14 Measurement and Coding

As the number of electrons or photons used to represent a bit becomes small enough to be counted without taking one's shoes off, the means to measure them must become correspondingly sophisticated. Weak signals must be separated from strong backgrounds, using devices that may present a range of constraints on how they can and cannot be used. The only certainty is that mistakes will be made; to be useful, a system must be able to anticipate, detect, and correct its errors. And all this must of course be done at the lowest cost, highest speed, greatest density, ...

This chapter will study a collection of techniques for addressing these problems, starting with the low-level instrumentation that measures a signal, turning to the mid-level modulation used to detect it, and closing with the high-level coding that represents information in it. A striking example that both demonstrates and helped develop these ideas is communication with deep-space probes. As they've traveled further and further out into the solar system the rate at which they can send data back to the Earth has remained roughly constant, because the decreasing signal strength has been matched by increasing communications efficiency due to using bigger antennas, with more sensitive electronics, and better compression and error correction [Posner & Stevens, 1984].

These important topics might appear to be mundane matters of engineering detail, hardly worth considering in a book about physics. That's wrong at three levels. First, without these details all the clever physical insight in the world would not be able to influence anything, so they provide the context needed to understand how to develop mechanisms into working devices. Second, these details make or break practical systems, turning fundamental physical limits into engineering design constraints. And finally, there are in fact very deep connections between these ideas and the character of physical law. We'll see that as we come to understand both engineering and nature better and better, it makes less and less sense to distinguish between the physical laws governing a system and the information represented in it. Novel physical mechanisms such as quantum logic (Chapter 16) offer promising replacements and enhancements to the present practice described here.

14.1 INSTRUMENTATION

14.1.1 Amplifiers

Measuring a signal usually requires some combination of amplification and filtering. The workhorse for manipulating analog signals is the *operational amplifier* (*op-amp*), an (almost) ideal amplifier that is remarkably versatile. Op-amps are available with input noise floors down to $\mathrm{nV}/\sqrt{\mathrm{Hz}}$, and output power up to kilowatts, at costs ranging from pennies to hundreds of dollars.

The key insight that led to the development of op-amps is that, while it is difficult to build an amplifier with a specified gain, a differential amplifier that has an enormous gain can have its properties determined solely by a feedback network. Furthermore, since the input–output relationship is determined by passive components in the feedback network, such an amplifier can also be very linear even though its transistors or vacuum tubes are not [Black, 1934].

An op-amp has two inputs; the output is the difference between the signal at the positive side and the signal at the negative side, multiplied by a gain of $\sim 10^6$. The exact value of the gain is not a reliable parameter, but consider the circuits shown in Figure 14.1. The op-amp will drive the output so that its non-inverting input is at the same potential as the inverting input. In these cases the non-inverting connection is grounded, therefore the inverting lead acts as a *virtual ground*: it isn't actually connected to ground, but it behaves like one as long as the op-amp is able to drive its output so that the inverting input matches the grounded non-inverting input.

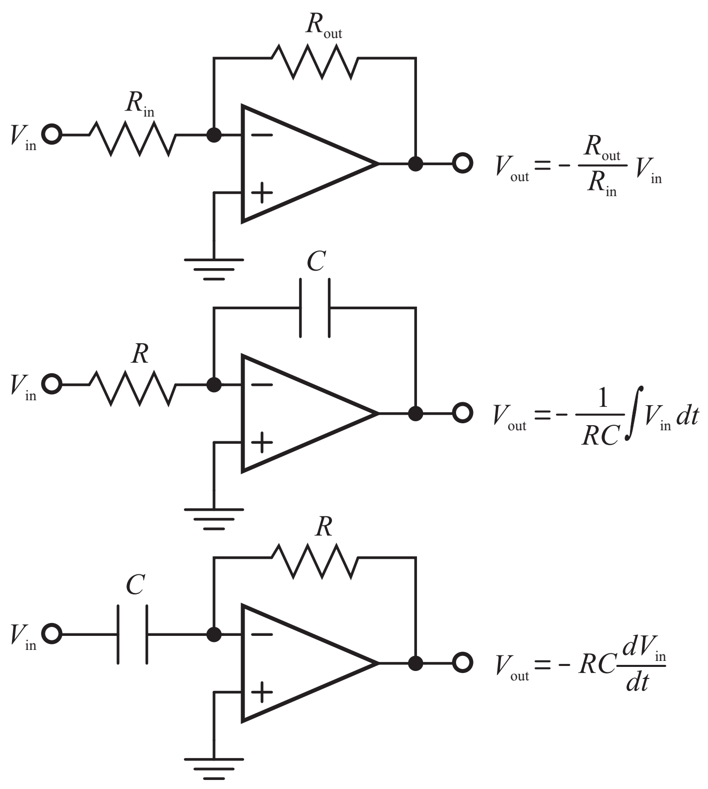
Most op-amps draw so little input current that it is a good approximation to assume that no current flows into the inputs. Requiring that the total current coming into and going out of the inverting node of the first circuit in Figure 14.1 adds up to zero gives the relationship

$$\frac{V_{\mathrm{in}} - 0}{R_{\mathrm{in}}} + \frac{V_{\mathrm{out}} - 0}{R_{\mathrm{out}}} = 0 \;\Rightarrow\; V_{\mathrm{out}} = -\frac{R_{\mathrm{out}}}{R_{\mathrm{in}}} V_{\mathrm{in}} \quad . \tag{14.1}$$

The output, which is inverted relative to the input, is given simply by the ratio of the two resistors. Related configurations accept current inputs or provide current outputs (Problem 14.1), and replacing one or the other of the resistors with a capacitor gives an integrator or a differentiator (Figure 14.1). Note that in a practical integration circuit a large resistor is usually added in parallel with the feedback capacitance, otherwise any small offset voltage error in the op-amp will be integrated up and eventually drive the output to the power supply rails (limits).

Op-amp integrators and differentiators can be used as low- or high-pass filters, and even to solve differential equations in an *analog computer* (although analog computers usually solve equations written just in terms of integrals because differentiation can increase the noise in the result). They were very important up to the 1950s for solving differential equations, and although they've been almost entirely replaced by digital computers they are still useful when fast, cheap, and continuous solutions are needed.

Balancing currents at the inverting nodes shows that the circuits in Figure 14.2 sum or difference their inputs. A differential amplifier is particularly useful for instrumentation because it can be used to measure a small difference in two signals that have a large common component, such as the same external interference. Because the performance



R_{out}
R_{in}
V_{in}
$V_{\text{out}} = -\frac{R_{\text{out}}}{R_{\text{in}}} V_{\text{in}}$
C
R
V_{in}
$V_{\text{out}} = -\frac{1}{RC}\int V_{\text{in}}\, dt$
R
C
V_{in}
$V_{\text{out}} = -RC\frac{dV_{\text{in}}}{dt}$

Figure 14.1. Op-amp amplifier, integrator, and differentiator.

is limited by how close the resistor values are, carefully matched pairs of resistors are available for differential amplifiers. Another limitation is the *Common Mode Rejection Ratio* (*CMRR*) of the op-amp, the ratio of the response to the difference in the input signals divided by the common value of the signals. This can easily be over 100 dB.

Common op-amps are internally *compensated* with a single-pole filter [Gershenfeld, 1999a] to ensure stability [Millman & Grabel, 1987]. Their gain as a function of frequency is

$$G(\omega) = \frac{G_{\text{ol}}}{1 + i\frac{\omega}{\omega_{\text{ol}}}} \quad , \tag{14.2}$$

where G_{ol} is the open-loop DC gain without an external feedback circuit, and ω_{ol} is where the open-loop filter rolls off. The frequency ω_1 where the gain is reduced to unity is easily found to be

$$\begin{aligned} 1 &= \left| \frac{G_{\text{ol}}}{1 + i\frac{\omega_1}{\omega_{\text{ol}}}} \right| \\ &= \frac{G_{\text{ol}}}{\sqrt{1 + \omega_1^2/\omega_{\text{ol}}^2}} \\ \omega_1^2 &= (G_{\text{ol}}^2 - 1)\omega_{\text{ol}}^2 \end{aligned}$$

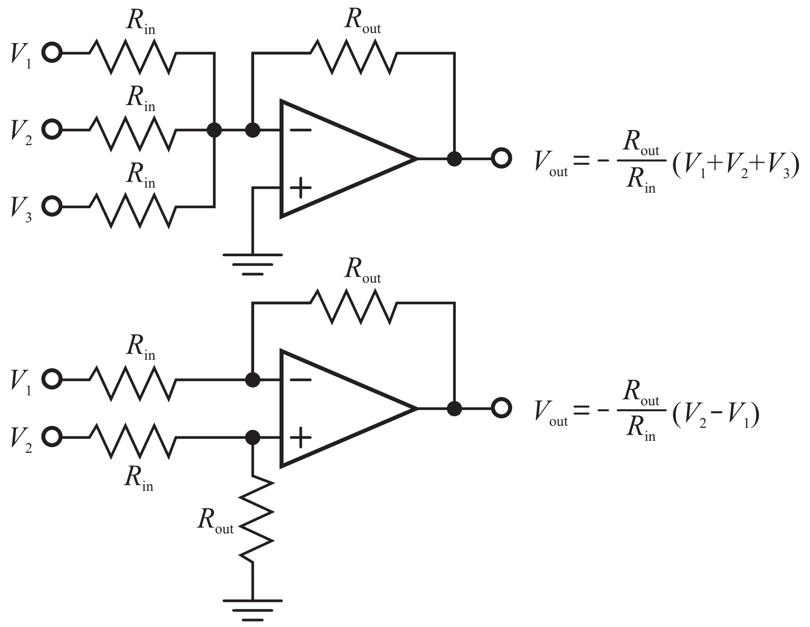

R_{in}
R_{out}
V_1
V_2
V_3
$V_{out} = -\frac{R_{out}}{R_{in}}(V_1+V_2+V_3)$
$V_{out} = -\frac{R_{out}}{R_{in}}(V_2-V_1)$

Figure 14.2. Summing and differential amplifiers.

$$\omega_1 \approx G_{ol}\omega_{ol} \tag{14.3}$$

(since $G_{ol} \gg 1$). This is why ω_1 is called the *gain–bandwidth product*. It determines the highest frequency that an op-amp can operate at; if the frequency response is reduced then higher gain is possible (Problem 14.2).

