do not begin with $x_0 = 0$ or $x_0 = 1$. This is hinted at in Figure 5.8, but it is both simple and instructive for the reader to experiment with various r and x_0 and become convinced.

The idea of attraction can be made clear by using the 'return map', which is a graphical representation of the dynamics. First, as before, we plot the recursion $x_{n+1} = f(x_n)$ and the 45° line $x_n = x_{n-1}$. We start with a point on the line (x_0, x_0) . Now imagine a vertical line that intersects this point; it crosses the recursion curve at some point x_1 . We draw the vertical line from (x_0, x_0) to this new point (x_0, x_1) . Next we imagine a horizontal line that intersects this new point. It crosses the 45° line at (x_1, x_1) , and we proceed to draw a horizontal line to there. The net result of the previous two operations is to draw two lines connecting (x_0, x_0) to (x_1, x_1) , corresponding to one iteration from x_0 to x_1 .

We can now continue to iterate (as in Figure 5.9) until an attractor is found. In part (A) of that figure we see that when r < 1 (no matter where we begin) we converge to the zero fixed point. Part (B) demonstrates that when 1 < r < 3, we almost always converge on the new fixed point (the exceptions being $x_0 = 0$ and $x_0 = 1$, which remain at the old zero fixed



Figure 5.9: Use of return maps to depict the dynamics of a simple recursion. Each iteration starts on the 45° line, proceeds vertically until intersecting the recursion curve, and then returns to the diagonal line. Here we see that after enough iterations we converge on a fixed point, which is the intersection of the recursion curve with the diagonal line. In (A) we have r = 0.5 and the only fixed point is zero, while in (B) we see convergence to the nonzero fixed point p_r .

point). We will see shortly that for r > 3 even this fixed point ceases being an attractor; if one starts exactly at it, one stays there, but if one strays even slightly the recursion drives the signal away.

How can we mathematically determine if a fixed point p is an attractor? The condition is that the absolute value of the derivative of the recursive relation f must be less than unity at the fixed point.

$$\left|\frac{df(x)}{dx}|_{x=p}\right| < 1 \tag{5.15}$$

This ensures that the distance from close points to the fixed point *decreases* with each successive recursion. It is now easy to show that for $r > r_2 = 3$ the fixed point p_r becomes unattractive; but what happens then? No new fixed point can appear this time, since the reasoning that led to the discovery of p_r as the sole nonzero fixed point remains valid for all r! To see what happens we return to the return map. In Figure 5.10.A we see that starting from some initial point we approach a 'square', which translates to alternation between two points. Once the signal reaches its steady state it simply oscillates back and forth between these two values, as can be seen in Figures 5.10.B and 5.11. This dual-valued signal is the new attractor; unless we start with $x_0 = 0, 1, 1 - \frac{1}{r}$ or $f^{-1}(1 - \frac{1}{r})$ we eventually oscillate back and forth between two values the distance between the two values that make up this attractor also increases.

So attractors can be more complex than simple fixed points. What happens when we increase r still further? You may have already guessed that this two-valued attractor also eventually becomes unattractive (although if



Figure 5.10: Return map representation of the logistics equation for r > 3. In (A) r = 3.2 and we see that from an arbitrary initial state x_0 we converge on a 'non fixed point' attractor close to p_r . The attractor contains two points, one on either side of p_r . In (B) r = 3.4 and we display only the long time behavior (steady state behavior after the transient has died down).



Figure 5.11: The signal resulting from recursion of the logistics equation for r = 3.2. In (A) we see the steady state signal in the time domain. It oscillates between the two values that make up the attractor, which means that $x_{n+1} = f(x_n)$ and $x_{n+2} = f(x_{n+1}) = x_n$. In (B) we see the same signal in the frequency domain. The DC component represents the nonzero average of the two points. Since the signal oscillates at the maximum possible frequency, we have a spectral line at digital frequency $\frac{1}{2}$.

one starts at *exactly* one of its points one stays trapped in it) and a new more complex attractor is born. In this case, this happens at $r_3 = 1 + \sqrt{6}$ and the new attractor is composed of a cycle between four signal values, as depicted in Figure 5.12. If we call these points a_1 , a_2 , a_3 , and a_4 , the requirement is $a_2 = f(a_1)$, $a_3 = f(a_2)$, $a_4 = f(a_3)$, and $a_1 = f(a_4)$. Note that the 2-cycle's a_1 split up into our present a_1 and a_3 , while its a_2 became our new a_2 and a_4 . So the attractor's components obey $a_1 < a_3 < a_2 < a_4$, which



Figure 5.12: The return map, signal, and spectrum for the steady state behavior when r = 3.5. The attractor is a 4-cycle.



