

EXERCISES

- 18.11.1 In exercise 8.7.1 we introduced the games of doublets. What relationships exist between this game and the decoding of convolutional codes?
- 18.11.2 Each of the convolutions that make up a convolutional code can be identified by its impulse response, called the generator sequence in coding theory. What is the duration of the impulse response if the shift register is of length K ? The constraint length is defined to be the number of output bits that are influenced by an input bit. What is the constraint length if the shift register length is K , and each time instant k bits are shifted in and n are output?
- 18.11.3 Which seven-tuples of bits never appear as outputs of the simple convolutional code given in the text?
- 18.11.4 Repeat the last exercise of the previous section for convolutional codes. You need to replace the first program with `cencode` and the third with `cdecode`. Use the Viterbi algorithm.
- 18.11.5 The convolutional code $y_{2n} = x_n + x_{n-1}$, $y_{2n+1} = x_n + x_{n-2}$ is even simpler than the simple code we discussed in the text, but is not to be used. To find out why, draw the trellis diagram for this code. Show that this code suffers from *catastrophic error propagation*, that is, misinterpreted bits can lead to the decoder making an unlimited number of errors. (Hint: Assume that all zeros are transmitted and that the decoder enters state 3.)

18.12 PAM and FSK

Shannon's information capacity theorem is not constructive, that is, it tells us what information rate *can* be achieved, but does not actually supply us with a method for achieving this rate. In this section we will begin our quest for efficient transmission techniques, methods that will approach Shannon's limit on real channels.

Let's try to design a modem. We assume that the input is a stream of bits and the output a single analog signal that must pass through a noisy band-limited channel. Our first attempt will be very simplistic. We will send a signal value of $+1$ for every one in the input data stream, and a 0 for every zero bit. We previously called this signal NRZ, and we will see that it is the simplest type of **Pulse Amplitude Modulation (PAM)**. As we know, the bandwidth limitation will limit the rate that our signal can change, and so we will have a finite (in fact rather low) information transmission rate.

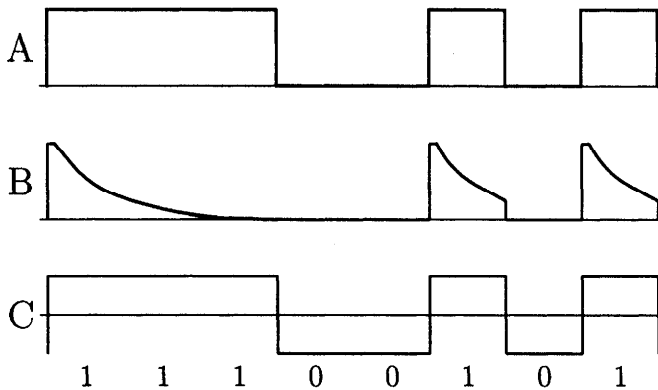


Figure 18.17: NRZ signals and DC. In (A) we see a straight NRZ signal. For each 1 bit there is a positive voltage, while there is no voltage for a 0 bit. In (B) we see what happens to this NRZ signal when DC is blocked. The positive voltages tend to decay, wiping out the information in long runs of 1 bits. In (C) we see an attempt at removing DC from the NRZ signal. For each 1 there is a positive voltage, while for each 0 bit there is a negative voltage.

NRZ has a major problem. Unless all the input bits happen to be zero, the NRZ signal will have a nonzero DC component. This is not desirable for an information transmission system. We want the transmitter to supply information to the receiver, not DC power! To ensure that DC power is not inadvertently transferred many channels block DC altogether. The telephone system supplies DC power for the telephone-set operation on the same pair of wires used by the signal, so low frequencies cannot be made available for the signal's use. This restriction is enforced by filtering out all frequencies less than 200 Hz from the audio. Attempts at sending our simple NRZ signal through a channel that blocks DC will result in the signal decaying exponentially with time, as in Figure 18.17.B. We see that single 1 bits can be correctly interpreted, but long runs of 1s disappear.

A simple correction that eliminates the major part of the DC is to send a signal value of $+\frac{1}{2}$ for every one bit, and $-\frac{1}{2}$ for every zero bit, as in Figure 18.17.C. However, there is no guarantee that the input bit stream will be precisely balanced, with the same number of ones and zeros. Even if this is true for long time averages, for short times there is still some DC (either positive or negative), and so the bits still decay.

Encoding methods discussed in Section 18.6 such as RZ or Manchester coding completely remove DC, but at the price of doubling the bandwidth; this is a price Shannon doesn't want us to pay. A better DC removing method

is **Alternate Mark Inversion (AMI)**, where a binary zero is encoded as 0 but a binary one is alternately encoded as $+\frac{1}{2}$ and $-\frac{1}{2}$. However, the decision levels for AMI are closer together than those of NRZ, resulting in decreased noise robustness, once again paying a capacity theorem price.

There are more sophisticated methods of eliminating the DC component. One way is to differentially encode the input bit stream before transmission. In differential coding a zero is encoded as no shift in output value and a one as a shift; and it is a simple matter to decode the differential coding at the demodulator. With differential coding a stretch of ones becomes 10101010 with many more alternations (although stretches of zeros remain). A more general mechanism is the 'bit scrambler' to be discussed in further detail in Section 18.15. The scrambler transforms constant stretches into alternating bits using a linear feedback shift register (LFSR), and the descrambler recovers the original bit stream from the scrambled bits. Although LFSR scramblers are often used they are not perfect. They are one-to-one transformations and so although common long constant stretches in the input are converted to outputs with lots of alternations, there must be inputs that cause the scrambler to output long streams of ones!

Even assuming the DC has been completely removed there are still problems with our simplest digital modem signal. One limitation is that it is only suitable for use in a baseband (DC to BW) channel (such as a pair of wires), and not in pass-band channels (like radio). The simplest 'fix' is to upmix NRZ to the desired frequency. The resultant signal will be either zero (corresponding to a 0 bit in the input) or a sinusoid of some single predetermined frequency f when we wish to send a 1 bit. This simple pass-band technique, depicted in Figure 18.18.A, is called **On Off Keying** or OOK.

Morse code was originally sent over cables using NRZ, using short and long periods of nonzero voltage. At the receiving end a relay would respond to this voltage, duplicating the motion of the sending key. This could be used to draw dashes and dots on paper, or the 'clicks' could be decoded by a human operator. In order to send Morse code over radio the OOK method was adopted. A carrier of constant frequency is sent and at the receiver IF mixed with a **Beat Frequency Oscillator (BFO)** to produce an audible difference frequency. Short intervals of tone are recognized by the receiving operator as 'dits' while tones of three times the duration are perceived as 'dahs'.

In theory we could use OOK to send more complex signals, such as Baudot encoded text, but this is rarely done. The reason is that OOK signal reception is extremely susceptible to noise and interference. To understand why think of how to decide whether a 1 or a 0 was sent. You can do no better

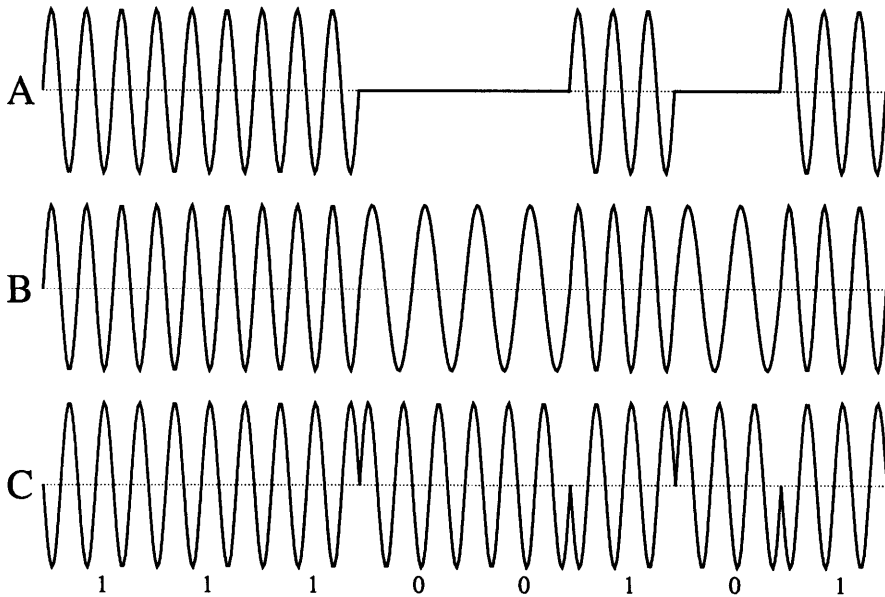


Figure 18.18: Simple digital communications signals. In (A) we see on-off keying (OOK), in (B) frequency shift keying (FSK), and in (C) phase shift keying (PSK).

than to simply look for energy at the carrier frequency, probably by using a band-pass filter of some sort, and judge whether it has passed a threshold value. Passing merits a 1 bit, lower energy is taken as a 0. If noise appears in the area of the frequency of interest, an intended 0 will be wrongly classified as a 1, and negative noise values summing with the signal of interest can cause a 1 to masquerade as a 0.