The *input impedance* and *output impedance* of an amplifier are other important specifications. These are the effective impedances seen by a device driving, or being driven by, the amplifier. The input impedance should be as large as possible so that the amplifier does not load its source; in an FET op-amp it can be $\sim 10^{12}\ \Omega$, while in a bipolar op-amp it can be as small as $\sim 10^9\ \Omega$ The output impedance should be as low as possible, otherwise the output voltage will depend on how much current is being drawn. Typical values range from ohms to kilo-ohms.

A differential amplifier has two practical constraints: its CMRR depends on how well the resistors are matched, and the input impedance is set by the input resistors. For very high output-impedance sources, the current drawn by these input resistors can be unacceptable. These problems can be fixed by using an *instrumentation amplifier*, shown in Figure 14.3. The inputs go directly into buffer amplifiers so that the input impedance is just the (large) amplifier input impedance. The outputs are connected in a clever divider circuit that amplifies the difference between the signals but not their common mode, and this goes to a unity gain differential amplifier that can have precision trimmed on-chip resistors. Balancing currents at the inverting pins gives

$$\frac{V_{out-} - V_-}{R_1} + \frac{V_+ - V_-}{R_2} = 0 \qquad \frac{V_{out+} - V_+}{R_1} + \frac{V_- - V_+}{R_2} = 0 \tag{14.4}$$

or

$$V_{\text{out}-} = \frac{R_1}{R_2}(V_- - V_+) + V_- \qquad V_{\text{out}+} = \frac{R_1}{R_2}(V_+ - V_-) + V_+ \quad . \tag{14.5}$$

The important change here is that the difference between the inputs is being amplified by R_1/R_2, while the individual signals which can contain common mode noise are passed through without gain. The output from the differential amp is then

$$V_{\text{out}} = V_{\text{out}+} - V_{\text{out}-} = \left(1 + \frac{2R_1}{R_2}\right)(V_+ - V_-) \quad . \tag{14.6}$$

The differential amp has a much easier job than before, removing the smaller common mode noise from the larger amplified difference signal from the first stage.

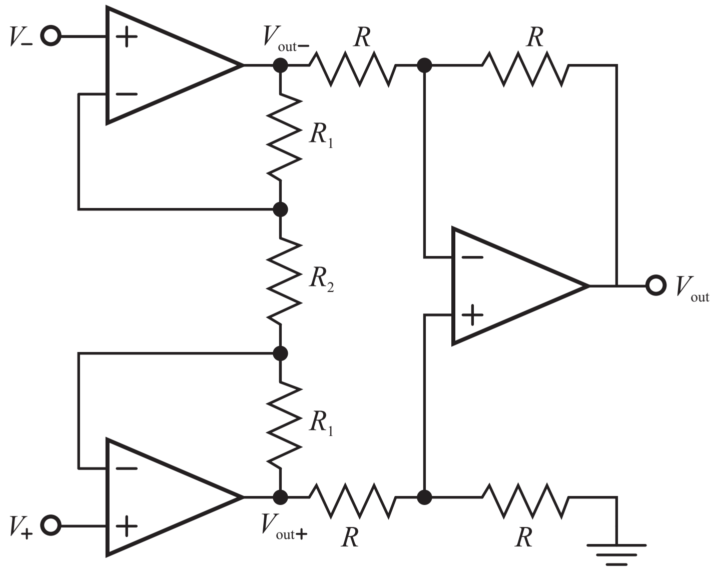


Figure 14.3. An instrumentation amplifier.

14.1.2 Grounding, Shielding, and Leads

An amplifier is only as good as its leads. While this reasonable observation has led to the unreasonable marketing of rather pathological cables to gullible audiophiles, it is true that small changes in wiring can have a very large impact on a system's performance (both good and bad). The goal is to make sure that as much of a signal of interest gets to its destination, and as little as possible of everything else. The polite term for this area is *electromagnetic compatibility*, asking, say, how to ground your battleship so that its electronics can withstand a nuclear electromagnetic pulse [Hunt & Fisher, 1990; Mitchell & George, 1998].

Although the principles for good wiring practice can appear to be closer to black magic than engineering design, they are really just an exercise in applying Maxwell's equations. Consider the series of circuits shown in Figure 14.4. In (*a*), a source is directly connected to a single-ended amplifier, introducing two serious problems. First, any other fluctuating

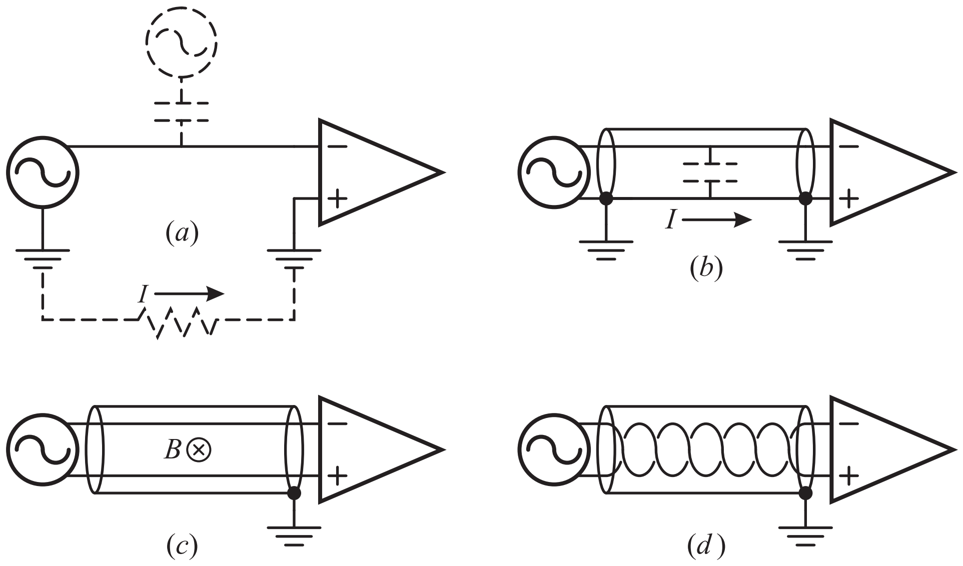


Figure 14.4. Grounding woes: (*a*) ground loop and capacitative pickup, (*b*) cross-talk and improper shield grounding, (*c*) magnetic pickup, and (*d*) shielded twisted pair.

voltages around the signal lead can capacitatively couple into it, producing interference. Second, the source and amplifier are grounded in different locations. Current must flow through the pathway connecting the grounds, and so any resistance there will lead to a change in the relative potentials. Even worse, this difference will depend on the load, and on everything else using the ground. This is called a *ground loop*; thick conducting braid is a favorite tool for combatting it by reducing the resistance between ground locations. Well-designed systems go further to maintain separate ground circuits for each function, with plenty of capacitance added to each as filters: one ground for digital logic with its high-frequency noise, one for motors with their large current surges, a quiet one for sensors requiring little current but good voltage stability, and so forth. These join only at a single *ground mecca* node.

Circuit (*b*) cures the capacitative pickup by surrounding the wire with a conducting shield, establishing an equipotential around it. Related tricks are building a conducting box around a sensitive circuit to provide electrostatic protection, and winding leads coming into and out of a circuit around a toroidal transformer core to provide inductive filtering of high-frequency noise. A cable shield comes at the cost of introducing a large capacitance from the source to the shield; for typical coaxial cable this can be tens of picofarads per foot, resulting in significant signal loss. That can be cured by using a unity-gain amplifier to drive the outer shield with the potential of the inner conductor. As long as the amplifier is fast enough, the shield will track the signal, effectively removing the cable capacitance. Special *followers* are available for this purpose, because if the amplifier is not fast enough, or cannot source enough current, then the dynamics of the cable-shield system can swamp the signal of interest. Circuit (*b*) also grounds the shield at both ends. This is effective if a heavy shield is used, so that the resistance of the connection is very small, but otherwise it brings the ground loop even closer to the signal lead.

In (*c*), both ends of the signal source are connected to a differential amplifier, and the cable shield is tied at one end. Not only is the shield not used as a continuous circuit, we don't want it to be available as one: its job is just to maintain the equipotential around the signal leads. And because the signals now arrive differentially, the amplifier can remove any common-mode interference that remains. That unfortunately does not help with another important noise source, time-varying magnetic flux linking the circuit, frequently coming from power lines. Even a high-permeability shield can't keep all the flux from threading between the conductors, and the induced potential appears as a voltage difference rather than a common mode shift.

The straight conductors are replaced in (*d*) with *shielded twisted pair*. The loops do two things: they reduce the cross-sectional area for flux pickup, and the direction of the induced current alternates between loops, approximately averaging it out. This is why shielded twisted pair, grounded at one end, is used most often for low-level signals. It ceases to be useful when the signal wavelength becomes on the order of the conductor spacing, but that cutoff can extend up to microwave frequencies.

If the measurement apparatus need only respond to a signal, a high input-impedance amplifier can be used that does not load the source. But if the apparatus is also responsible for providing current to excite the measurement, there can be a substantial voltage drop across the connecting leads that will vary as the load changes. This problem is cured by making a *four-terminal measurement*, shown in Figure 14.5. Each lead on the device under test, here taken to be a variable resistor, has two connections. One goes to a voltage or current source that drives the current through the leads and the device. And the others are used to measure the voltage drop across the device. The resistance in the current loop does not matter, because the current is the same everywhere. And the resistance in the voltage loop does not matter, because the voltmeter draws essentially no current. This is why precision reference resistors have four terminals, even though they appear in two apparently identical pairs.

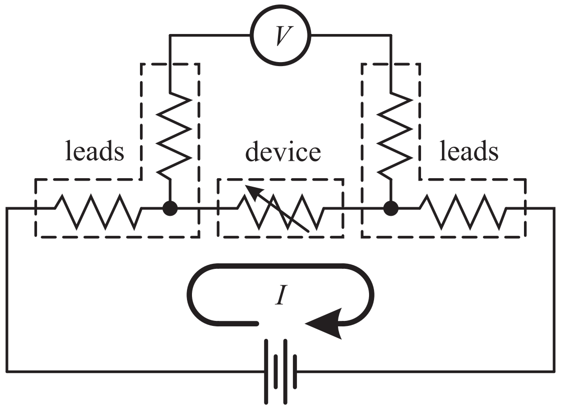


Figure 14.5. A four-terminal measurement.