Figure 5.13: The return map, signal, and spectrum for the steady state behavior when r = 3.55. The attractor is a 8-cycle.



Figure 5.14: The return map, signal, and spectrum for the steady state behavior when r = 3.5675. The attractor is a 16-cycle.

NOISE

means that the closest together in time are the farthest in space and vice versa. This induces a spectrum wherein an additional spectral line appears at twice the period, or half the frequency of the previous line.

We saw that when r increased above r_3 each component of the 2-cycle splits into two, just as the fixed point had earlier split. The same thing happens in turn for the 4-cycle when r goes above r_4 and an 8-cycle is born. The critical feature is that at each stage all components of the present attractor become unattractive simultaneously, a phenomenon known as *pitchfork bifurcation*. Due to the bifurcation, with increasing r we find 16-cycles, 32cycles, and all possible 2^n -cycles. Examples of such cycles are depicted in Figures 5.12 through 5.14. The rule of 'closest in time are farthest in space' continues to be obeyed, so that new spectral lines continue to appear at harmonics of half the previous basic frequency. Eventually the lines are so close together that the spectrum becomes white, and we have chaotic noise.

The transition from periodicity to chaos can best be envisioned by plotting the attractors as a function of r, as in Figure 5.15. The transition points r_n as a function of n approach a limit

$$r_n \xrightarrow{n \to \infty} r_\infty \approx 3.57$$

so that the regions where these cycles exist become smaller and smaller. By the time we reach r_{∞} we have finished all the 2^n -cycles.



Figure 5.15: The attractors of the recursion as a function of r. Observe the zero attractor for 0 < r < 1, the fixed point p_r for 1 < r < 3, the 2-cycle for $3 < r < 1 + \sqrt{6}$, and the 2^n -cycles for $3 < r < r_{\infty}$. Certain odd cycle regions can also be clearly seen for $r > r_{\infty}$. At r = 4 chaos reigns.



Figure 5.16: Non 2^n -cycle attractors for $r > r_\infty$. We present return maps for a 3-cycle (r = 3,83), a 6-cycle that results from bifurcation of that 3-cycle (r = 3.847), a quite different 6-cycle (r = 3.63), and a 5-cycle (r = 3.74).

What happens between here and r = 4? It turns out that every length attractor is possible. For example, in Figure 5.16 we see 3-cycles, 5-cycles and 6-cycles. There is a theorem due to Šarkovskii that states that the order of first appearance of any given length is

$$1, 2, 4, 8, \cdots 2^k, \cdots 2^k \cdot 9, 2^k \cdot 7, 2^k \cdot 5, 2^k \cdot 3, \cdots 4 \cdot 9, 4 \cdot 7, 4 \cdot 5, 4 \cdot 3, \cdots 9, 7, 5, 3$$

so that once a 3-cycle has been found we can be certain that all cycle lengths have already appeared.

For $r_{\infty} < r \leq 4$ there are other types of behavior as well. Let us start at r = 4 where all possible x values seem to appear chaotically and decrease r this time. At first x seems to occupy the entire region between $\frac{r}{4}$ and $r(1-\frac{r}{4})$, but below a certain r'_1 this band divides into two broad subbands. The signal always oscillates back and forth between the two subbands, but where it falls in each subband is unpredictable. Decreasing r further leads us past r'_2 where each subband simultaneously splits into two somewhat narrower subbands. The order of jumping between these four subbands is 'closest in time are farthest in space', but the exact location inside each subband is chaotic. Decreasing further leads us to a cascade of r'_n in between which there are 2^n chaotic subbands, a phenomenon known as 'reverse bifurcation'. Interspersed between the reverse bifurcations are regions of truly periodic behavior (such as the 3-, 5-, and 6-cycles we saw before). The r'_n converge precisely onto r_{∞} where the reverse bifurcations meet the previous bifurcations.

We have seen that the simplest possible nonlinear recursion generates an impressive variety of periodic and chaotic signals; but although complex, these signals are still deterministic. In the next section we will see what a truly random signal is.

EXERCISES

- 5.5.1 What is the change of variables that converts the socialistic economy equation into the fish pond one?
- 5.5.2 Write a simulator that graphically depicts the behavior of the signal generated by the logistics equation (5.13). Vary r and by trying various starting points identify the attractors in the different regions.
- 5.5.3 Write a program to plot the attractors as a function of r. For each r go through the possible x_0 systematically and identify when periodic behavior has been reached and plot all points in this attractor. Can you identify the various regions discussed in the text?
- 5.5.4 Extend the simulator written above to display the spectrum as well. Reproduce the results given in the text.
- 5.5.5 Prove that equation (5.15) is indeed the criterion for attractiveness. Prove that for $r \leq 1$ zero is indeed an attractor. Prove that for $1 the fixed point <math>p_r$ is an attractor.
- 5.5.6 At r = 4 a change of variable

$$x = \frac{1}{2}(1 - \cos 2\pi\theta)$$

brings us to a variable θ , which is homogeneously distributed. Show that x is distributed according to $\frac{1}{\sqrt{x(1-x)}}$.