In order to ameliorate this problem noise should be explicitly taken into account, and this could be done in two nonexclusive ways. The first is based on the supposition that the noise is white and therefore its energy is the same over all frequencies. We can thus require that the energy at the output of a matched band-pass filter be much larger than the energy at nearby frequencies. The second way is based on the assumption that the noise is stationary and therefore its energy does not vary in time. We can thus monitor the energy when the signal is not active or sending zeros, and require that it pass a threshold set higher than this background energy. Such a threshold is often called a **Noise Riding Threshold (NRT)**.

While these mechanisms make our decision-making process more robust they are not immune to error. Impulse noise spikes still thwart NRTs and narrow-band noise overcomes a threshold based on nearby frequencies. In

addition, we have added the possibility of new error types (e.g., when the noise fluctuates at adjacent frequencies a perfectly good 1 can be discarded). While the effects of noise can never be entirely overcome, OOK does not give us very much to work with in our efforts to combat it.

Perhaps the simplest method to combat noise at adjacent frequencies is to replace OOK with **F**requency **S**hift **K**eying (FSK). Here we transmit a sinusoid of frequency f_0 when a 0 bit is intended and a sinusoid of frequency f_1 when we wish to send a 1 bit (see Figure 18.18.B). One can build an FSK demodulator by using two band-pass (matched) filters, one centered at f_0 and the other at f_1 . Such a demodulator can be more robust in noise since two energies are taken into account. One decision method would be to output a 0 when threshold is exceeded at f_0 but not f_1 and a 1 when the reverse occurs. When neither energy is significant we conclude that there is no signal, and if both thresholds are surpassed we conclude that there must be some noise or interference. When such a demodulator *does* output a 0 or 1 it is the result of two independent decisions, and we are thus twice as confident. An alternative to FSK is **P**hase **S**hift **K**eying (PSK), depicted in Figure 18.18.C. Here we employ a single frequency that can take on two different phases; a demodulator can operate by comparing the received signal with that of a sinusoid of constant phase. At first this seems no better than OOK, but we will see that PSK is a highly effective method.

There are several ways of understanding why FSK is better than OOK. Our first interpretation consisted of treating FSK as OOK with 'frequency diversity' (i.e., two independent OOK signals carrying the same information but at different frequencies). Such diversity increases the robustness with which we can retrieve information at a given SNR. This is as useful since we can increase channel capacity by attacking either the bandwidth or the noise constraints.

A second interpretation has to do with the orthogonality of sinusoids of different frequencies. An alternative to the dual band-pass filter FSK demodulator multiplies the received signal by sinusoids of frequencies f_0 or f_1 and integrates the output over time. Since this is essentially downmixing and low-pass filtering, this demodulator is actually a specific implementation of the dual band-pass filters, but we can give it a new interpretation. From equation (A.34) we know that sinusoids of different frequencies are orthogonal, so multiplication by one of the sinusoids and integrating leads to a positive indication if and only if this frequency is being transmitted. This exploitation of sinusoid orthogonality is a new feature relative to OOK.

Were the component signals in FSK truly orthogonal then FSK would be the answer to all our wishes. The problem is that sinusoids are only

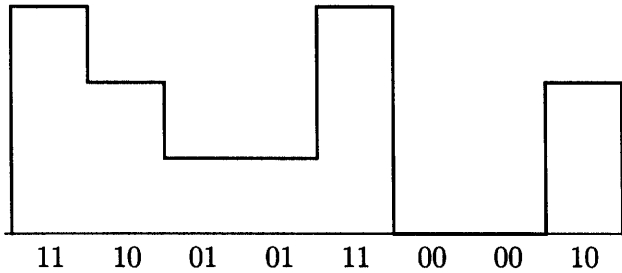


Figure 18.19: A four level PAM signal. Here the same information bits are transmitted as in the previous figures, but twice as many bits are sent in the same amount of time.

orthogonal when integrated over all time. When only a short time duration ΔT is available, the uncertainty theorem puts a constraint on the accuracy of recognizing the difference between the two frequencies Δf .

$$\Delta f \Delta t \geq 2\pi$$

For example, the telephone channel is less than 4 KHz wide, and so a rather large separation would be $\Delta f \approx 2$ KHz. This implies that telephone line FSK information transfer rates will not exceed around 300 b/s.

Of course the uncertainty theorem only directly limits the rate at which we can change between different frequencies. By using a repertoire of more than two frequencies we can increase the information transfer rate. Using four possible frequencies f_0 , f_1 , f_2 , or f_3 we simultaneously transmit two bits of information at each time instant, doubling the information rate. This technique is not limited to FSK; simple NRZ can be extended to multilevel PAM by sending one of four different voltages, as in Figure 18.19. The signal sent at each time period is usually called a *symbol* or *baud* (after Emile Baudot), and the bit rate is double the symbol rate or *baud rate*. If we use symbols that can take on 2^m possible values, the data rate in bits per second is m times the baud rate. Of course increasing the data rate in this way has its drawbacks. The demodulator becomes more complex, having to distinguish between many different levels. More significantly, if we compare two signals with the same transmission power, the one with more levels has these levels closer together. So we cannot increase the number of levels without incurring a higher probability of misdetection in noise. The eventual limit is when the level spacing is of the order of the noise intensity. This is the way Shannon limits the capacity of multilevel PAM and multifrequency FSK signals.

EXERCISES

- 18.12.1 A differential encoder encodes a 1 bit as a change in the signal and a 0 bit as no change. Show how to transmit differential NRZ, FSK and PSK. For which of these three is the differential coding most useful? How does one make a decoder that reconstructs the original bit stream from the differentially encoded one? How is the ambiguity broken? What signal causes an output of all 1s? All 0s?
- 18.12.2 If an FSK modulator employs two completely independent oscillators $s_1(t) = A \sin(2\pi f_1 t)$ and $s_2(t) = A \sin(2\pi f_2 t)$ then at the instant of switching the signal will generally be discontinuous. Continuous Phase FSK (CPFSK) changes frequency without phase jumps. Why is CPFSK better than non-continuous phase FSK? Write a routine that inputs a bit stream and outputs a CPFSK signal.
- 18.12.3 Program a multifrequency FSK modulator on a computer with an audio output or speaker. How high can the symbol rate be before your ear can no longer distinguish the individual tones?
- 18.12.4 The fundamental limitation on the FSK symbol rate is due to the time-frequency uncertainty relation. There is no fundamental time-value uncertainty relationship, so what is the source of the limitation on PAM symbol rates?
- 18.12.5 Figure 18.19 uses the natural encoding of the numbers from 0 to 2. The PAM signal called 2B1Q (used in ISDN and HDSL) uses the following mapping: $-3 \rightarrow 00$, $-1 \rightarrow 01$, $+1 \rightarrow 11$, $+3 \rightarrow 10$. What is the advantage of this 2B1Q encoding, which is a special case of a Gray code? How can this be extended to eight-level PAM? 2^m -level PAM?
- 18.12.6 In the text we didn't mention Amplitude Shift Keying (ASK). Draw a signal with four-level ASK. Why isn't this signal popular?