When all these techniques fail, it's still possible to give up on electromagnetic shielding entirely and couple optically. For long runs, information can be sent in optical fibers, and many kinds of sensing are possible with all-optical devices (Chapter 15). Even within an electronic circuit, *optoisolators* pair an LED with a photodiode in a single package to provide a logical connection without an electrical one. These are used, for example, in the *Musical Instrument Digital Interface* (*MIDI*) specification to prevent ground loops in audio equipment [Lehrman & Tully, 1993], and in medical instruments to prevent ground loops in people.

14.1.3 Bridges

Many sensors, such as strain gauges and magnetoresistive heads, require detecting small impedance changes. While it's almost always preferable to measure small changes in small signals rather than find small changes in the value of a large signal, it's rarely possible to arbitrarily change the baseline impedance of these devices. *Bridge* circuits provide a solution to this problem.

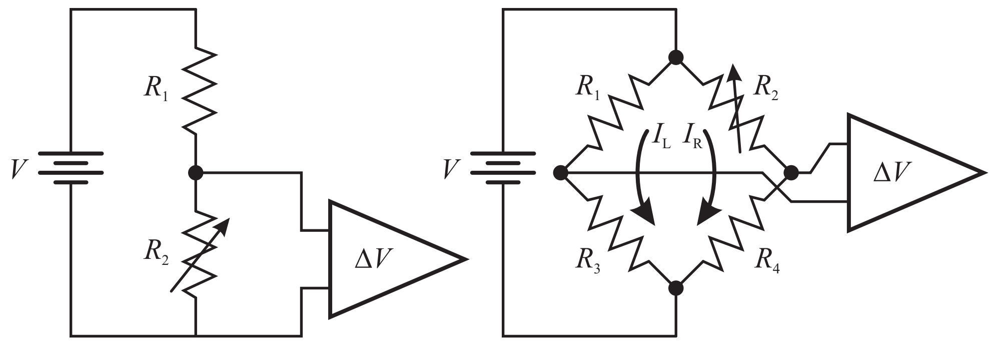


Figure 14.6. Measuring a small resistance change with a bridge.

If a voltage V is dropped over two resistors in series, a fixed one with resistance R_1 and a variable one R_2 (Figure 14.6), the measured voltage across the variable resistor will be

$$\Delta V = \frac{VR_2}{R_1 + R_2} \quad . \tag{14.7}$$

If the two resistors differ by the desired sensor signal, $R_1 = R$ and $R_2 = R + \delta$, then

$$\Delta V = \frac{VR}{2R + \delta} \approx \frac{V}{2}\left(1 - \frac{\delta}{2R}\right) \quad . \tag{14.8}$$

A small resistance change leads to a small change in a large voltage. If instead the resistances are arranged in a *Wheatstone bridge*, the voltage difference between the arms of the bridge is

$$\begin{aligned} V &= I_L(R_1 + R_3) = I_R(R_2 + R_4) \\ \Delta V &= I_R R_4 - I_L R_3 \\ &= V\left(\frac{R_4}{R_2 + R_4} - \frac{R_3}{R_1 + R_3}\right) \quad . \end{aligned} \tag{14.9}$$

Now if $R_1 = R_3 = R_4 = R$ and $R_2 = R + \delta$ then

$$\begin{aligned}\Delta V &= V\left(\frac{R}{2R+\delta} - \frac{R}{2R}\right) \\ &= V\left(\frac{1}{2+\delta/R} - \frac{1}{2}\right) \\ &\approx -\frac{V}{2}\frac{\delta}{2R} \quad .\end{aligned} \tag{14.10}$$

The small voltage change can now be measured directly without a large offset. This same analysis applies to complex impedances for variable capacitors and inductors.

14.2 MODULATION AND DETECTION

So far we've worried a great deal about the integrity of the signals we're trying to measure, but not at all about their design. Since this can usually be selected in an engineered system, the next group of techniques seek representations of information that help with *signal separation* (distinguishing between noise and the signal), and with satisfying constraints such as limited bandwidth.

14.2.1 Synchronous Detection

If the quantity of interest can periodically be modulated by the measurement apparatus, *synchronous detection* with a *lock-in* amplifier can find a weak signal buried in much larger noise (Figure 14.7). For a bridge circuit the modulation could be done by replacing the DC voltage source with an AC one; for an optical measurement the modulation might periodically vary the intensity of the light source. For this to work the noise must not depend on the excitation. A lock-in can reduce amplifier Johnson noise that is present independent of the input, but not photodetector shot noise that turns on and off with the light. Problem 14.3 looks at typical numbers for this kind of noise reduction.

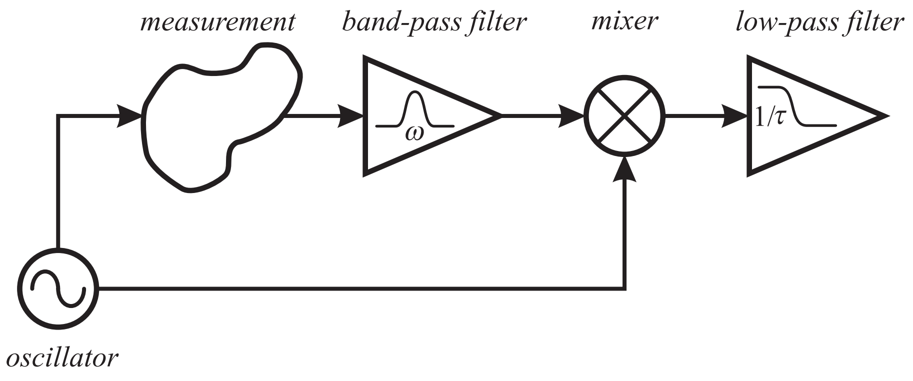


Figure 14.7. A lock-in amplifier.

In a lock-in an oscillator generates a periodic excitation $\sin(\omega t)$ that drives the measurement, resulting in a signal $A(t)\sin(\omega t) + \eta(t)$ that includes the desired response $A(t)$

along with unwanted noise $\eta(t)$. Since multiplication in the time domain is equal to convolution in the frequency domain, the detected signal is convolved around the oscillator (the positive-frequency components are shown in Figure 14.8). An immediate advantage of this is that the subsequent amplification can happen at the oscillator's frequency rather than near DC, away from the amplifier's $1/f$ noise. The front end also includes a band-pass filter centered on the oscillator that is broad enough to include the bandwidth of $A(t)$, but that rejects the remaining out-of-band noise in $\eta(t)$.

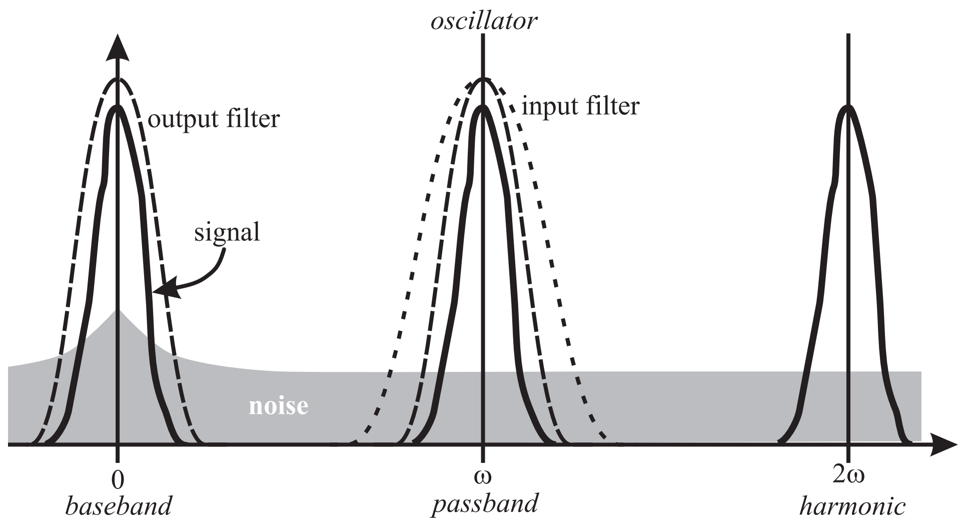


Figure 14.8. Lock-in amplification in the frequency domain (not to scale).

Next, the output from the filter is multiplied by the same oscillator signal to demodulate it, generating sum and difference terms:

$$\begin{aligned}[A(t)\sin(\omega t) + \eta(t)]\sin(\omega t) &= A(t)\sin^2(\omega t) + \eta(t)\sin(\omega t) \\ &= \frac{1}{2}A(t)\cos(\omega t - \omega t) - \frac{1}{2}A(t)\cos(\omega t + \omega t) \\ &\quad + \eta(t)\sin(\omega t) \\ &= \frac{1}{2}A(t) - \frac{1}{2}A(t)\cos(2\omega t) + \eta(t)\sin(\omega t) \quad . \end{aligned} \tag{14.11}$$

This is called *homodyne* detection; if a different signal source is used in the mixer it is *hetereodyne* detection. Heterodyne detection is used, for example, to down-convert a radio signal to an *IF* (*Intermediate Frequency*) stage for further amplification before final demodulation.

The final step in a lock-in is to pass the demodulated output through a low-pass filter to separate out the measurement component near DC from the modulated noise and sum signals, leaving just $A(t)/2$. In the time domain, the low-pass filter response is found by convolving the input with its impulse response, which performs a weighted average of the signal

$$\langle [A(t)\sin(\omega t) + \eta(t)]\sin(\omega t)\rangle = \langle A(t)\sin^2(\omega t)\rangle + \langle \eta(t)\sin(\omega t)\rangle$$

$$\approx \frac{1}{2}A(t) \quad . \tag{14.12}$$

This assumes that the noise is uncorrelated with the oscillator; the actual value of their overlap will depend on the duration over which the average is taken.