- 5.5.7 Plot the signal and spectrum of the 3-, 5-, and 6-cycles.
- 5.5.8 Henon invented a number of area preserving two-dimensional chaotic signals, for example,

$$x_{n+1} = x_n \cos \alpha - (y_n - x_n^2) \sin \alpha$$

$$y_{n+1} = x_n \sin \alpha + (y_n - x_n^2) \cos \alpha$$

which is dependent on a single parameter α , which must obey $0 \le \alpha \le \pi$. Show that the origin is a fixed point, that large x diverge to infinity, and that there is a symmetry axis at angle $\alpha/2$. Are there any other fixed points?

5.5.9 Write a program to plot in the plane the behavior of the Henon map for various α . For each plot start from a large number of initial states (you can choose these along the 45° line starting at the origin and increasing at constant steps until some maximal value) and recurse a large number of times. What happens at $\alpha = 0$ and $\alpha = \pi$? Increase α from zero and observe the behavior. For what α are 'islands' first formed? Zoom in on these islands. When are islands formed around the islands? Observe the sequence of *multifurcations*. When is chaos achieved?

5.6 Stochastic Signals

In this section we will briefly introduce the formal theory of stochastic signals. This topic is a more advanced mathematically, and we assume the reader has a working knowledge of basic probability theory (see Appendix A.13). Since a full discussion of the theory would require a whole book we will have to content ourselves with presenting only the basic terminology.

Recall that a signal is *deterministic* if we can precisely predict its value at any time; otherwise it is *stochastic*. What do we mean by a random signal? By our original definition a signal must be precisely defined for all times, how can it be random? When we speak of a random signal in the formal sense, we are actually not referring to a single signal at all, but to an infinite number of signals, known as an *ensemble*. Only individual realizations of the ensemble can be actually observed, (recall Figures 5.1, 5.2, and 5.3) and so determining a signal value requires specification of the realization in addition to the time. For this reason many authors, when referring to the entire ensemble, do not use the term 'signal' at all, prefering to speak of a *stochastic process*.

Often a stochastic signal is the sum of a deterministic signal and noise.

$$s^{r}(t) = x(t) + \nu^{r}(t)$$
 A D $s^{r}_{n} = x_{n} + \nu^{r}_{n}$ (5.16)

Here the superscript r specifies the specific realization; for different r the deterministic component is identical, but the noise realization is different. Each specific realization is a signal in the normal sense of the word; although it might not be possible to find an explicit equation that describes such a signal, it nonetheless satisfies the requirements of signalhood, with the exception of those we usually ignore in any case. While taking a specific r results in a signal, taking a specific time t or n furnishes a random variable, which is a function of r over the real numbers that can be described via its probability distribution function. Only when both r and t or n are given do we get a numeric value; thus a stochastic signal can be better described as a function of two variables.

When describing a stochastic signal we usually specify its *ensemble statistics*. For example, we can average over the ensemble for each time, thus obtaining an average as a function of time. More generally, for each time we can find the Probability Density Function (PDF) of the ensemble.

ı.

$$\begin{aligned} f(t,s)ds &= \\ \operatorname{Prob}(s \leq s(t) \leq s + ds) \end{aligned} \quad \mathbf{A} \quad \mathbf{D} \qquad \begin{aligned} f_n(s)ds &= \\ \operatorname{Prob}(s \leq s_n \leq s + ds) \end{aligned} \quad (5.17) \end{aligned}$$

Here, even for the digital case we assumed that s was unquantized, $f_n(s)ds$ representing the probability that the signal value at time n will be between s and s+ds. Note that unlike the statistics of a regular random variable, statistics of stochastic signals are functions of time rather than simple numbers. Only for the special case of *stationary signals* are these statistics constants rather than time-dependent.