18.13 PSK

FSK demodulation is based on the orthogonality of the signals representing the bit 0 and the bit 1; unfortunately, we have seen that this orthogonality breaks down as we try to increase the information transfer rate. Over telephone lines FSK can be readily used for rates of 300–1200 b/s, but becomes increasingly problematic thereafter. In the previous section we mentioned PSK; here we will present it in more detail and show why it can carry information at higher rates. PSK is commonly used for 1200–2400 b/s over telephone lines.

Consider the two signals

$$\begin{aligned}s_0(t) &= \sin(2\pi f_c t) \\ s_1(t) &= \sin(2\pi f_c t + \frac{\pi}{2})\end{aligned}$$

where f_c is the carrier frequency. We suggest using these two signals as our information carriers, transmitting s_0 when a 0 bit is to sent, and s_1 for a 1 bit. We call this signal BPSK for **B**inary **P**SK. There are basically two methods of demodulating PSK signals. *Coherent demodulators* maintain a local oscillator of frequency f_c and compare the frequency of incoming signals to this clock. *Incoherent demodulators* do not maintain a precise internal clock, but look for jumps in the incoming signal's instantaneous phase.

Both BPSK demodulators are more complex than those we have seen so far. Are BPSK's advantages worth the extra complexity? Yes, since unlike FSK, where the two basic signals become orthogonal only after a relatively long time has elapsed, s_0 and s_1 are already orthogonal over a half cycle. So we can transmit one of the signals s_0 or s_1 for as little as one-half of a cycle of the carrier, and still discriminate which was transmitted. This is a major step forward.

Is this the best discrimination available? The coherent demodulator multiplies the incoming signal by $\sin(2\pi f_c t)$ and so after filtering out the component at twice f_c its output is either 0 or $\frac{\pi}{2}$. We can increase the phase difference by using the two signals

$$\begin{aligned}s_0(t) &= \sin(2\pi f_c t - \frac{\pi}{2}) \\ s_1(t) &= \sin(2\pi f_c t + \frac{\pi}{2})\end{aligned}$$

and the difference between the output signals is now maximal. It is not hard to see that $s_1 = -s_0$, so using a sinusoid and its inverse results in the best discrimination. Plotting multiple traces of the demodulator output results in an eye pattern, such as that of Figure 10.8. Using a phase difference of π opens the eye as much as is possible.

We can now go to a multiphase signal in order to get more bits per symbol. QPSK uses four different phases

$$\begin{aligned}s_0(t) &= \sin(2\pi f_c t) \\ s_1(t) &= \sin(2\pi f_c t + \frac{\pi}{2}) \\ s_2(t) &= \sin(2\pi f_c t + \pi) \\ s_3(t) &= \sin(2\pi f_c t + 3\frac{\pi}{2})\end{aligned}$$

or equivalently any rigid rotation of these phases. Unfortunately, multiplying by $\sin(2\pi f_c t)$ does not differentiate between the signals $s_a(t) = \sin(2\pi f_c t + \phi)$ and $s_b(t) = \sin(2\pi f_c t + (\pi - \phi))$, so our coherent demodulator seems to have broken down; but if we multiply by $\cos(2\pi f_c t)$ as well, we can discriminate between any two angles on the circle. This is not surprising since what we have done is to construct the instantaneous representation of Section 4.12. Calling the output of the sine mixer I and that of the cosine mixer Q , we can plot a two-dimensional representation of the analytic signal, called the 'I-Q plot'. The four points in the plane corresponding to the QPSK signal values are on the unit circle and correspond to four distinct angles separated by 90° . The two points of BPSK or four points of QPSK are called the *constellation*, this name originating from their appearance as points of light on an oscilloscope displaying the I-Q plot.

Let's generalize our discussion to nPSK. An nPSK signal is of the form

$$s(t) = e^{i(\omega_c t + \phi(t))} \quad (18.24)$$

with all the information being carried by the phase. The phase is held constant for the baud duration t_b , the reciprocal of which, f_b , is the baud rate. For nPSK this phase can take one of n different discrete values, and usually n is chosen to be a power of two $n = 2^m$. Hence the information rate in an nPSK signal is $m f_b$ bits per second.

What values should the phases take? We want the different values to be as far apart as possible, in order for the demodulator to be able to distinguish between them as easily as possible. One optimal way to choose the phases for nPSK is

$$\Phi_0 = 0, \quad \Phi_1 = \frac{2\pi}{n}, \quad \dots \quad \Phi_k = \frac{2\pi k}{n}, \quad \dots \quad \Phi_{n-1} = -\frac{2\pi}{n}$$

for example, the BPSK constellation should consist of two points with angles $0, \pi$, QPSK should have four points with angles multiples of 90° , and for 8PSK we choose eight points with angles multiples of 45° . These choices are good, but any rigid rotation of the entire constellation is equally acceptable. For example, BPSK can have phases $0, \pi$ as suggested here, or $\pm \frac{\pi}{2}$ or $\frac{\pi}{4}, \frac{5\pi}{4}$. Actually, there is no real difference between the different choices; from equation (18.24) it is obvious that an overall rotation of the phases is equivalent to resetting the clock (i.e., changing when $t = 0$ was).

When receiving an nPSK signal we first find its I and Q components, and from these calculate its instantaneous phase. Figure 18.20 is an I-Q plot of a received QPSK signal. Because of additive channel noise, channel distortion, and various inadequacies of the demodulation process, the actual symbols

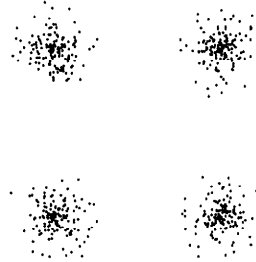


Figure 18.20: I-Q plot for a QPSK signal with noise. We see that the effect of additive channel noise is to replace the true constellation points with clouds centered at the original points. Although with this noise level we could still make accurate decisions, were we to add four more points to make 8PSK the clouds would touch and reception errors would be inevitable.

detected are not exactly those transmitted. We see that the four points on the unit circle have been transformed into four small ‘clouds’ centered on the original points. In order to recover the original information we have to decide to which true constellation point each received point should be associated. The decision is performed by a slicer and will be discussed in Section 18.18. How much noise can be tolerated before the decisions become faulty? From the figure it is obvious that if the radius of the noise cloud is less than half the Euclidean distance between the constellation points, then most of the decisions will be correct.

EXERCISES

- 18.13.1 What is the Euclidean distance between constellation points of an n PSK signal? Why can’t we increase the distance between the constellation points by simply placing them on a larger circle?
- 18.13.2 Simulate a baseband n PSK signal and find its empirical spectrum. Vary n . Do you see any change? Vary t_b . What happens now?
- 18.13.3 In exercise 18.12.5 we saw how to use a Gray code for multilevel PAM. What is the difference between a Gray code for PAM and one for n PSK? How is this related to the Hamiltonian cycles of Section 12.1?
- 18.13.4 We saw that resetting of the clock rotates the n PSK constellation. How can we ever be sure that we are properly interpreting the data?
- 18.13.5 Write programs that implement a QPSK modulator and demodulator. Try adding noise to the signal. What happens if the demodulator carrier frequency is slightly wrong? What if the baud rate is inaccurate?

18.14 Modem Spectra

If the data is ...10101010... then the NRZ signal is a square wave and its spectrum consists of discrete lines at odd harmonics of the baud rate. What is its spectrum when it carries random information? This spectrum is only defined in the sense of Section 5.7, and hence we should compute autocorrelations and use Wiener-Khintchine. Let's assume that the bits are white (i.e., that there is no correlation between consecutive bits). Then the autocorrelation of the NRZ signal will be zero for lags greater (in absolute value) than the baud rate. It requires only slightly more thought to convince oneself that the autocorrelation decreases linearly from its maximum, forming a triangle, as in Figure 18.21.A. The FT of this, and hence the desired PSD, is a sinc squared, depicted in Figure 18.21.B. The first zero is at f_b , consistent with uncertainty theorem constraints.