The lock-in has projected out the component of its input with the phase and frequency of the excitation. The noise rejection will depend on the output filter time constant, which can be quite long for a measurement near DC. It's instructive to view the output filter from before the mixer, where it appears to be a band-pass filter centered around the oscillator. But, unlike a conventional band-pass filter, we can make this one as narrow as we want by increasing the output filter time constant, and if the oscillator drifts the effective band-pass filter will automatically track it. In theory the input band-pass filter is not even needed at all, but in practice too much noise can lead to nonlinearities and cross-talk in the input stage that do get detected as a signal.

If there is any delay in the measurement there will be a phase shift that turns A into a complex quantity. To distintuish between changes in amplitude and phase, it's necessary to follow the input amplifier by two mixers and low-pass filters, one using $\sin(\omega t)$ to find the real component, and the other $\cos(\omega t)$ for the imaginary component. This is called *quadrature* detection, and the resulting values I and Q (for *In-phase* and *Quadrature*). The analog multiplication can be performed by a *Gilbert cell*, based on varying the current flowing through a differential amplifier [Gilbert, 1975]. The multiplier (or *mixer*) can also be replaced by a switch toggling between the signal and its inverse. This is the same as demodulating with a square wave; although there will be some noise pickup on the harmonics, it's much easier to make a nearly-ideal switch than a multiplier.

If the signal is digitized, the detection algorithm can instead be implemented in software in a digital signal processor. This provides algorithmic flexibility along with greater demands for power, complexity, and cost than a comparable analog circuit, although a periodic signal can be synchronously *undersampled* at less than its period as long as there is good phase stability [Smith, 1999].

14.2.2 Phase Detection and Encoding

If a lock-in measures components I and Q, the phase angle of the signal is given by $\tan^{-1}(I/Q)$. While the phase can be determined this way, a *Phase-Locked Loop* (*PLL*) is a handy cousin of the lock-in that eliminates the need for an inverse trigonometric function and can be used as a signal source as well as a signal analyzer [Wolaver, 1991]. An example is shown in Figure 14.9, with a mixer and *Voltage-Controlled Oscillator* (*VCO*) connected in a feedback loop around an active filter.

The multiplier is called a phase detector here. If $V_{\text{in}} = \cos(\omega t + \theta_{\text{in}})$, and $V_{\text{VCO}} = \sin(\omega t + \theta_{\text{VCO}})$, then if $\theta_{\text{in}} \approx \theta_{\text{VCO}}$ their product will be

$$\begin{aligned} 2\sin(\omega t + \theta_{\text{VCO}})\cos(\omega t + \theta_{\text{in}}) &= \sin(\theta_{\text{VCO}} - \theta_{\text{in}}) + \sin(2\omega t + \theta_{\text{VCO}} + \theta_{\text{in}}) \\ &\approx \theta_{\text{VCO}} - \theta_{\text{in}} \end{aligned} \tag{14.13}$$

(the sum signal is removed by the filter). The output is a DC value proportional to the phase difference

$$V_{\text{PD}} = K_{\text{PD}}(\theta_{\text{VCO}} - \theta_{\text{in}}) \tag{14.14}$$

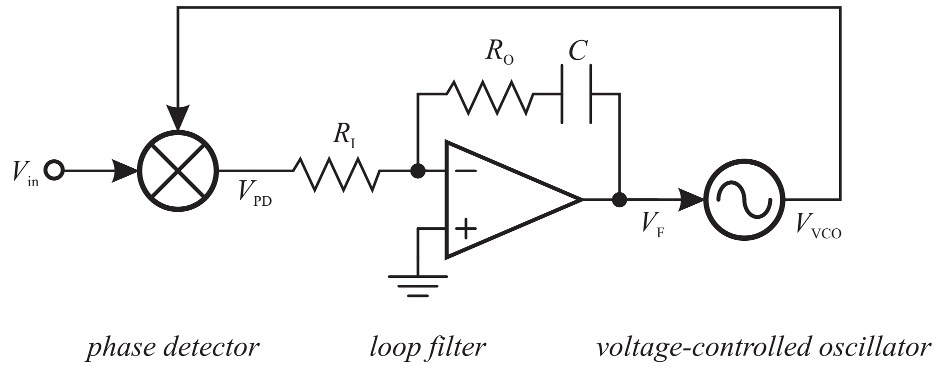


Figure 14.9. A Phase-Locked Loop.

with a coefficient K_{PD} that can include gain from the multiplier. As with a lock-in, this can also be implemented with a switch instead of a multiplier.

If the signal and VCO instead have a small frequency difference, then

$$\begin{aligned} 2\sin\left[(\omega_{in}+\delta\omega)t\right]\cos(\omega_{in}t) &= \sin(\delta\omega\ t)+\sin(2\omega_{in}t+\delta\omega\ t) \\ &\approx \delta\omega\ t \end{aligned} \tag{14.15}$$

the result is a slow ramp with a slope given by the frequency error.

Next comes the loop filter. Balancing currents into the non-inverting node (Problem 14.1),

$$\frac{dV_F}{dt} = -\frac{R_O}{R_I}\frac{dV_{PD}}{dt} - \frac{V_{PD}}{R_I C} \quad . \tag{14.16}$$

This is followed by the VCO, which has an instantaneous frequency $V_{VCO} = \cos(\omega_{VCO}t)$. Since we want to compare this to the input, their difference defines the time-dependent phase

$$\sin(\omega_{VCO}t) = \sin\left(\omega_{in}t + \theta_{VCO}(t)\right) \quad . \tag{14.17}$$

Since the frequency is the time derivative of the argument,

$$\omega_{VCO} = \frac{d\omega_{VCO}t}{dt} = \omega_{in} + \frac{d\theta_{VCO}}{dt} \quad . \tag{14.18}$$

The VCO puts out a frequency that is proportional to the input voltage, with a constant offset

$$\omega_{VCO} = K_{VCO}V_F + \omega_0 \quad . \tag{14.19}$$

Therefore

$$\frac{d\theta_{VCO}}{dt} = K_{VCO}V_F + \omega_0 - \omega_{in} \quad . \tag{14.20}$$

Now if the input frequency and phase are constant, the derivative of equation (14.14) will be

$$\frac{dV_{PD}}{dt} = K_{PD}\frac{d\theta_{VCO}}{dt} \quad , \tag{14.21}$$

so that

$$\frac{1}{K_{\rm PD}}\frac{dV_{\rm PD}}{dt} = K_{\rm VCO}V_{\rm F} + \omega_0 - \omega_{\rm in} \quad , \tag{14.22}$$

or taking the second derivative,

$$\frac{d^2V_{\rm PD}}{dt^2} = K_{\rm PD}K_{\rm VCO}\frac{dV_{\rm F}}{dt} \quad . \tag{14.23}$$

Plugging this into equation (14.16) gives

$$\frac{1}{K_{\rm PD}K_{\rm VCO}}\frac{d^2V_{\rm PD}}{dt^2} + \frac{R_2}{R_1}\frac{dV_{\rm PD}}{dt} + \frac{1}{R_1C}V_{\rm PD} = 0 \quad . \tag{14.24}$$

The phase detector output satisfies the equation of motion for a simple harmonic oscillator. The mass is set by the gain, the damping by the resistance ratio, and the restoring force depend on the feedback capacitance. These can be chosen to critically damp the PLL so that it locks onto the signal as quickly as possible. Because it can track changes in frequency as well as phase this is really a PFLL, but that doesn't have quite the same ring to it.

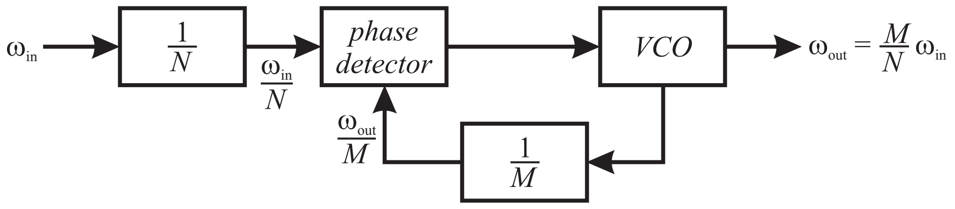

$\omega_{\rm in}$
$\frac{1}{N}$
$\frac{\omega_{\rm in}}{N}$
phase detector
VCO
$\omega_{\rm out} = \frac{M}{N}\omega_{\rm in}$
$\frac{\omega_{\rm out}}{M}$
$\frac{1}{M}$

Figure 14.10. A digital PLL frequency synthesizer.

One of the most important applications of a PLL is in generating and recovering timing clocks. Consider the digital variant shown in Figure 14.10. A reference oscillator at $\omega_{\rm in}$ goes to a counter which divides the frequency down by a divisor N. This alone could be used to synthesize other frequencies, but the resolution would be very poor for small values of N, requiring a very high input frequency. But here it is compared in the phase detector to the VCO output divided down in a second counter by a ratio M. The phase detector will move the VCO's frequency until these are equal, so that $\omega_{\rm in}/N = \omega_{\rm out}/M$, or $\omega_{\rm out} = M\omega_{\rm in}/N$. Now the frequency is determined by the ratio of M/N, giving much higher and more uniform frequency resolution for a given reference frequency.