As for regular random variables, in addition to the density function f(s)we can define the Cumulative Distribution Function (CDF)

$$F(t,s) = \operatorname{Prob}(s(t) \le s)$$
 A D $F_n(s) = \operatorname{Prob}(s_n \le s)$ (5.18)

and it is obvious that for every time instant the density is the derivative of the cumulative distribution. These distribution functions are in practice cumbersome to use and we usually prefer to use statistics such as the mean and variance. These can be derived from the density or cumulative distribution. For example, for analog signals the mean for all times is calculated from the density by integration over all x values

$$m_s(t) = \left\langle s(t) \right\rangle = \int_{-\infty}^{\infty} s(t) f(t,s) \, ds \tag{5.19}$$

and the variance is as one expects.

$$\sigma_s^2(t) = \int_{-\infty}^{\infty} \left(s(t) - m_s(t) \right)^2 f(t, s) \, ds \tag{5.20}$$

There are also other statistics for stochastic signals, which have no counterpart for simple random variables. The simplest is the correlation between the signal at different times

$$C_{s}(t_{1},t_{2}) = \left\langle s(t_{1})s(t_{2})\right\rangle \qquad \mathbf{A} \quad \mathbf{D} \qquad C_{s}(n_{1},n_{2}) = \left\langle s_{n_{1}}s_{n_{2}}\right\rangle \quad (5.21)$$

which for stationary signals is a function only of the time difference $\tau = t_1 - t_2$ or $m = n_1 - n_2$.

$$C_s(\tau) = \left\langle s(t)s(t-\tau) \right\rangle$$
 A D $C_s(m) = \left\langle s_n s_{n-m} \right\rangle$ (5.22)

This correlation is usually called the *autocorrelation*, since there is also a crosscorrelation between two distinct signals $C_{xy}(t_1, t_2) = \langle x(t_1)y(t_2) \rangle$. The autocorrelation tells us how much the value of the signal at time t_1 influences its value at time t_2 , and is so important that we will devote all of Chapter 9 to its use. The single time variance is simply the autocorrelation when the two time variables coincide $\sigma_s^2(t) = C_s(t,t)$, and so for stationary signals $\sigma_s^2 = C_s(0)$. We also sometimes use the *autocovariance* $V_s(t_1, t_2) = \langle (s(t_1) - m_s(t_1)) (s(t_2) - m_s(t_2)) \rangle$ and it is easy to show that $V_s(t_1, t_2) = C_s(t_1, t_2) - m_s(t_1) m_s(t_2)$.

More generally we can have statistics that depend on three or more time instants. Unlike single time statistics, which can be calculated separately for each time, these multitime statistics require that we simultaneously see the entire stochastic signal (i.e., the entire ensemble for all times). We often use only the mean as a function of time and the correlation as a function of two times. These are adequate when the probability density function for all times is Gaussian distributed, since Gaussians are completely defined by their mean and variance. For more general cases *higher-order signal processing* must be invoked (see Section 9.12), and we define an infinite number of *moment functions*

$$M_s(t_1, t_2, \dots, t_k) \equiv \left\langle s(t_1)s(t_2)\cdots s(t_k) \right\rangle$$
(5.23)

that should not be confused with 'statistical moments'

$$m_s = \left\langle t^s s(t) \right\rangle$$

which are simply numbers. A stochastic signal is said to be 'stationary to order k' if its moments up to order k obey

$$M_s(t_1, t_2, \ldots, t_k) = M_s(t_1 + \tau, t_2 + \tau, \ldots, t_k + \tau)$$

and stationarity implies stationarity to order k for all finite k.

A few concrete examples will be helpful at this point. An analog *Markov* signal is a stationary, zero mean stochastic signal for which the autocorrelation dies down exponentially.

$$C(t_1, t_2) = rac{e^{-rac{|t_2 - t_1|}{ au}}}{ au}$$

Thus there is essentially only correlation between signal values separated by about τ in time; for much larger time differences the signal values are essentially uncorrelated. When τ approaches zero we obtain white noise, which thus has a delta function autocorrelation.

$$C(t_1, t_2) = \sigma^2 \delta(t_1 - t_2)$$

This means that for any two distinct times, no matter how close these times are, there is no correlation at all between signal values of white noise.

For discrete time signals we define Markov signals of different orders. A first-order Markov signal is one for which s_n depends on s_{n-1} but not directly on any previous value. A second-order Markov signal has the signal value depending on the two previous values.

There is an important connection between white noise and Markov signals; A Markov signal s_n can be generated by filtering white noise ν_n . We cannot fully explain this result as our study of filters will only begin in the next chapter, but the main idea can be easily understood. Signal values of white noise at different times can be independent because of the high-frequency components in the noise spectrum. Filtering out these high frequencies thus implies forcing the signal value at time n to depend on those at previous instants. A particular type of low-pass filtering produces precisely Markovian behavior.

$$s_n = \alpha s_{n-1} + \nu_n \tag{5.24}$$

Low-pass filtering of white noise returns us to a Markov signal; bandpass filtering results in what is often called 'colored noise'. These signals have nonflat power spectra and nondelta autocorrelations.