What is the PSD of a multilevel PAM signal? We could calculate the autocorrelation, but we can find the answer by a simpler argument. It is

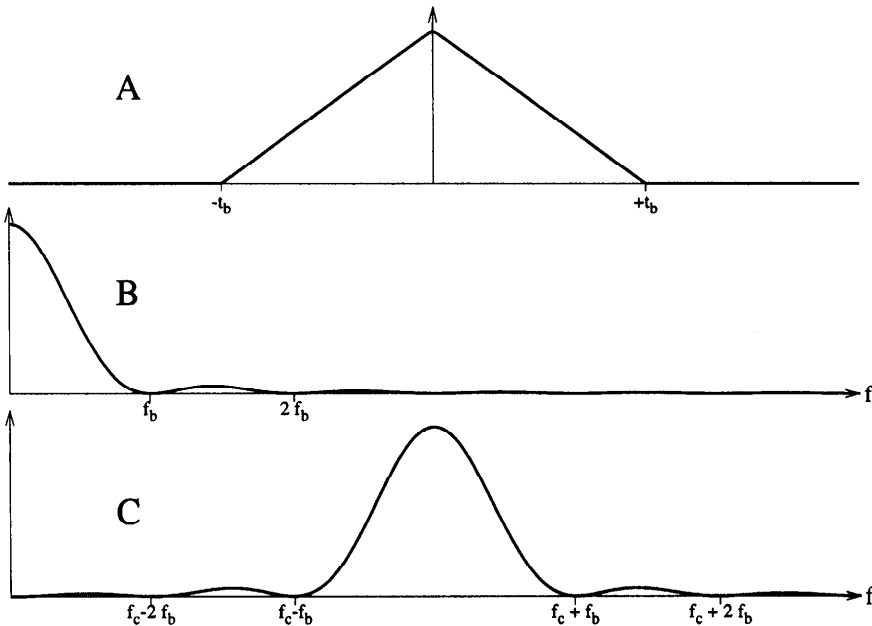


Figure 18.21: PSD of digital communications signals. In (A) we see the autocorrelation of a NRZ signal carrying white random data. In (B) is depicted the PSD of this signal, a sinc squared with its first null at the baud frequency. In (C) we present the PSD of a OOK signal with the same data.

obvious that a PAM signal with n levels can be thought of as the sum of n NRZ signals. From the linearity of the FT we can conclude that multilevel PAM has exactly the same spectrum as NRZ. In particular, the bandwidth of PAM is independent of the number of levels, but of course more levels with the same energy effects the noise sensitivity. Of course we have been comparing signals with the same baud rate; when we compare PAM signals with the same bit rate, the signal with more bits per symbol has a lower baud rate and hence a lower bandwidth.

What about an OOK signal? We don't have to recompute autocorrelations since we know that OOK is simply NRZ upmixed by the carrier. Accordingly, we immediately conclude that its PSD is that depicted in Figure 18.21.C, centered on the carrier frequency and taking up double the bandwidth of the NRZ signal. This spectrum is shared by the multilevel ASK signal as well. In fact, it is a quite general result that only the carrier frequency and baud rate affect the spectrum.

Why should the bandwidth have doubled for the same baud rate? This result hints that there is another degree of freedom that we are not exploiting, but that would not change the PSD. This degree of freedom is the phase; by simultaneously modulating both the amplitude and the phase we can double the bit rate without increasing the bandwidth. We will return to this idea in Section 18.17.

EXERCISES

- 18.14.1 In our derivation of the PSD for the NRZ signal we didn't dwell on the DC. What is the difference between the spectra of the NRZ and DC-removed NRZ signals?
- 18.14.2 Derive the PSD of BPSK from that of DC-removed NRZ. Compare this spectrum with that of OOK.
- 18.14.3 Summing n independent NRZ signals does not result in a nPAM signal with equally probable levels. Why does this not affect the conclusion regarding the PAM spectrum?
- 18.14.4 Compare the PSDs for BPSK, QPSK, and 8PSK with the same information transfer rate.
- 18.14.5 Create random data NRZ and BPSK signals and compute their periodogram using the FFT. Now use 01010101 input ('alternations'). How is the spectrum qualitatively different? Starting from the deterministic input of alternations add progressively more randomness. How does the spectrum change?

18.15 Timing Recovery

Anyone who has learned Morse code can tell you that sending it is much simpler than ‘reading’ it. This is quite a general phenomenon—digital demodulators are always more complex than the corresponding modulators. One reason is the need to combat noise added by the communications channel, but there are many others. Some of the most problematic involve synchronizing the modulator and demodulator time sources.

We usually differentiate between two time sources. *Baud rate recovery* refers to synchronization of the demodulator’s baud clock with that of the modulator. Every digital communications system must perform some sort of timing recovery. *Carrier recovery* refers to recovery of the carrier frequency for those modulation types that use a carrier. Obviously NRZ and PAM do not need carrier recovery. Rotating the I-Q plot to correspond to the proper constellation can be considered to be carrier phase recovery.

Consider, for example, a modulator with baud rate f_b , nominally known to the intended demodulator. It sends a new symbol every $t_b = \frac{1}{f_b}$ seconds (i.e., the first symbol occupies time from $t = 0$ to $t = t_b$, the second from $t = t_b$ to $t = 2t_b$, and so on). Although the demodulator expects this baud rate, its clock may differ slightly from that of the modulator, both in phase (i.e., precisely when $t = 0$ occurs) and in frequency (i.e., its t_b may be slightly shorter or longer than the intended t_b). Left uncorrected such slight frequency differences add up, and soon valid symbols will be missed or counted twice and the demodulator will attempt to decide on the symbol value based on observing the signal during a transition, as can be seen in Figure 18.22. Similarly, proper reception of a pass-band signal may require the demodulator to agree with the modulator as to the precise frequency and phase of the carrier.

The simplest method for the demodulator to obtain the modulator’s

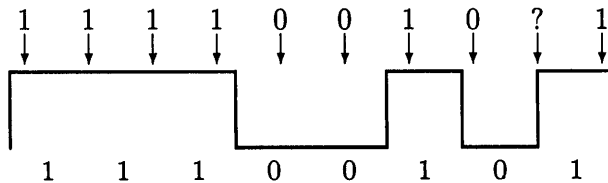


Figure 18.22: The effect of improper baud rate recovery. In this example the demodulator clock runs slightly faster than the modulator’s, so that bit insertions result. On occasion a symbol clock may fall directly on a transition, causing indefinite decisions.

clock is for it to be somehow delivered. For example, if the data-bearing signal is delivered on a pair of wires, then a second pair could be provided whose sole purpose is to carry the clock information. This information could be represented by pulses at precisely the moments transitions could take place, or by a sine wave with positive slope zero crossing at the moment of possible transition, or by any other previously agreed-upon method. The problem with this method is the need for an expensive second pair of wires, which could be more effectively used for carrying information. A slightly more efficient method would be to multiplex the clock signal onto the same pair of wires that carry the signal. For example, possible transition moments could be marked by pulses at some frequency not used by the data signal. This guarantees that the clock is delivered with the same delay as the signal, but of course wastes valuable bandwidth on the clock signal. These simple methods are attractive for systems that carry a large number of signals sharing the same clock, so that the overhead is small. In order not to waste wires or bandwidth the demodulator is often required to derive the clock from the information-bearing signal itself.

To demonstrate how baud rate recovery may be accomplished we'll discuss a simple NRZ signal, although the basic ideas remain intact for other signals after appropriate preprocessing. We will take the two levels of the NRZ signal to be ± 1 so that transitions are zero crossings. When the SNR is good these zero crossings are easily observable, and by detecting them and measuring the time between them the baud rate can be recovered. The time between two successive transitions must be an integral multiple of t_b , and this multiple is readily determined when the approximate t_b is known. Using observed transitions in this fashion, both the frequency and absolute phase of the modulator's clock can be recovered.