On the recovery side, a PLL can lock the phase of a receiver onto a carrier sent by a remote transmitter. Once they share a phase reference, it's possible to use phase as well as frequency and amplitude to store information. Some possible modulation schemes are shown in Figure 14.11. The first, *On–Off Keying* (*OOK*), simply turns the carrier amplitude on and off. This is the digital version of *Amplitude Modulation* (*AM*). It works, but there's no way to distinguish between the off state and interference that blocks reception of the on state. Better is *Binary Phase-Shift Keying* (*BPSK*). Once the PLL is locked, it's possible to keep the carrier amplitude constant and switch just its sign. Now the logical states are independent of the signal strength; BPSK receivers can intentionally clip the input and use a digital PLL to eliminate the amplitude information. This provides much more reliable reception of weak and fluctuating signals. As with

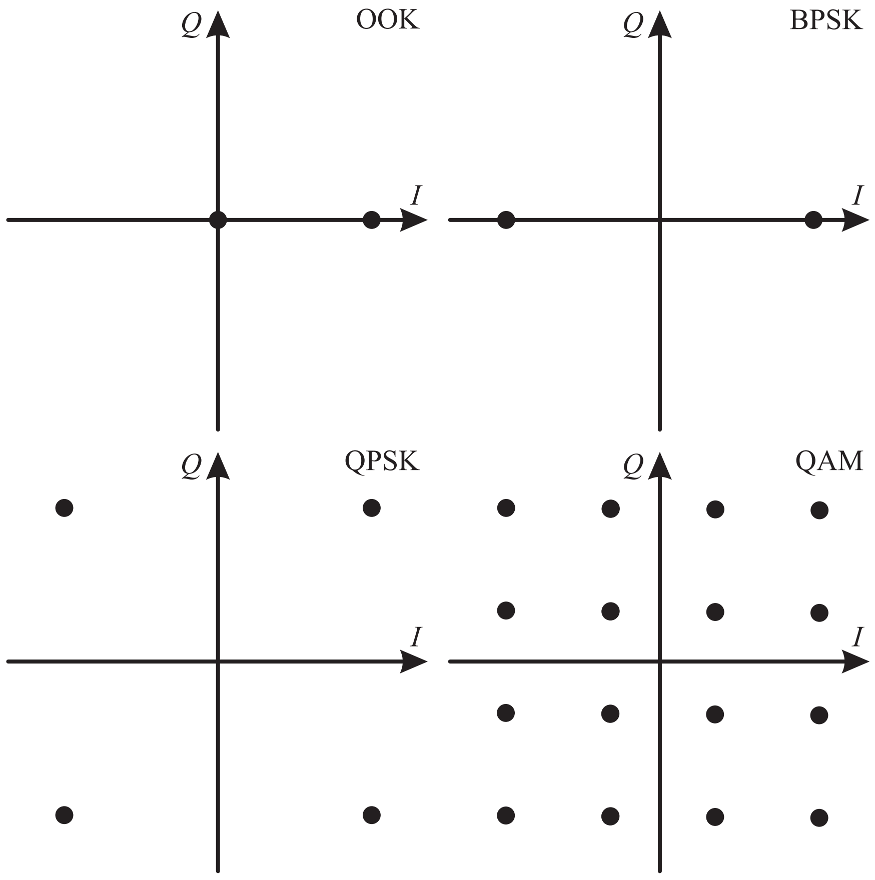


Figure 14.11. Modulation schemes.

a lock-in amplifier, it's possible to add a second demodulation channel with the VCO output phase-shifted by 90° to separately determine the I and Q components. Now *Quadrature Phase-Shift Keying* (*QPSK*) can be done, encoding information in four states based on the signs of the I and Q components. This send two bits instead of one per transmitted symbol (*baud*), but it's possible to do better still. The spacing of the states in the (I,Q) plane need only be as large as the expected channel noise. In *Quadrature Amplitude Modulation* (*QAM*) the amplitude information is used to squeeze in many more symbols; a V.32 modem sends 9600 bits per second in a 2400 Hz phone channel by using a constellation of 16 QAM states. By considering a string of symbols to be a vector in a higher-dimensional space it's possible to be even more efficient in arranging these. The underlying question of how to best pack spheres in a high-dimensional space is surprisingly deep [Conway & Sloane, 1993]. Finally, because the PLL can track changes in frequency as well as phase, information can also be sent that way in *Frequency-Shift Keying* (*FSK*). This can be less efficient in using available spectrum, and requires the transmitter and receiver to be designed to operate over a range of frequencies, but the analog version is familiar in *FM* (*Frequency Modulation*) radios.

14.2.3 Spread Spectrum

In a lock-in amplifier, or an AM radio link, the measurement or message is multiplied by a narrowband carrier signal. Before demodulation the signal retains its original bandwidth, now centered on the carrier (Figure 14.8). This means that the system is susceptible to interference at that frequency. Any background noise (or even intentional jamming) that is near to the carrier will be detected as a valid signal. Even worse, if many different messages are transmitted at adjacent frequencies in a given band, the edges of their distributions will overlap and lead to interference among them. These problems put a premium on reducing the bandwidth of the transmitted signal, using very stable oscillators and narrow filters. *Spread-spectrum* communications systems instead use as much bandwidth as possible for each signal. Although this might appear to be a perverse response, it leads to a number of significant benefits.

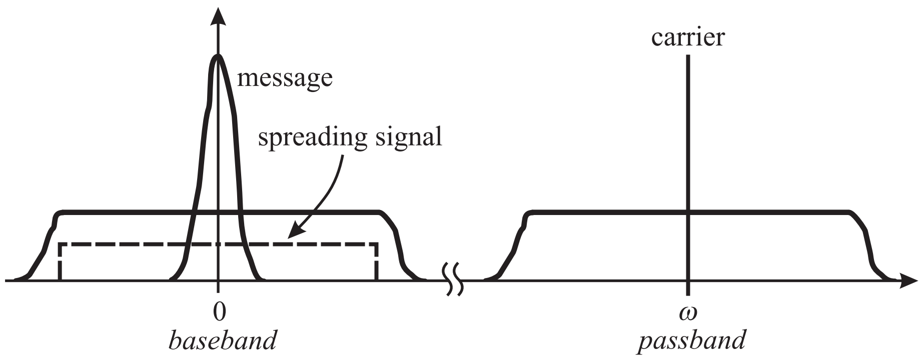


Figure 14.12. Spread-spectrum modulation.

A *direct-sequence* spread-spectrum system multiplies the original message by a broadband spreading signal that is generated by a *pseudo-random* noise source, one that appears to be random but is generated from a deterministic algorithm. This convolves the spectra so that the message fills the bandwidth of the spreading signal, which can then be modulated by another communications carrier to mix it up to the available transmission band (the positive-frequency component is shown in Figure 14.12). A variant is *frequency-hopping* spread spectrum, which uses the pseudo-random sequence to select the carrier frequency. The signal before modulation is said to be in the *baseband*, and after modulation it is in the *passband*. In the receiver, the high-frequency carrier is first removed by demodulation, then the message is retrieved from the spreaded signal by using a synchronized replica of the pseudo-random noise generator. We will henceforth ignore the (relatively) straightforward steps of modulation and demodulation by the carrier and consider just the spreading and de-spreading.

This is like a lock-in with a very noisy oscillator. The spreading process significantly increases the information content in the transmitted message and thereby reduces the overlap between the signal and interfering noise. For noise to be picked up in the receiver, it must now match the exact sequence of the pseudo-random generator rather than just a carrier's frequency and phase. Anything else will be rejected in the detector, up to a limit of how long we are willing to average the signal. This means that:

- Noise in the channel will be much less likely to be accepted as a valid signal.
- Different spreaded messages using the same bandwidth will interfere incoherently and so contribute only a small broadband component to the noise floor of the link. This kind of channel sharing, called *Code Division Multiple Access* (*CDMA*), degrades with load more gracefully than the alternatives of *TDMA* (*Time Division Multiple Access*, in which systems take turns using the channel), *FDMA* (*Frequency Division Multiple Access*, assigning them to different frequencies), and *CSMA* (*Carrier Sense Multiple Access*, where they take turns based on listening for channel activity).
- The chance of accidental or intentional reception by an unintended receiver is significantly reduced since without a synchronized replica of the spreading signal the message will appear to be random noise. Note that a determined eavesdropper can still decode this; for true security stronger cryptographic techniques must be used.
- A jamming signal must operate over a much larger bandwidth in order to be effective.
- Because there is energy everywhere in the spectrum, the peak power density is reduced, often an important regulatory consideration.
- The resolution in time that the arrival of a message can be measured to is approximately equal to the inverse of its bandwidth because of the *frequency–time uncertainty*, therefore increasing the bandwidth improves timing measurements.

For these reasons spread-spectrum links are more robust and make better use of the available communications bandwidth than the alternatives we've considered, and hence are increasingly common in new designs ranging from sensitive laboratory equipment to communications modems to the GPS satellite positioning system. Their noise rejection is measured by the *coding gain*, which is the ratio in decibels of the energy per bit required to obtain a given *Bit Error Rate* (*BER*) with and without coding, for a fixed noise power (Problem 14.4).

Any spread-spectrum system must provide the transmitter and receiver with identical synchronized copies of an ideal pseudo-random noise source. The earliest patent for an implementation was granted during World War II to the actress Hedy Lamarr (Hedy Markey) and the composer George Antheil (#2,292,387, *Secret Communication System*, 1942) based on storing the noise sequence in a piano-roll mechanism. For those applications without ready access to a piano, a *linear feedback shift register* (*LFSR*) can be used instead. An *order* N LFSR satisfies the recursion relation

$$x_n = \sum_{i=1}^{N} a_i x_{n-i} \quad (\text{mod } 2) \quad , \tag{14.25}$$

shown in Figure 14.13. x and a are binary variables, and the mod 2 operator gives the remainder after dividing by 2. The frequency at which the register gets updates is called the *chip rate*.

The last bit is fed back to the first after passing through as many adders as there are register stages, introducing a propagation delay that can be significant. This is corrected in the equivalent *Galois* or *modular* configuration, shown in Figure 14.14. This puts the

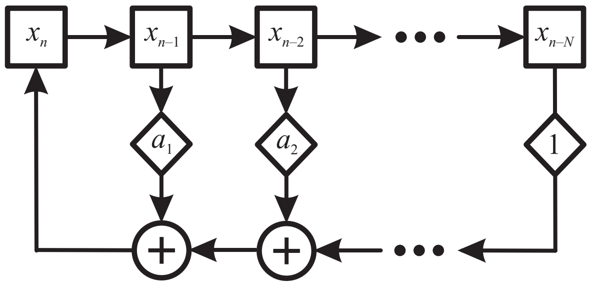


Figure 14.13. A linear feedback shift register.

adders between the register stages, so that the addition is accumulated into the bits as they advance through it.