Note that although we often use Gaussian white noise, these two characteristics are quite independent. Noise can be white without being Gaussian and vice versa. If for any two times the signal is uncorrelated, and all moments above the second-order ones are identically zero, we have Gaussian white noise. However, when the signal values at any two distinct times are statistically independent, but the distributions although identical at all times are not necessarily Gaussian, we can only say that we have an Independent Identically Distributed (IID) signal. Conversely, when there are correlations between the signal values at various times, but the joint probability function of n signal values is n-dimensional Gaussian, then the signal is Gaussian noise that is not white.

Stochastic signals are truly complex, but it is reassuring to know the most general stationary stochastic signal can be built from the elements we have already discussed. In the 1930s Wold proved the following theorem.

Theorem: Wold's Decomposition

Every stationary stochastic signal s can be written

$$s_n = x_n + \sum_{m=0}^{\infty} h_m w_{n-m}$$

as the sum of a deterministic signal x and filtered white noise.

In addition to the ensemble statistics we have been discussing, there is another type of statistics that can be computed for stochastic signals, namely *time statistics*. For these statistics we consider a single realization and average over the time variable, rather than hold the time constant and averaging over the ensemble. Thus the time average of a signal s at time zero is

$$\langle s \rangle = \frac{1}{T} \int_{\tau - \frac{T}{2}}^{\tau + \frac{T}{2}} s(t) dt \qquad \mathbf{A} \qquad \mathbf{D} \qquad \langle s \rangle = \frac{1}{N} \sum_{n = \nu - \frac{N}{2}}^{\nu + \frac{N}{2}} s_n \qquad (5.25)$$

• •

where T or N are called the 'integration windows'. This type of averaging is often simpler to carry out than ensemble averaging since for s(t) and s_n we can use any realization of the signal s that is available to us, and we needn't expend the effort of collecting multiple realizations. When we previously suggested combating noise for a narrow-band signal by averaging over time, we were actually exploiting time statistics rather than ensemble statistics.

What is the connection between ensemble statistics and time statistics? In general, there needn't be any relation between them; however, we often assume a very simple association. We say that a signal is *ergodic* if the time and ensemble statistics coincide. The name 'ergodic' has only historical significance, deriving from the 'ergodic hypothesis' in statistical physics that (wrongly) posited that the two types of statistics must always coincide. To see that in general this will not be the case, consider the ensemble of all different DC signals. The ensemble average will be zero, since for every signal in the ensemble there is another signal that has precisely the opposite value. The time average over any one signal is simply its constant value, and *not* zero! A less trivial example is given by the digital sinusoid

$$s_n = A\sin(\omega n)$$

with A chosen in the ensemble with equal probability to be either 1 or -1. Here both the ensemble and time averages are zero; but were we to have chosen A to be either 0 or 1 with equal probability, then the time average would remain zero, while the ensemble average becomes the time-dependent $\frac{1}{2}\sin(\omega n)$.

What does ergodicity really mean? Simply that rather than acquiring an ensemble of N signal generators we can use only a single generator but restart our experiment N times. If the signal with all the possible different initial times reproduces the entire ensemble of the stochastic signal, then the signal is ergodic. Not only must all possible realizations be reproduced, they must be reproduced the same number of times. When we thinking about it this way, ergodicity is rather too strong a statement; no signal can really be so random that a single realization completely samples all the possibilities of the ensemble! The number of realizations generated by restarting the experiment at all possible times equals the number of points on the real line, while there are many more different functions of time! However, ergodicity makes life so simple that we most often assume it anyway.

For ergodic signals we can redefine the correlations in terms of time averages. For example, the autocorrelation becomes

$$C_s(\tau) = \int s(t)s(t-\tau)dt \qquad \mathbf{A} \quad \mathbf{D} \qquad C_s(m) = \sum_n s_n s_{n-m} \qquad (5.26)$$

and it is these forms that we shall use in Chapter 9.

EXERCISES

- 5.6.1 Consider a stationary signal that can only take the values 0 and 1. What is the probability that the signal is nonzero at two times τ apart? What is the meaning of moments for this type of signal?
- 5.6.2 Derive the relation between autocovariance and autocorrelation $V_s(t_1, t_2) = C_s(t_1, t_2) m_s(t_1)m_s(t_2)$.
- 5.6.3 Show that for white signals (for which all times are independent) the autocovariance is zero except for when $t_1 = t_2$.
- 5.6.4 In the text we discussed the filtering of white noise, although white noise is not a signal and thus we have never properly defined what it means to filter it. Can you give a plausible meaning to the filtering of a stochastic signal?
- 5.6.5 Prove that a first-order Markov signal can be obtained by low-pass filtering white noise. Assuming that s_n is created from a noise signal ν_n by equation 5.24 with $|\alpha| < 1$, what is the probability distribution of s_n given that we already observed s_{n-1} ?
- 5.6.6 Show that the signal s_n generated by the recursion $s_n = \alpha_1 s_{n-1} + \alpha_2 s_{n-2} + \nu_n$ (where ν_n is white) is a second-order Markov signal.