The following simplistic algorithm for NRZ demodulation can be run in real-time. After running for some time T is the current estimate of t_b and p is the time of the previous zero crossing.

```

input the next signal value  $s_n$ 
compute the time since the last transition  $d \leftarrow n - p$ 
if  $d$  is 'approximately half integer'
    output  $\text{sgn}(s_n)$ 
if  $s_n \cdot s_{n-1} < 0$ 
    interpolate  $r \leftarrow n - \frac{s_n}{s_{n-1} - s_n}$ 
    time between transitions  $\tau = r - p$ 
    compute the multiple  $m = \text{round} \frac{\tau}{T}$ 
    update estimate  $T \leftarrow \alpha T + (1 - \alpha) \frac{\tau}{m}$ 

```

One could significantly improve this algorithm with little effort. It's better to base the output decision on several successive s_n , so some filtering or median filtering should be performed. If the signal has some residual DC (that may be time varying) it should be removed immediately after input of the signal. Also, the *linear* interpolation between s_n and s_{n-1} can be improved. However, all such algorithms rapidly deteriorate in performance with SNR degradation. Noise pulses can look like transitions or alternatively hide true transitions, and even when a transition is properly identified its correct location becomes obscure. What we really need is a method that exploits the entire signal, not just those few signal values that straddle a transition; exploiting the entire signal in order to derive a frequency requires spectral analysis or narrow band-pass filtering.

Were the data to alternate like 01010101, the transmitted signal would be a square wave, and its Fourier series would consist of a basic sinusoid at half the baud rate, and all odd harmonics thereof. Even in severe noise this harmonic structure would be easily discernible and by band-pass filtering a sinusoid related to the desired clock could be recovered. A more direct method would be to differentiate the signal (accenting the transitions), and to take the absolute value (removing the direction of the transition) thus creating a pulse train whose spectrum has a strong line at precisely the baud rate. Of course the differentiation operation is very sensitive to noise but the baud line will be strong enough to stand out.

When the NRZ data is random the differentiation and absolute value operations produce a train of pulses similar to that of the alternations, but with many of the pulses missing. The basic frequency of this signal is still the baud rate, but the baud line in the spectrum is not as strong. However, as long as there are enough transitions the baud rate can still be determined. Using a PLL is helpful, since it is designed to lock onto approximately known frequencies in noisy signals.

If there are long stretches of constant zeros or ones in the data the baud spectral line will tend to disappear, and no amount of filtering will be able to bring it back. We mentioned previously that by using a bit scrambler we can eliminate long runs of 1 bits. The most popular scrambler in use is the two-tap self-synchronizing LFSR scrambler, depicted in Figure 18.23.A. Why is this scrambler called self-synchronizing? Contrast it with the alternative method of running the LFSR locked upon itself (see Section 5.4) to create an LFSR sequence, and xoring the data with this sequence. That method also increases the number of alternations in sections where there are long

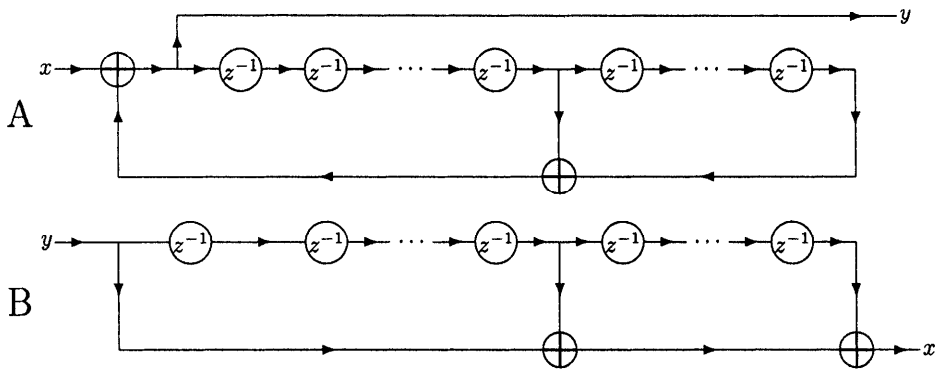


Figure 18.23: Two-tap scrambler and descrambler. In (A) we see the LFSR-based scrambler. The output is composed of the xor of the input bit with two previous input bits. The length of the shift register equals the delay of the oldest bit required. In (B) we see the descrambler. Note that there is no feedback and hence bit errors do not propagate without limit.

runs of either 1 or 0 bits, but in order to properly decode the sequence we have to use the proper phase of the LFSR sequence. The LFSR descrambler depicted in Figure 18.23.B correctly recovers the bits without the need for synchronizing an LFSR to an external clock.

When the signal is PSK another trick is popular. Rather than basing the baud rate recovery on the phase demodulation, we base it on the received signal's amplitude. It may seem surprising that the AM demodulation of a PSK signal contains any information at all; doesn't a PSK signal have constant amplitude? It does, but sharp phase transitions require wide bandwidth, and when a PSK signal is filtered to the channel bandwidth this high-frequency energy is lost, resulting in amplitude dips at the phase transitions. The amplitude demodulation is thus constant except for at the phase transitions, and its dips provide a reliable indication of the phase transitions.

We turn now to the recovery of the carrier frequency of a PSK signal. Were the data being transmitted to be constant (and no scrambler used), then the n PSK signal would be a sinusoid, and its spectrum would consist of a single discrete line the frequency of which is easily determined. However, in the more interesting case of a signal carrying information, the frequent phase jumps widen the spectral line into a broad (sinc squared) band centered around the carrier frequency. The precise carrier frequency is no longer evident.

Assume for just a moment that the constellation is chosen so that the signal points are $e^{i\pi \frac{k}{n}}$ (e.g., for BPSK ± 1 and for QPSK $\pm 1, \pm i$). It is obvious

that these signal points are precisely the n^{th} roots of unity, and so raising them to the n^{th} power gives one. The usual convention is rotated by 45° with respect to this, (i.e., all the signal points are multiplied by $e^{i\frac{\pi}{4}}$). Hence in the usual convention all the signal points raised to the n^{th} power still return the same value, only now that value is $e^{i\frac{n\pi}{4}}$. Thus for BPSK squaring the signal points at 45° and 225° gives i and for QPSK raising any of the points $45^\circ, 135^\circ, 225^\circ, 315^\circ$ to the fourth power gives -1 . The important fact is that raising any of the constellation points to the n^{th} power returns the same value.

Now the signal in the time domain is $\sin(\omega t + \phi_k)$, where the ϕ_k are the n possible signal phases. It is clear from the result of the previous paragraph that raising the signal to the n^{th} power on a sample-by-sample basis will wipe out the ϕ_k dependence; and so the n^{th} root of this will be a simple sine of constant phase at the carrier frequency.

EXERCISES

- 18.15.1 Assume that a signal contains no runs of single bits, but only runs of two, three and longer. Can the baud rate be recovered? Suggest a method.
- 18.15.2 Generate a PSK signal with random data, limit its bandwidth by FIR low-pass filtering, and perform amplitude demodulation. Do you see the AM dips? Now filter the AM using a narrow IIR filter centered at the nominal baud rate. Empirically determine the delay between the zero crossings of the sinusoidal output of this filter and the center of the symbols. Does this system give accurate baud rate recovery?
- 18.15.3 Can baud or carrier recovery be performed on signals attaining the Shannon capacity?
- 18.15.4 In systems with very high baud rate there can be a problem in providing the timing on a second pair of wires. What is this problem and how can it be overcome?