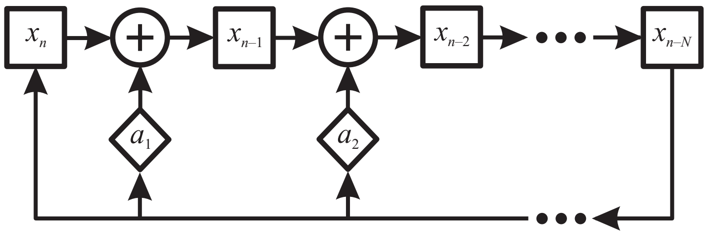


Figure 14.14. The Galois configuration for a shift register.

The sequence of bits generated by an LFSR will repeat after the register returns to the same state; the maximum possible sequence length is $2^N - 1$ (the all-0's state is not allowed because the system would get stuck there). Values for the taps a_i that give such *maximal* sequences can be found by asking that the z-transform of the recursion relation not have smaller polynomial factors, in much the same way as prime numbers are found [Gershenfeld, 1999a]. The resulting sequences satisfy many tests of randomness [Simon *et al.*, 1994], including a power spectrum that is flat up to the recurrence time, an autocorrelation function that equals 1 for a delay of 0 and the inverse of the sequence length otherwise, and the same number of 0's and 1's (plus 1, because the all-0's state has been omitted). Table 14.1 gives the coefficients for maximal LFSRs for a number of register lengths.

The hard part in any spread-spectrum implementation is synchronizing the transmitting and receiving shift registers. This comprises two parts: *acquisition* (setting the registers to the correct bit sequence) and then *tracking* (following drifts between the local clocks). The most straightforward, and widely-used, solution is the brute-force one of incrementally shifting the receiving register and cross-correlating with the incoming signal until an overlap is found. This slow process limits the speed of signal recovery.

An approximate alternative is to add the message into the transmitting LFSR

$$x_n = m_n + \sum_{i=1}^{M} a_i x_{n-i} \tag{14.26}$$

Table 14.1. *For a maximal LFSR* $x_n = \sum_{i=1}^{N} a_i x_{n-i}$ (mod 2)*, lag i values for which* $a_i = 1$ *for the given order N (all of the other* a_i*'s are 0).*

N	i	N	i	N	i
2	1, 2	13	1, 3, 4, 13	24	1, 2, 7, 24
3	1, 3	14	1, 6, 10, 14	25	3, 25
4	1, 4	15	1, 15	26	1, 2, 6, 26
5	2, 5	16	1, 3, 12, 16	27	1, 2, 5, 27
6	1, 6	17	3, 17	28	3, 28
7	3, 7	18	7, 18	29	2, 29
8	2, 3, 4, 8	19	1, 2, 5, 19	30	1, 2, 23, 30
9	4, 9	20	3, 20	31	3, 31
10	3, 10	21	2, 21	32	1, 2, 22, 32
11	2, 11	22	1, 22	33	13, 33
12	1, 4, 6, 12	23	5, 23	34	1, 2, 27, 34

and then add this to the output of a receiving register fed with the same sequence

$$\begin{aligned} r_n &= x_n + \sum_{i=1}^{M} a_i x_{n-i} \\ &= m_n + \sum_{i=1}^{M} a_i x_{n-i} + \sum_{i=1}^{M} a_i x_{n-i} \\ &= m_n \end{aligned} \tag{14.27}$$

(remember that $x + x = 0$ mod 2). This *self-synchronizing* configuration automatically recovers the message, but because the message enters into the register the noise is no longer guaranteed to be optimal, and the receiver is more susceptible to errors and artifacts.

An interesting alternative starts with the recognition that a PLL performs the desired acquisition and tracking for a periodic signal. This can be extended to pseudo-random sequences by replacing the LFSR with an *Analog Feedback Shift Register* (*AFSR*), which is a real-valued map

$$x_n = \frac{1}{2}\left[1 - \cos\left(\pi \sum_{i=1}^{N} a_i x_{n-i}\right)\right] \tag{14.28}$$

with the same taps a_i as the corresponding LFSR. These two functions agree for digital values, but the analog freedom of the AFSR lets it lock onto a pseudo-random sequence coupled into it [Gershenfeld & Grinstein, 1995, Vigoda *et al.*, 2006].

14.2.4 Digitization

After a signal is amplified, filtered, and demodulated, the final step is usually to digitize it in an *Analog-to-Digital Converter* (*ADC* or *A/D*). These usually start with a *sample-and-hold* circuit to store the voltage on a capacitor to keep it steady during the conversion, followed by one of a number of strategies for turning the analog voltage into a digital number. *Flash* A/Ds are the fastest of all, having as many analog comparators as possible output states (e.g., 2^8=256 comparators for an 8-bit A/D). The conversion occurs in a single step, but it is difficult to precisely trim that many comparators, and even harder to scale this approach up to many bits. A *successive approximation* A/D uses a tree of comparisons to simplify the circuit, at the expense of a slower conversion, by first checking to see if the voltage is above or below the middle of the range, then testing whether it is in the upper or lower quarter of that half, and so forth.

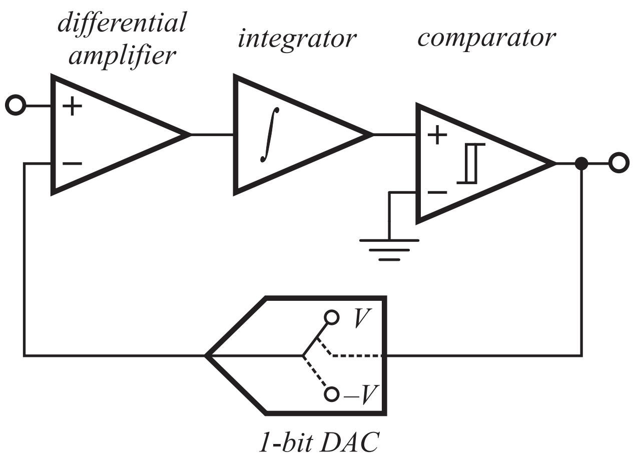


Figure 14.15. A delta-sigma ADC.

In a *dual-slope* A/D, the input voltage is used to charge a capacitor, then the time required to discharge it is measured. This eliminates the need for many precise comparators, and also can reject some noise becomes the result depends only on the average charging rate. The number of bits is fixed by the timing resolution. A *delta-sigma* A/D also converts the voltage into the time domain, but in a way that permits the resolution to be dynamically determinted as needed (14.15). The input goes first to a differential amplifier, then to an analog integrator. After that a comparator outputs a digital 1 or a 0 if the integrator output is positive or negative. It is configured as a *Schmitt trigger* (Figure 14.16), which has some hysteresis in its response so that it doesn't rattle around as the input crosses the threshold voltage. This controls a switch between the voltage rails that goes to the negative differential amplifier input.

Consider what happens if the input is grounded. The integrator will be charged up by the supply voltage until it hits the comparator's threshold to turn on, flipping the output bit. Then, the switch will move to the opposite rail, driving the integrator down until it hits the lower comparator threshold. The output bit will cycle between 1 and 0, with a frequency set by the integrator time constant. If the input is non-zero, there will be an asymmetry between the charging and discharging times, with the relative rates set by where the input is in the voltage range. The fraction of time the resulting bit string is a 1 or a 0, which can be determined by a digital filter, gives the input voltage. The beauty of this approach is that more resolution can be obtained simply by changing the coefficients of the digital filter to have a longer time constant.

Something similar is possible with any A/D as long as it is noisy enough. The digitization process introduces errors into the signal, which can be approximated by a Gaussian

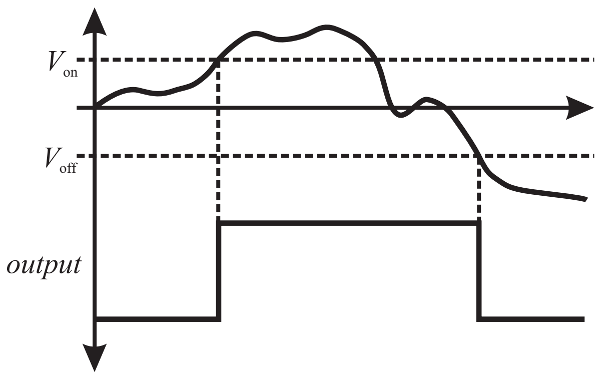


Figure 14.16. A Schmitt trigger.

noise source with a magnitude equal to the least significant bit. If in fact there is noise of that magnitude then repeated readings can be used to improve the estimate below the bit resolution. For this reason, high-performance converters intentionally add that much noise to the signal before digitization. This process is called *oversampling* because the conversions happen faster than what's required according to the Nyquist sampling theorem.

Similar devices operate in the opposite direction in a *Digital-to-Analog Converter* (*DAC* or *D/A*). A resistor ladder can be used to convert a set of bits to a voltage, but as with a flash A/D this requires precision trimming of the components and does not scale to high resolution. Here too a delta-sigma approach lets temporal resolution be used to obtain voltage resolution, using the same circuit as Figure 14.15 but now with digital logic. The difference is taken between the input and the output, summed into an accumulator, and used to trigger a comparator. This now controls a switch between the analog output rails, producing a waveform with the correct average voltage. An analog filter smooths this to produce the desired resolution; the frequency response is determined by the clock speed for updating the loop. Because that's much easier to increase than the precision of component values, delta-sigma converters dominate for high-performance applications such as digital audio.

14.3 CODING

Machines, like people, make mistakes, can talk too much, and have secrets. This final section takes a peek at some of the many techniques to reduce redundancy (*compression*), anticipate errors (*channel coding*) and fix them (*error correction*), and protect information (*cryptography*). These will all be phrased in terms of communicating digital messages through a channel, but the same ideas apply to anything that accepts inputs and provides outputs, such as a processor or a memory.

14.3.1 Compression

Our first step is *compression*. If there is redundancy in a message so that something is repeated over and over and over and over and over and over and over and over and over and over and over and over and over and over and over and over and over, it's more efficient to eliminate the redundancy by saying (and over)^{15}. This is a simple example of a *run-length code* that replaces repeating blocks with a description of their length.