- 5.6.7 Given the Markov signals of equation (5.24) and the previous exercise, can you recover the white noise signal ν_n ? What can you learn from the expression for ν_n ?
- 5.6.8 What is the power $\langle s_n^2 \rangle$ of the Markov signal of equation (5.24)? Why did we require $|\alpha| < 1$? The special case $\alpha = 1$ is called the *random walk* or *Wiener signal*. What happens here?
- 5.6.9 Pink noise is a term often used for a noise whose power spectrum decreases 3 dB per octave (doubling of frequency). What is the spectral density's dependence on frequency? How does the power per octave depend on frequency?
- 5.6.10 Blue noise is the opposite of pink, with power spectrum increasing 3 dB per octave; red noise has a 6 dB drop per octave. How do these spectral densities depend on frequency?

5.7 Spectrum of Random Signals

We know what the spectrum of a signal is, and thus we know what the spectrum of a single realization of a stochastic signal is; but can we give meaning to the spectrum of the entire stochastic signal? The importance of the frequency domain in signal processing requires us to find some consistent definition for the spectrum of noisy signals. Without such an interpretation the concept of filtering would break down, and the usefulness of DSP to real signals (all of which are noisy to some degree) would be cast in doubt. Fortunately, although a much more formidable task than it would seem, it *is* possible to define (and compute) the spectrum of a stochastic signal. Unfortunately, there are several different ways to do so.

If we consider the entire ensemble and take the FT of each realization individually we obtain an ensemble of transforms. Well, almost all realizations of a *stationary* stochastic signal will have infinite energy and therefore the FT won't converge, but we already know (see Section 4.6) to use the STFT for this case. Similarly, for nonstationary signals whose statistics vary slowly enough we can use the STFT over short enough times that the signal is approximately stationary. Thus from here on we shall concentrate on the STFT of stationary random signals.

We could consider the entire ensemble of spectra as *the spectrum*. Such a 'spectrum' is itself stochastic, that is, for every frequency we have a complex random variable representing the magnitude and angle. To see why these are truly random variables consider a realization of a white noise signal. Many

more realizations of the same stochastic signal can be created by shifting this one in time by any arbitrary interval. Thus the phases of the spectrum of such a signal should be uniformly distributed random variables. There is no way to resolve this problem other than to avoid it. Thus we concentrate on the short time power spectrum of stationary stochastic signals. Returning to our ensemble of transforms we square the values and discard the phases and obtain an ensemble of power spectra.

For well-behaved stationary stochastic signals (the type we are interested in) a unique (nonrandom) power spectrum can be defined. In practice we do not have access to the entire ensemble of signals but can observe one particular realization of the stationary signal for some amount of time. Assuming ergodicity, this can be just as good. Thus, if we compute the short time power spectrum of the realization we happen to have, we expect to obtain a good estimate of the aforementioned true power spectrum.

What do we mean by a 'good' estimate? An estimator is considered good if it is unbiased and has a small variance. For example, consider the mean value of a stationary signal s. Were we to have access to the entire ensemble we could take any single moment of time, and calculate the mean of the signal values in all realizations of the ensemble at that time. This calculation provides the true mean. Since the signal is assumed stationary, we could repeat this at any other time and would obtain precisely the same result. Alternately, assuming ergodicity, we could perform the average over time in a single realization. For a digital signal this entails adding all signal values from $n = -\infty$ to $n = \infty$, which would take quite a long time to carry out. Instead we could estimate the mean by

$$m = \frac{1}{N} \sum_{n=1}^{N} s_n$$

summing over N consecutive signal values. Such an estimator is unbiased; it will be too large just as many times as it will be too small. More precisely, if we carry out the estimation process many times, the mean of the results will be the true mean. Also this estimator has a variance that decreases with increasing N as $\frac{1}{N}$. That is, if we double the number of times we estimate the mean, the average variance will drop by half; the variance vanishes in the limit $N \to \infty$.

Returning to power spectra, we expect that our estimation of the power spectrum based on the STFT of a single realization to be unbiased and have variance that vanishes asymptotically. Unfortunately, neither of these expectations is warranted. If we calculate the power spectrum based on a single realization, the estimated power spectrum thus obtained will be biased and will have a standard deviation of about the same size as the value being estimated. Increasing the size of the window of the STFT does reduce the bias but doesn't reduce the variance at all!