18.16 Equalization

We discussed adaptive modem equalizers in Section 10.3. The problem with standard (linear) equalizers for telephone modems is that near the band edges (under 400 Hz or above 3600 Hz) there can be 10 to 20 dB of attenuation. In DSL modems the higher frequency ranges can be attenuated by 50

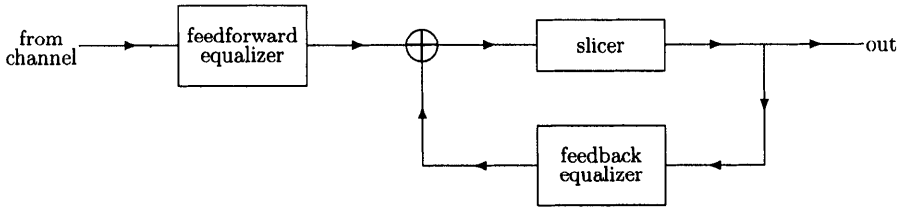


Figure 18.24: Decision feedback equalizer. The DFE consists of two FIR filters, one in the feedforward path and one in the feedback path. The slicer becomes an integral part of the equalizer.

dB or more! In order to compensate for this loss, an equalizer at the input of the demodulator must provide that much gain. Unfortunately, this gain is applied not only to the desired signal, but to the noise as well, causing significant noise enhancement. No filter is able to overcome this problem, since from linearity the sum of the signal and noise are filtered separately and identically; however, we may be able to build a nonlinear system that can apply gain to the signal without enhancing the noise too.

In Figure 18.24 we see such a nonlinear system, its nonlinearity deriving from the slicer. It is easy to see why this system, called a **Decision Feedback Equalizer (DFE)**, can effectively combat ISI without appreciable noise enhancement. Assuming the output of the slicer to be correct, it is essentially the signal that was originally transmitted. Based on this reconstructed signal we can reproduce the intersymbol interference as caused by the channel response and subtract it from the signal. This is performed by the feedback equalizer in the figure.

There are two problems with decision feedback equalizers. First, if the slicer *does* make mistakes, then (at least theoretically) the feedback can cause the system to deviate more and more from correct behavior. This lack of stability is rarely seen in practice for a DFE initially trained on known data and continuously updated. A more problematic aspect of placing the slicer into the equalization path is that its decisions do not take TCM (see Section 18.19) into account. A TCM modem does not have reliable decisions until much later, long after the DFE needed them.

The Tomlinson equalizer explored in exercise 10.3.1 is another solution to the noise enhancement problem. By placing the inverse filter at the modulator, before the noise is added, the whole problem becomes moot.

EXERCISES

- 18.16.1 Taking the ISI to be weak and from precisely one previous symbol, show that each constellation point splits into a small cluster of points that resembles the entire constellation. What happens if the ISI is from two previous symbols? What happens when the ISI is large and its duration long?
- 18.16.2 Simulate a QPSK signal traversing a noisy channel sharply attenuated at its edges. Compare the optimal linear equalizer with the optimal DFE.
- 18.16.3 An inverse filter at the modulator may cause the transmitted signal to reach values much larger than originally intended. The Tomlinson equalizer overcomes this by a modulo operation, and a compensating operation at the demodulator. Explain how this can be accomplished.

18.17 QAM

In Section 18.14 we saw that the bandwidth of an n PSK signal is not n dependent. Accordingly, we can achieve higher information transfer rates in a given bandwidth simply by increasing n . The problem is that for larger n the constellation points are closer together. Since channel noise causes the received signal phase to deviate from that transmitted, as depicted in the constellation plot of Figure 18.20, there is a limit on how close we can place constellation points. This is how the channel capacity theorem limits capacity for PSK signals. Were there to be no noise we could achieve arbitrarily large transfer rates by using large n ; were there no bandwidth limitation we could use BPSK and arbitrarily large baud rates.

Looking closely at the constellation plot of Figure 18.20 we can see a way out. The additive channel noise expands the constellation points into circular clouds in the I-Q plane, and our decision making is optimized by maximizing the Euclidean distance between constellation points. One way this can be done is by placing constellation points as shown in Figure 18.25. Here the symbols differ in both phase and amplitude. This type of signal (being simultaneously PSK and ASK), is sometimes called names like APSK, but more usually goes under the name **Quadrature Amplitude Modulation (QAM)**. Understanding the meaning of QAM requires thinking of the I and Q components as two independent PAM signals ‘in quadrature’. This is indeed another way of building a QAM signal; rather than altering the amplitude and phase of a single carrier, we can independently amplitude modulate a sine and its corresponding cosine and add the results.

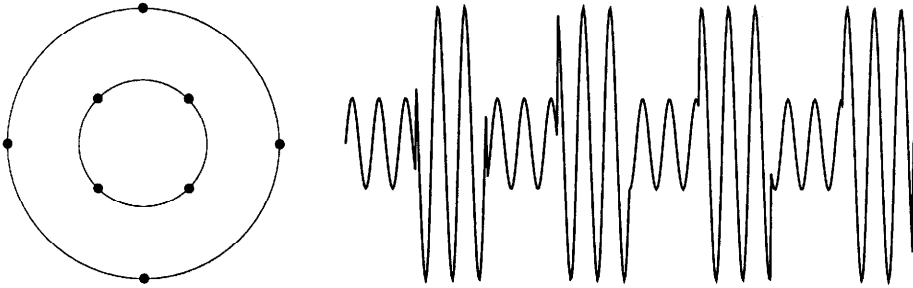


Figure 18.25: Two-ring constellation and signal. From either representation it can be seen that the symbols differ in both amplitude and phase. The two-ring signal allows precisely two different amplitudes. The phases are chosen to maximize the minimum distance between constellation points.

Let's calculate how much we gained by using QAM. The minimum distance between points in the 8PSK constellation is $2 \sin(\frac{\pi}{8}) \approx 0.765$. The two-ring constellation consists of the symbols $(1, 1)$, $(-1, 1)$, $(-1, -1)$, $(1, -1)$, $(3, 0)$, $(0, 3)$, $(-3, 0)$, and $(0, -3)$ and so its minimum distance is 2. However, this is not a fair comparison since the average energy of the two-ring constellation is higher than that of 8PSK. We can always increase the minimum distance by increasing the energy; were we to put the 8PSK on a circle of radius 4 the minimum distance would be $4 \cdot 0.765 > 3!$ The proper way to compare two constellations is to first normalize their average energies. Every point of the 8PSK constellation has unit energy, so the average energy is obviously unity. The two-ring constellation's energy can be easily calculated as follows. There are four points with energy $(\pm 1)^2 + (\pm 1)^2 = 2$ and four points with energy $(\pm 3)^2 + 0^2 = 9$, hence the average energy is $(4 \cdot 2 + 4 \cdot 9)/8 = 5\frac{1}{2}$. In order to force the energy to unity we need only divide all the symbol coordinates by the square root of this energy, i.e. by about 2.345. Instead of doing this we can directly divide the minimum distance by this amount, and find that the normalized minimum distance is $2/2.345 = 0.852 > 0.765$. This increase in minimum distance is due to better exploitation of the geometrical properties of two-dimensional space.

The two-ring constellation managed to increase the minimal distance between constellation points without increasing the energy. In this way a demodulator will make fewer errors with the same SNR, or alternatively we can attain the same error rate with a lower SNR. This is the goal of a good constellation, to maximize the minimum two-dimensional Euclidean

distance between points for a constellation with given energy. The problem is thus purely a geometric one, and some of the solutions presently in use are shown in Figure 18.26.

It has become conventional to use square constellations with odd integer coordinates. For example, the 16QAM constellation consists of symbols $(-3, 3), (-1, 3), \dots, (1, -3), (3, -3)$. What is the energy of this constellation? It has four points with energy $(\pm 1)^2 + (\pm 1)^2 = 2$, eight with $(\pm 1)^2 + (\pm 3)^2 = 10$, and four with $(\pm 3)^2 + (\pm 3)^2 = 18$, so that the average is $(4 \cdot 2 + 8 \cdot 10 + 4 \cdot 18)/16 = 10$. Since the unnormalized minimum distance is 2, the normalized minimum distance is $2/\sqrt{10} \approx 0.632$. This is lower than that of the previous constellations, but each symbol here contains 4 bits of information, one bit more than that of the eight-point constellations.

We will see in the next section that it is easiest to build slicers for square constellations, but rectangular constellations have a drawback. The corners have high energy, and may even be illegal in channels with maximum power restrictions. The optimum constellation boundary would be a circle, and this is closely approximated by the V.34 constellation. The cross-shaped constellations are a compromise whereby the worst offenders are removed, the slicer remains relatively simple, and the number of points in the constellation remains a power of two.