Better still is to recognize that common messages should require fewer bits to send than uncommon ones. This is accomplished by *Huffman* coding. The idea is shown in Figure 14.17, which shows the relative probabilities of vowels in the King James Bible. If the letters are simply encoded as bits then some bit strings will occur more often than others because of the unequal letter probabilities. In an optimally encoded string, 1's and 0's and all possible combinations are equally likely. Huffman coding starts by grouping the symbols with the smallest probabilities to define a new effective symbol, and proceeding in this manner trying to balance the probabilities in the branches. Reading back from the right to decode a string, a variable number of bits is used depending on the frequency of the letter. A run-length Huffman code is used in the CCITT fax standard.

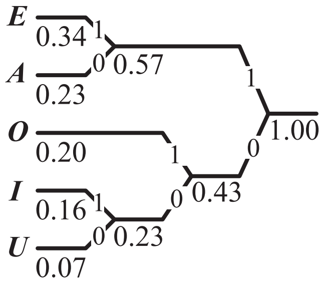


Figure 14.17. Huffman encoding of vowels.

The success of Huffman compression depends on how well the probabilities in the tree can be matched up. For asymptotically long strings it will attain the Shannon entropy bound, but for shorter strings it won't. A more recent approach, *arithmetic* compression, comes much closer to this limit (Figure 14.18). The unit interval is divided up into segments with lengths corresponding to the relative probabilities of the symbols. A single number will be used to encode a string. It is constrained to be in the interval associated with its first symbol. Then that interval is subdivided in the same fractions to find the subinterval fixed by the second symbol. That interval is divided again according to the third symbol, and so forth. Now the whole Bible could by written by one number. That alone is of course not the compression, because the number is a very long one. But the compression comes in when the intervals are written as fixed-precision binary fractions rather than infinite-precision real numbers. Then the average number of bits used per symbol will reflect their relative probabilities, up to that precision.

Arithmetic compression still requires advance knowledge of the source's probabilities. This may not be possible because of non-stationarity or unfamiliarity. *Universal* compressors attempt to attain the Shannon bound for arbitrary sources by adaptively building a descriptions of them. The *Lempel–Ziv–Welch* (*LZW*) scheme [Welch, 1984] is shown

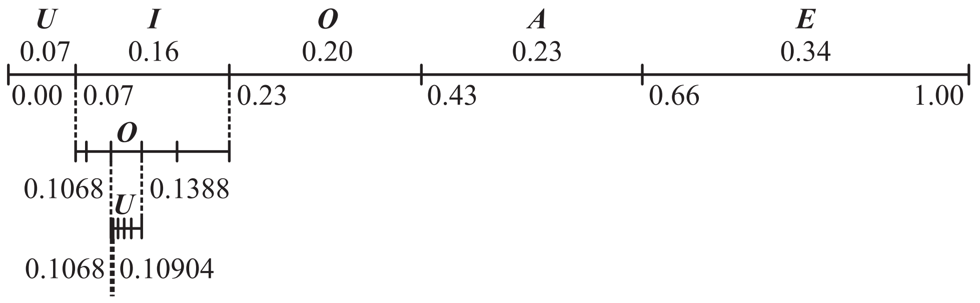


Figure 14.18. Arithmetic encoding of IOU.

in Figure 14.19; variants are used in most computer file compression utilities and modem compression standards. The encoder starts with a dictionary containing the possible symbols, in this case 0 and 1. It then works through the string, adding new entries to the dictionary as they are encountered, and transmitting the address of the known prefix to the decoder which can follow the reverse algorithm to reconstruct both the dictionary and the string. If there are N address bits used for dictionary addresses it's possible to store 2^N strings, which can be much longer than N. As the dictionary fills up, the encoder and decoder need to share an algorithm for pruning it, such as throwing out the least-used entry.

$\underline{1}\ \ \underline{0}\ \ \underline{1\ 0}\ \ \underline{1\ 0\ 1}\ \ \underline{0\ 1}$

prefix	*next character*	*transmitted entry*	*dictionary*
1	0	1	0 1 10
0	1	0	0 1 10 01
1	0	*	0 1 10 01
10	1	10	0 1 10 01 101
1	0	*	0 1 10 01 101
10	1	*	0 1 10 01 101
101	0	101	0 1 10 01 101 1010
0	1	*	0 1 10 01 101 1010
01	0	01	0 1 10 01 101 1010 010

Figure 14.19. Lempel-Ziv-Welch encoding of a periodic string.

So far we've been covering *lossless* compression which can be inverted to find the input string. *Lossy* compression cannot. While this might appear to be a dereliction of engineering duty, if the goal is to transmit a movie rather than a bank balance then all that matters is that it look the same. To see why this is needed, consider NTSC analog video, which provides roughly 640×480 bits of screen resolution at 30 frames per second [Pritchard & Gibson, 1980]. If we allow ourselves eight bits each for red, green, and blue color values, sending an NTSC channel digitally requires

$$640 \text{ pixels} \times 480 \text{ pixels} \times 24 \, \frac{\text{bits}}{\text{pixel}} \times 30 \, \frac{\text{frames}}{\text{s}} = 221 \times 10^6 \, \frac{\text{bits}}{\text{s}} \quad . \tag{14.29}$$

A fast network would be saturated by a standard that dates back to 1941. The *MPEG* (*Moving Picture Experts Group*) standards reduce by a few orders of magnitude the bit rate needed to deliver acceptable video [Sikora, 1997]. They accomplish this by taking advantage of a number of perceptual tricks, which is why lossy coding departs from rigorous engineering design and becomes an art that depends on insight into the application.

The details of the fine structure in an image are usually not important; the exact arrangement of the blades of grass in a field cannot be perceived. *Vector quantization* takes advantage of this insight to expand a signal in basis vectors and then approximate it by using nearby templates [Clarke, 1999]. And the ear will mask frequencies around a strong signal, so these can be discarded [Schroeder *et al.*, 1979]. MPEG compression also does *predictive* coding to send just updates to what a model forecasts the signal will do. This is only as effective as the model; the most sophisticated video coders build in enough physics to be able to describe the objects in a scene rather than the pixel values associated with a particular view of them [Bove, 1998].

14.3.2 Error Correction

Once a message is communicated as efficiently as possible, the next job is to make sure that it is sent as reliably as necessary. This is done by undoing some of the compression, carefully adding back enough redundancy so that errors can be detected and corrected.

The simplest error detection is to add up (mod 2) all of the bits in a data word to find the *parity* and append this value to the string. The receiver can then use the parity bit to catch any single bit error because it will change the parity. This prevents erroneous data from being used, but does not remove the error. If each bit is sent three times, then a majority vote can be taken, not only catching but correcting single-bit errors in the triple. This unfortunately also triples the data rate. Majority voting really overcorrects: it can repair as many errors as there are encoded bits, which may be far more than what's needed.

A *block code* corrects fewer bits with less overhead. In an (n, k) block code, k data symbols are sent in a block of n coded symbols, introducing $n - k$ extra ones for error correction. For a *Hamming* code of order m, $n = 2^m - 1$ and $k = 2^m - 1 - m$. The construction starts with the $(2^m - 1 - m) \times (2^m - 1 - m)$ *generator matrix* $\mathbf{G}$, which for $m = 3$ is

$$\mathbf{G} = [\mathbf{P}^T \ \mathbf{I}] = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad , \tag{14.30}$$

where $\mathbf{I}$ is the $m \times m$ identity matrix, and $\mathbf{P}$ has as its columns all possible m-element vectors with more than one non-zero element. A data vector $\vec{d}$ with $(2^m - 1 - m)$ components is associated with a $(2^m - 1)$-element codeword $\vec{c}$ by

$$\vec{c} = \mathbf{G}^T \, \vec{d} \quad . \tag{14.31}$$

This is received as $\vec{r} = \vec{c} + \vec{\eta}$, with possible errors $\vec{\eta}$. The received vector is then multiplied by the *parity check matrix*

$$\mathbf{H} = [\mathbf{I}\ \mathbf{P}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{14.32}$$

to find the *syndrome*

$$\begin{aligned} \vec{s} &= \mathbf{H}\,\vec{r} \\ &= \mathbf{H}\mathbf{G}^T\,\vec{d} + \mathbf{H}\,\vec{\eta} \\ &= [\mathbf{I}\ \mathbf{P}]\begin{bmatrix} \mathbf{P} \\ \mathbf{I} \end{bmatrix}\vec{d} + \mathbf{H}\,\vec{\eta} \\ &= \mathbf{P} + \mathbf{P} + \mathbf{H}\,\vec{\eta} \\ &= \mathbf{H}\,\vec{\eta} \quad . \end{aligned} \tag{14.33}$$

The last line follows because in binary arithmetic $1 + 1 = 0 + 0 = 0$. Since each column of the parity check matrix is unique, if there is a single bit error the offending element of $\vec{\eta}$ can be read off and corrected. Errors of more than one bit will also be recognized, but because the syndrome is no longer unique they can't be corrected. This procedure works because all of the codewords differ by at least three bits (their *Hamming distance* is 3 or more), so that a vector within one bit can uniquely be identified.

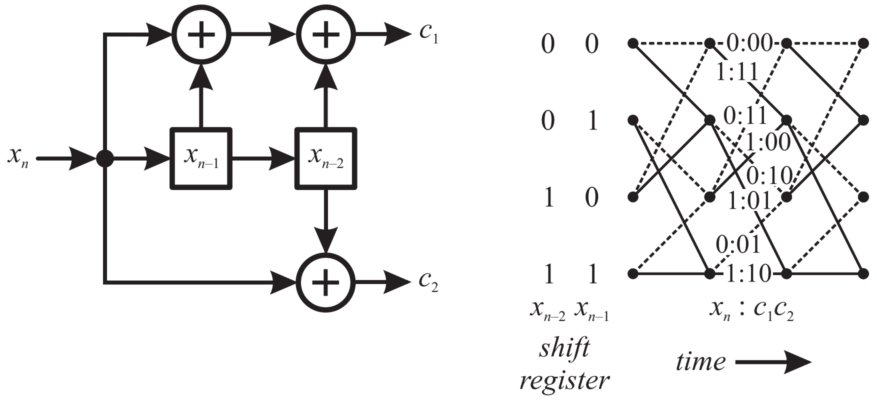


Figure 14.20. A convolutional coder, and the trellis that describes its output. The links are labelled by *data bit:code bits*, with dashed lines for links associated with 0's.