It is informative to understand the reasons for these enigmas. The bias problem is the less severe of the two and the easier one to understand. Simply stated, the bias comes from comparing two different entities. When we use the STFT to estimate the energy at a given frequency, we are actually dividing the frequency axis into bins, each of width determined by the number of signal points in the transform. The STFT estimated spectrum averages together the true spectrum's values for all frequencies in the bin. Thus the STFT power spectrum's value at some frequency f should not be expected to precisely replicate the true spectrum's value there. However, as the number of points in the STFT becomes larger, the bins become smaller and the difference between the two decreases. Another way of looking at this is to think of the STFT as the FT of the original signal multiplied by the data window. This will of course equal the desired FT convolved with the FT of the window function. For any given window duration use of good window functions can help (see Section 13.4), but the fundamental uncertainty remains. As the duration of the window increases the FT of the window function approaches a delta function and the bias disappears.

The true problem is the variance of our estimator. The spectral variance, unlike the variance of the mean, does not decrease with increasing the number of data points used. At first this seems puzzling but the reason (as first realized by Tukey in the late 1940s) is quite simple. When we double the size of the STFT we automatically double the number of frequency bins. All the information in the new data goes toward providing more frequency resolution and not toward improving the accuracy of the existing estimates. In order to decrease the variance we must find a way to exploit more of the signal without increasing the frequency resolution. Two such methods come to mind.

Assume that we increase the number of input signal values by a factor of M. Bartlett proposed performing M separate power spectra and averaging the results rather than performing a single (M times larger) STFT. This averaging is similar to the mean estimator discussed above, and reduces the estimator's variance by a factor of M. Welch further improved this method by overlapping the data (with 50% overlap being about ideal). Of course performing multiple transforms rather than a single large transform is somewhat less efficient if the FFT is being used, but this is a small price to pay for the variance reduction. The second way to reduce the variance does perform a single STFT but then sums adjacent bins to reduce the resolution. This effectively smooths the estimated power spectrum resulting in a similar variance reduction. We will delve further into these techniques in Section 13.3.

Earlier we stated that a unique (nonrandom) power spectrum can be defined. This was first done by Wiener and Khintchine based on the following theorem.

Theorem: Wiener-Khintchine

The autocorrelation and the power spectral density are an FT pair.

In Chapter 9 we will prove this theorem for the deterministic case (after properly defining the autocorrelation for deterministic signals). Here we take this theorem as the definition for the stationary stochastic case. The basic idea behind the theorem is clear. If we are only interested in the square of the spectrum then we should only have to look at second-order entities in the time domain; and the autocorrelation is the most basic of these.

Basing ourselves on Wiener-Khintchine we can now compute power spectra of noisy signals in a new way, due to Blackman and Tukey. Rather than directly computing the signal's FT and squaring, we calculate the autocorrelation and then take the FT. All that we have seen above about bias and variance still holds, but averaging the computed spectra still helps. Since we can use the FFT here as well, the Blackman-Tukey technique is similar in computational complexity to the more direct Bartlett and Welch methods.

EXERCISES

- 5.7.1 Generate a finite-duration digital signal consisting of a small number of sinusoids and create K realizations by adding zero-mean Gaussian noise of variance σ^2 . Compute the power spectrum in the following three ways. Compute the FT of each of the realizations, average, and then square. Compute the FT, square, and then average. Compute the autocorrelation from the realizations and find the power spectrum from Wiener-Khintchine. Compare your results and explain.
- 5.7.2 Generate a single long (a power of two is best) realization of a signal as above. Compare power spectrum estimates using windows without overlap, overlapping windows, and smoothing of a single long FFT.

5.8 Stochastic Approximation Methods

Sometimes we are allowed access to the ensemble of signals, in which case rather different techniques can be employed. As a concrete example we will briefly consider the Robbins-Monro algorithm for finding a zero of a function corrupted by additive noise. The zero of a function f(t) is a z such that f(z) = 0. Finding the zero of a purely deterministic function is relatively straightforward. The standard way is to search for intervals $[t_1 \dots t_2]$ where the sign of f(t) changes, i.e., $f(t_1) < 0$ and $f(t_2) > 0$ or $f(t_1) > 0$ and $f(t_2) < 0$. Then we look at some t in the interval $t_1 < t < t_2$, and check if f(t) = 0 to within the desired accuracy. If not, we replace either t_1 or t_2 with t, depending on the sign of f(t). The various algorithms differ only in the method of choosing t.

In the Robbins-Monro scenario we can only observe the noisy signal $g(t) = f(t) + \nu(t)$, where the noise is assumed to be zero-mean $\langle \nu(t) \rangle = 0$ and of finite variance $\langle \nu^2(t) \rangle < \infty$. However, we are allowed to make as many measurements of g(t) as we desire, at any t we wish. One way to proceed would be to imitate the standard procedure, but averaging out the noise by sampling g(t) a sufficient number of times. However, the smaller the absolute value of g(t), the more susceptible is its sign to noise. This causes the number of samples required to diverge.