EXERCISES

- 18.17.1 Why are cross constellations used for odd numbers of bits per symbol and square-shaped constellations for even numbers?
- 18.17.2 Write a program to compute the average energy and normalized minimum distance for all the constellations in Figure 18.26.
- 18.17.3 Can you write a program that outputs the points in the V.34 constellation? (Hint: There are 1664 odd-integer-coordinate points bounded by a circle.)
- 18.17.4 Show that PAM constellations with m bits have average energy $E = \frac{1}{3}(4^m - 1)$ and hence require about four times (6 dB) more energy to add a bit. Repeat the calculation for square QAM constellations.
- 18.17.5 Square QAM constellations suffer from the same 90° ambiguity as nPSK. Show how differential encoding can be combined with 16QAM.
- 18.17.6 Some people have suggested using hexagonal constellations. What are these and why have they been suggested?
- 18.17.7 Prove that by squaring a QAM signal one can recover the baud rate. Prove that taking the fourth power of a QAM signal enables carrier recovery. Show that the rounder the constellation the harder it is to recover the carrier.

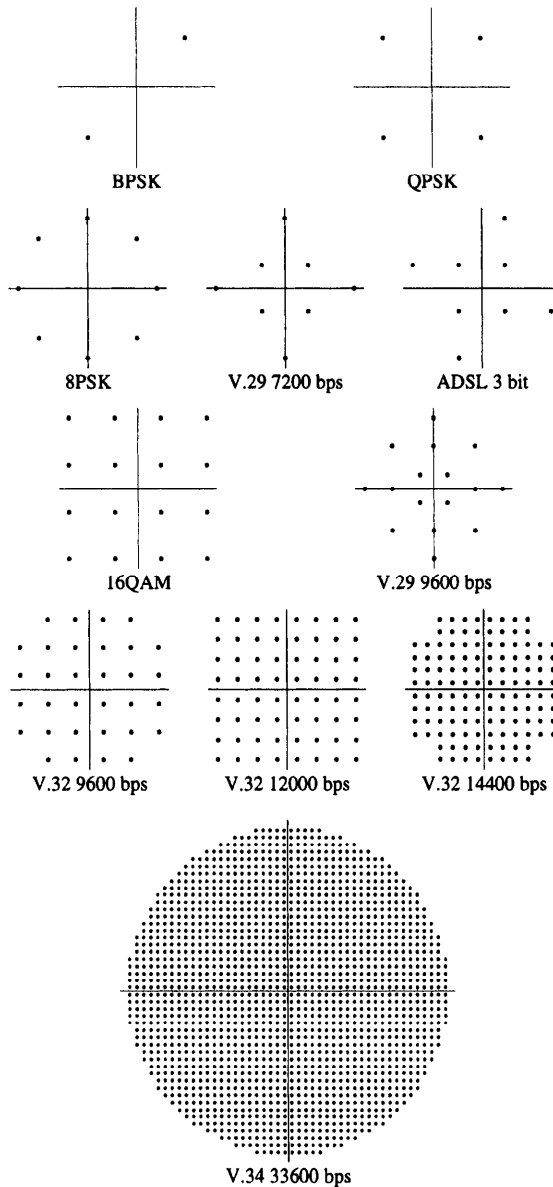


Figure 18.26: Some popular QAM constellations. The BPSK and QPSK constellations are used by the simplest modems (e.g., V22bis telephone-grade modem at 2400 b/s). The second row all have three bits per symbol, and the third row all four bits per symbol. The fourth row contains the constellations used by the V.32bis standard modem, with 5, 6, and 7 bits per symbol. The V.34 1664-point constellation has been magnified for clarity.

18.18 QAM Slicers

The slicer is the element in the QAM demodulator that is responsible for deciding which symbol was actually transmitted. The slicer comes after AGC, after carrier and symbol timing recovery, after equalization and after symbol rate resampling, although it is intimately related to all of these. Its input is a point in two-dimensional space (Figure 18.27 gives an example) and its output is a symbol label.

Figure 18.27 demonstrates the difficulty of the slicer's task. The transmitted constellation points have become contaminated by noise, distortion, and uncorrected ISI from the channel, and possibly by nonoptimalities of the previous stages of the demodulator. The combined effect of all these disturbances is that the constellation points have expanded into 'clouds' centered on their original positions. If the residual noise is too large, the clouds join, the constellation becomes unrecognizable, and the demodulator can no longer reliably function. The obvious requirement for dependable demodulation is for the radii of the noise clouds to be smaller than half the distance between constellation points.

The noise clouds are not always circularly symmetric, for example, when the demodulator has not properly locked on to the carrier frequency rota-

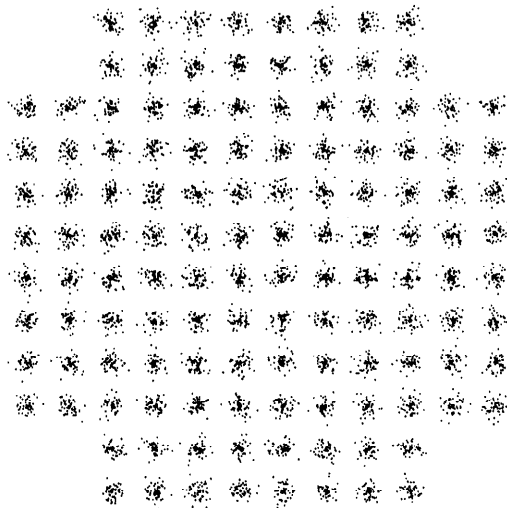


Figure 18.27: Input to the slicer. The input represents three seconds of received two-dimensional points from a V.32bis modem operating at 14.4 Kb/s. The constellation is readily recognizable to the eye, but the slicer's decisions are not always clear cut.

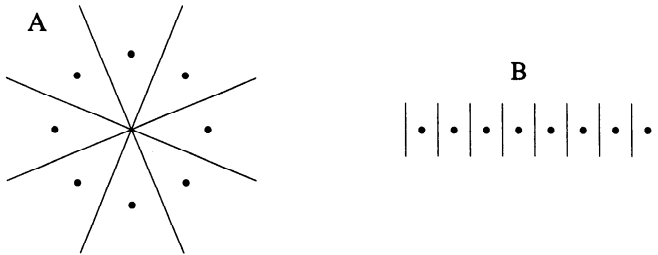


Figure 18.28: Operation of slicer for 8PSK. In (A) we see that the eight Voronoy regions are pie-slice in shape, being only phase dependent. In (B) only the phase of the same constellation is depicted, and we see that the slicer has been reduced to a quantizer.

tional smearing dominates. However, we'll assume that the clouds *are* circularly symmetric, as would be the case if the major contribution is from additive channel noise. Under this assumption the optimal operation of the slicer is to choose the constellation point that is closest to the received point. So a straightforward slicer algorithm loops on all the N constellation points and selects that constellation point with minimal distance to the received point. This algorithm thus requires N computations of Euclidean distance (sum of squares) and comparisons. Were the constellation points to be randomly chosen this complexity would perhaps be warranted, but for the types of constellation actually used in practice (see Figure 18.26) much more effective algorithms are available.

The principle behind all efficient slicers is exploitation of *Voronoy region* symmetries. Given an arbitrary collection of points, the Voronoy region associated with the n^{th} point is the set of all points closer to it than any of the other points; the collection of all the Voronoy regions tessellate the plane. For the nPSK modem, having all its constellation points on the unit circle, it is not hard to see that the Voronoy zones are 'pie-slice' in shape (see Figure 18.28.A). The optimal slicer will slice up space into these pie slices and determine into which slice a received point falls. In particular we needn't consider the amplitude of the received point, and the optimal decision involves only its phase (as was assumed when we originally discussed the nPSK demodulator). When depicted in one-dimensional (phase-only) space (see Figure 18.28.B) the decision regions are even simpler. Neglecting the wrapping around of the phase at the edges, the slicing is reduced to simple inequalities. By correctly choosing the scale and offset, the slicing can even be reduced to simple quantizing!