Errors don't have to stick to blocks, and neither do coders. In a *convolutional code* memory is introduced so that the decoding depends on the history of what's received, helping fix errors by taking advantage of information that is not adjacent in time. The idea is shown in Figure 14.20. Data bits enter into a shift register, which is tapped and summed to obtain the code bits. This example has a rate of 1/2, turning each data bit into two code bits. There is not a design theory analogous to that for maximal LFSRs to find optimal tap sequences, but good values have been found experimentally [Larsen, 1973].

The action of a convolutional encoder can best be understood through the *trellis* shown in Figure 14.20. There are four possible shift register states, each of which can be followed by an input 0 or 1. All of the possible transitions at each time step are shown, labelled by the code bits associated with them.

When the decoder receives a string of code bits it can determine what was transmitted by finding the path through the trellis with the smallest Hamming distance from what was received. Because of the correlation created by the encoding, this can depend on the full history of the signal. Decoding might appear to be a daunting task: if there are N time steps and M code words, there are N^M sequences to check. But as the trellis makes clear, when two sequences join at a node then it's only necessary to keep track of the most likely one. Given a received string, decoding can progress by a forward pass through the trellis, evaluating the smallest error path arriving at each node, and then a reverse pass reading back from the final node with the smallest final error. This is called the *Viterbi* algorithm [Viterbi & Omura, 1979]. It drops the computational cost from N^M to order NM, quite a savings! Given that difference, it's not surprising that this insight recurs in probabilistic estimation [Gershenfeld, 1999a], and in statistical mechanics, where it is possible to design spin systems that have as their ground state a decoded sequence [Sourlas, 1989]. Problem 14.6 works through an example of Viterbi decoding.

14.3.3 Channel Coding

After compression and error correction comes an essential final step: *channel coding*. This is where errors are prevented by modifying the message to satisfy a channel's constraints. For example, if too many identical digits are written in a row to a magnetic disk then the read head will saturate in that direction, if transitions happen too infrequently then the system clock will lose synchronization, if the average number of 1's does not match the average number of 0's there will be a net magnetization of the readout, and if bit reversals happen too quickly it will not be possible to follow them. This is a *Run-Length Limited* (*RLL*) system.

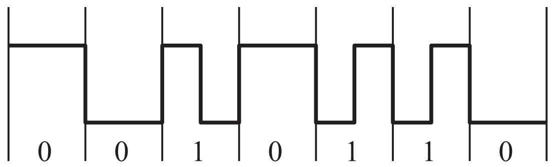


Figure 14.21. A Manchester code.

A simple solution is to use a *Manchester* code, shown in Figure 14.21. This always flips the output at the beginning of each interval, and then flips it again in the middle for a 1. The logical bits are now represented by the timing of the transitions in the channel bits, with at least one transition per bit guaranteed. This is easy to understand and implement, but has the very great disadvantage of doubling the rate of the code. It is still used in applications for which the bit rate can vary significantly, such as credit card readers.

Much more efficient use can be made of the medium by explicitly building in its limits. Disk drives for many years used an RLL(2,7) code, which restricts the minimum distance between bits to 2 and the maximum to 7 by encoding the data according to a variable-length block lookup table (Table 14.2). Because of the importance of maximizing storage density still more efficient codes have superseded this; the most general way to understand them is through the language of *symbolic dynamics*, devising dynamical systems that transform symbols subject to a set of constraints [Lind & Marcus, 1995].

Table 14.2. *Encoding table for an RLL(2,7) code.*

Data	Code word
00	1000
01	0100
100	001000
101	100100
111	000100
1100	00001000
1101	00100100

14.3.4 Cryptography

The preceeding techniques establish a reliable channel through imperfect devices, but this can be a bug rather than a feature if the receiver is not an intended one. Encoding information so that access can be controlled is the domain of *cryptography* [Simmons, 1992]. The essential insight behind modern cryptosystems is that there can be an asymmetry in the information required for encoding and decoding a message. In public key cryptography [Diffie & Hellman, 1976; Merkle, 1978] someone who wants to receive a secure transmission can openly publish a *public key* number that can be used by anyone to encrypt a message, but a secret *private key* is required to decrypt it. This relies on the existence of *one-way functions* which are easy to evaluate but very hard to invert.

The ubiquitous *RSA* scheme [Rivest *et al.*, 1978] relies on the difficulty of factoring. It starts by picking two large prime numbers p and q, with a product $n = pq$. Then two other integers e and d are selected for which

$$ed = 1 + (p-1)(q-1)r \tag{14.34}$$

for some integer r, i.e., d is the inverse of e mod $(p-1)(q-1)$. This combination is chosen because according to a version of *Fermat's Little Theorem* due to Euler [Koblitz, 1994]

$$m^{(p-1)(q-1)} = 1 \pmod{n} \tag{14.35}$$

for integer m that are not divisible by n. The number m can be formed from the bits of a message to be sent, and then encrypted with n and the public key e by

$$\mathcal{E}(m) = m^e \pmod{n} \quad . \tag{14.36}$$

This is easy to do, but hard to undo. But if the private key d is known, then it can be decrypted by another modular exponentiation

$$\mathcal{D}[\mathcal{E}(m)] = \mathcal{D}[m^e \pmod{n}]$$

$$\begin{aligned}
&= [m^e (\mathrm{mod}\ n)]^d\ (\mathrm{mod}\ n) \\
&= m^{ed}\ (\mathrm{mod}\ n) \\
&= m^{1+(p-1)(q-1)r}\ (\mathrm{mod}\ n) \\
&= m\left[m^{(p-1)(q-1)}\right]^r\ (\mathrm{mod}\ n) \\
&= m\,[1\ (\mathrm{mod}\ n)]^r\ (\mathrm{mod}\ n) \\
&= m\ (\mathrm{mod}\ n) \quad . \qquad (14.37)
\end{aligned}$$

Anyone with access to e can encrypt m, but only the holders of d can read it.

The security of this scheme rests on the presumed difficulty of finding prime factors, because if p and q could be found from n then equation (14.34) gives the secret key d. The best-known factoring algorithm is the *number field sieve* [Lenstra & Lenstra, Jr., 1993] which requires a number of steps on the order of $\mathcal{O}(e^{1.9(\log N)^{1/3}(\log\log(N))^{2/3}})$ to factor a number N. Because this is exponential in the number of digits in N, a linear increase in the key length imposes an exponential increase in the effort to find the factors. It is widely believed (but not proven) that it's not possible to factor in less than exponential time, unless you're fortunate enough to have a quantum computer (Chapter 16). We'll also see that quantum mechanics offers a way to distribute the private keys, which can't go through a public channel.

Classical cryptography can also be implemented physically, with one-way functions in coherent scattering [Pappu *et al.*, 2002]. This can reduce cost and simplify form factors as well as improve security; cryptography isn't useful if it can't be used because of constraints introduced by conventional electronics.

For some applications the presence of secret information itself must be kept secret, such as hidden IDs used to detect forgeries and copying. *Steganography*, the very old idea of hiding one kind of data in another kind of media, is becoming increasingly important as the range and value of new types of media and data proliferate [Johnson & Jajodia, 1998].

14.4 SELECTED REFERENCES

[Horowitz & Hill, 1993] Horowitz, Paul, & Hill, Winfield. (1993). *The Art of Electronics.* 2nd edn. New York: Cambridge University Press.

A great electronics handbook, full of practical experience that is equally useful for beginners and experts.

[Wolaver, 1991] Wolaver, Dan H. (1991). *Phase-Locked Loop Circuit Design.* Englewood Cliffs: Prentice Hall.

Fun with PLLs.

[Simon *et al.*, 1994] Simon, M.K., Omura, J.K., Scholtz, R.A., & Levitt, B.K. (1994). *Spread Spectrum Communications Handbook.* New York: McGraw-Hill.

[Dixon, 1984] Dixon, R.C. (1984). *Spread Spectrum Systems.* New York: John-Wiley & Sons.

Most everything there is to know about spread spectrum.

[Sklar, 1988] Sklar, Bernard. (1988). *Digital Communications: Fundamentals and Applications*. Englewood Cliffs: Prentice Hall.

[Blahut, 1990] Blahut, Richard E. (1990). *Digital Transmission of Information*. Reading: Addison-Wesley.

Good introductions to the theory of coding.

[Schroeder, 1990] Schroeder, M.R. (1990). *Number Theory in Science and Communication*. 2nd edn. New York: Springer-Verlag.

A lovely introduction to the deep mathematical framework supporting coding theory.

14.5 Problems

(14.1) (*a*) Show that the circuits in Figures 14.1 and 14.2 differentiate, integrate, sum, and difference.

(*b*) Design a non-inverting op-amp amplifier. Why are they used less commonly than inverting ones?

(*c*) Design a transimpedance (voltage out proportional to current in) and a transconductance (current out proportional to voltage in) op-amp circuit.

(*d*) Derive equation (14.16).

(14.2) If an op-amp with a gain–bandwidth product of 10 MHz and an open-loop DC gain of 100 dB is configured as an inverting amplifier, plot the magnitude and phase of the gain as a function of frequency as $R_{\text{out}}/R_{\text{in}}$ is varied.

(14.3) A lock-in has an oscillator frequency of 100 kHz, a bandpass filter Q of 50 (remember that the Q or *quality factor* is the ratio of the center frequency to the width between the frequencies at which the power is reduced by a factor of 2), an input detector that has a flat response up to 1 MHz, and an output filter time constant of 1 s. For simplicity, assume that both filters are flat in their passbands and have sharp cutoffs. Estimate the amount of noise reduction at each stage for a signal corrupted by additive uncorrelated white noise.

(14.4) (*a*) For an order 4 maximal LFSR work out the bit sequence.

(*b*) If an LFSR has a chip rate of 1 GHz, how long must it be for the time between repeats to be the age of the universe?

(*c*) Assuming a flat noise power spectrum, what is the coding gain if the entire sequence is used to send one bit?

(14.5) What is the SNR due to quantization noise in an 8-bit A/D? 16-bit? How much must the former be averaged to match the latter?

(14.6) The message 00 10 01 11 00 (c_1, c_2) was received from a noisy channel. If it was sent by the convolutional encoder in Figure 14.20, what data were transmitted?