The Robbins-Munro algorithm recursively updates the present estimate z_k for the zero instead.

$$z_{k+1} = z - \frac{g(z_k)}{k}$$
(5.27)

It can be shown that this procedure both converges to the desired root in the mean square, i.e.,

$$\lim_{k\to\infty}\left\langle (z_k-z)^2\right\rangle = 0$$

and converges with probability 1, i.e.,

$$\operatorname{Prob}(\lim_{k\to\infty}z_k=z)=1$$

although the convergence may, in practice, be very slow.

EXERCISES

- 5.8.1 Is the division by k required for the deterministic case? Code the algorithm and check for a few polynomials and a sinusoid.
- 5.8.2 Add noise to the signals you used in the previous exercise and run the full algorithm. How does the error in the zero location depend on the noise level?

5.9 Probabilistic Algorithms

The Robbins-Monro algorithm is a way to *combat* noise, but we have mentioned that there are probabilistic algorithms that actually *exploit* noise. The usual definition of 'algorithm' is a precisely defined (i.e., deterministic) prescription of the solution of a problem; why would we want to make an algorithm probabilistic? The reason has to do with practicalities; sometimes the standard deterministic algorithm takes too long to compute its answer, while a probabilistic algorithm may be able to come up with an usable estimate much faster.

Numerical integration is a good example. The deterministic approach requires dividing the x axis into small intervals and summing the value of the function in these intervals. The function needs to be approximately constant over each interval so for rapidly varying functions many functional values must be evaluated and summed. Multidimensional integration is much more demanding; here all of the axes corresponding to independent variables must be divided into sufficiently small intervals, so that the computational complexity increases exponentially with the dimensionality.

As a concrete example consider finding the area of a circle, which can be expressed as a two-dimensional integral. The standard numeric approach requires dividing two-dimensional space into a large number of small squares, and the integration is carried out by counting the number of squares inside the circle. Of course there will always be the problem of those squares that straddle the circumference of the circle; only by using small enough squares can we ensure that these questionable cases do not overly effect the answer.

How can a probabilistic algorithm find the area? Circumscribe the circle by a square and choose at random any point inside this square. The probability that this point is inside the circle is exactly the ratio of the area of the circle to that of the square. So by generating a large number of random points (using any of the random number generators of section 5.4) and counting up how many fall inside the circle we can get an estimate of the area. Note that there isn't a well-defined end to this computation; each new random point simply improves the previous estimate. So there is a natural trade-off between accuracy and computational complexity.

This lucky integration technique is often called Monte-Carlo integration (for obvious reasons), and you can bet that it can be generalized to any integration problem in any number of dimensions.

EXERCISES

- 5.9.1 Compute π by Monte-Carlo determination of the area of the unit-radius circle. Monitor the error as a function of the number of points generated. How does the computation required to obtain a given accuracy compare with that of direct numerical integration?
- 5.9.2 Find the volume of a unit-radius sphere and the hypervolume of a unit-radius hypersphere in four dimensions. Make the same computational complexity comparisons as in the previous exercise.
- 5.9.3 In certain cases deterministic and probabilistic approaches to integration can be combined to obtain a faster and more accurate method. Explain the idea and apply to the previous exercise. (Hint: Inscribe the circle with a second square.)

Bibliographical Notes

Our treatment of noise has been very different, and a good deal less pedantic, than that found in engineering textbooks. For those who miss the formalistic treatment there are several good books on stochastic processes. The classic text is that of Papoulis [190]; only slightly less classic but much less friendly is van Trees [264]; but any text that has the words 'stochastic' and 'process' in its title will probably do. There are also texts with a major emphasis on stochastic processes that mix in a certain amount of straight signal processing, e.g., [250], and others with the opposite stress, such as [188].

Those interested in more information regarding radar systems can try anything by Skolnik [245, 243, 244] or the book by Levanon [145].

The generation of pseudorandom signals is discussed at length in the second volume of Knuth [136]. The transformation from uniform to Gaussian distributed random numbers (also found in Knuth) was discovered by Box and Muller [22]. The standard text on shift register sequences is by Golomb [81].

Deterministic chaos is quite a popular subject, with many books, each with its own approach. A suitable text for physicists is [234], while there are other books suitable for engineers or for mathematicians. The popular account by Gleick [75] is accessible and interesting.

Perhaps the earliest mathematical account of noise is [219, 220], which presented a complete theory including power spectra, statistical properties, and the effect of nonlinear systems on noise. Many would claim that the most important book on stochastic processes is that of Papoulis mentioned above [190].

An accessible source from which one can gain insight regarding the true meaning of noise is to be found in much of what is called modern music.