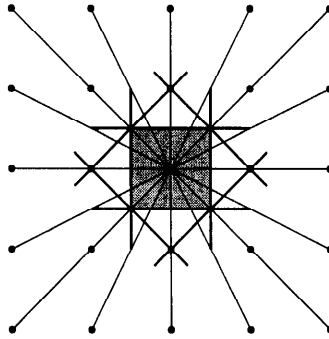



Figure 18.29: The Voronoy regions for square QAM constellations are square. To show this we investigate the immediate vicinity of an arbitrary symbol point, and connect this point with all neighboring symbols (the gray lines). We cut these lines with perpendicular bisectors (the dark lines) in order to separate points closer to the center symbol from those closer to the neighboring symbol. We shade in gray the area containing all points closest to the center symbol.

The slicer for n PSK was so simple that it could be reduced to a single quantization operation; but the more complex constellations are inherently two-dimensional. Many non-PSK constellations are based on square arrangements of symbol points, the Voronoy regions for which are themselves square, as can be seen from Figure 18.29. Square Voronoy regions are only slightly more complex to manipulate than their one-dimensional counterpart, evenly spaced points on the line. By properly choosing the scale and offsets the decision algorithm is reduced to independent quantizing along both axes. Of course some points quantize to grid points that are outside the constellation, and in such cases we need to project the decision back toward a constellation point. For cross-shaped constellations a slight generalization of this algorithm is required; for example, in Figure 18.27 we observe a point in the lower-left corner and another in the upper right that have to be specially handled.

We have succeeded in simplifying the slicer from the straightforward algorithm that required N distance computations and comparisons to one or two quantizations and some auxiliary comparisons. These simplifications depend on the geometric structure of the constellation. Indeed the efficiency of the slicer is often taken as one of the major design considerations when choosing the constellation.

EXERCISES

- 18.18.1 Write a routine that efficiently implements a slicer for square 16QAM. Be careful about how you handle inputs that quantize to grid points outside the constellation. Are divisions required?
- 18.18.2 Write a routine that efficiently implements a slicer for the cross-shaped 128-point constellation used in V.32bis. Inputs that quantize to grid points in the corners should only require a single additional comparison.
- 18.18.3 What are the Voronoy regions for the 16-point V29 constellation?
- 18.18.4 What are the Voronoy regions for a hexagonal constellation such as the 12 point ? How can a slicer be efficiently implemented here?

18.19 Trellis Coding

In Figure 18.9.C we saw how the error correction encoder could be placed before the modulator and the error correction decoder after the demodulator in order to protect the transmitted information against errors. As was mentioned in Section 18.7 this separation of the error correcting code from the modulation is not guaranteed to be optimal. In our discussion of error correcting codes in Section 18.9, we saw how ECCs increase the number of bits that need be transferred. This increase directly conflicts with our attempt at transferring the original amount of information in minimum time, but is perhaps better than receiving the information with errors and having to send it again.

Is there some better way of combining the ECC and modulation techniques? It is easy to see that the answer must be affirmative. As a simplistic example, consider a bilevel PAM signal protected by a parity check. Parity check of the demodulated bits can only be used to *detect* a single bit error, while if we observe the signal input to the demodulator we may be able to make a good guess as to which bit is in error. For example,

0.10	0.92	0.05	0.49	1.02	0.94	0.08	0.04	← input signal
0	1	0	0	1	1	0	0	← demodulated bits
0	1	0	1	1	1	0	0	← corrected bits

we have to correct a single demodulated bit and it is obvious which bit is the best candidate for correction.

So we believe that there *is* a way to combine error correction and modulation, but the two disciplines are so different that it is not obvious what that way is. It was Gottfried Ungerboeck from IBM in Zurich who, in the early 1980s, came up with the key idea of combining convolutional (trellis) codes with *set partitioning*. Set partitioning refers to recursively dividing a constellation into subconstellations with larger distance between nearest neighbors. What does this accomplish? If we know which subconstellation is to be transmitted then it is easier to determine which point was transmitted even in the presence of significant noise. How do we know which subconstellation is to be transmitted? Ungerboeck's suggestion was to divide the input bits to be transmitted into two groups, one group determining the subconstellation and the other the point in the subconstellation. If we err regarding which subconstellation was transmitted we are potentially worse off than before. So we protect the decision as to the subconstellation with an error correction code!

We will demonstrate the basic idea with the simplest possible case. A QPSK system sends two bits per symbol (that we call *A* and *B*) and has a minimal distance of $\sqrt{2} \approx 1.414$. In the proposed TCM system we expand the constellation to 8PSK using a 1/2 rate convolutional code. We will keep the same baud rate so that the bandwidth remains unchanged, but the minimum distance between constellation points is decreased to $2 \sin(\frac{\pi}{8}) \approx 0.765$. The set partitioning is performed as follows. First, as depicted in Figure 18.30.A, the eight points are partitioned into two QPSK subconstellations, named 0 and 1. This particular way of partitioning is optimal since there is no other way to partition the eight points into two subconstellations that will give

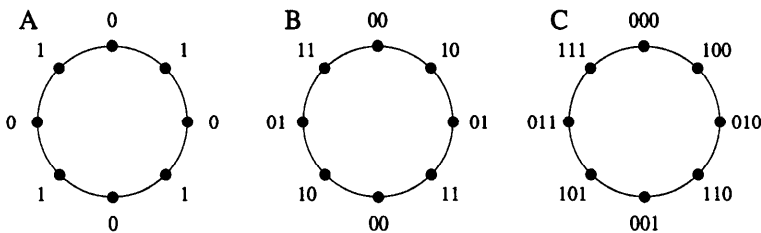


Figure 18.30: Set partitioning for the simplest TCM system. In step (A) the eight points are partitioned into two QPSK subconstellations, named 0 and 1. In step (B) each of the QPSK subconstellations is partitioned into two BPSK subsubconstellations, the 0 subconstellation into 00 and 01, and the 1 subconstellation into 10 and 11. In (C) the subsubconstellation points themselves are labeled.

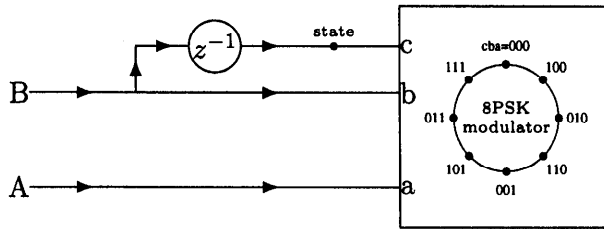


Figure 18.31: The modulator for the simplest TCM system. One of the input bits is passed directly to the 8PSK modulator, and the other is encoded by a trivial rate 1/2 convolutional code. The ‘state’ is the previous bit and the outputs are the present bit and the state.

greater Euclidean separation. In the next step (Figure 18.30.B) we partition each QPSK subconstellation into two BPSK subsubconstellations. The four points of subconstellation 0 are divided into subsubconstellations 00 and 01, and those of the 1 subconstellation into 10 and 11. In Figure 18.30.C we label the subsubconstellation points by suffixing the subsubconstellation label with a 0 or 1 bit.

Now that we have completely partitioned the constellation, we can proceed to build the modulator. Let’s name the two bits that were input to the uncoded QPSK constellation, A and B . In the TCM modulator, input bit A is passed directly to the 8PSK modulator (where it becomes a), and it will determine the point in the subsubconstellation. Bit B , which will determine the subsubconstellation, first enters a convolutional encoder, which outputs two bits b and c . What ECC should we use? We need a convolutional ECC that inputs a single bit but outputs two bits, that is, a rate 1/2 code. The simplest such code is the trivial code of Figure 18.12. Output bit b will simply be B , while c will be precisely the internal state of the encoder (i.e., the previous value of B). The bits a , b , and c determine the constellation point transmitted, as can be seen in Figure 18.31. It’s easy to see that if the encoder is presently in state zero then only constellation points $0 = 000$, $2 = 010$, $4 = 100$ and $6 = 110$ can be transmitted, while if the state equals one then only constellation points $1 = 001$, $3 = 011$, $5 = 101$, and $7 = 111$ are available.

Let’s draw the trellis diagram for this simple TCM modulator. It is conventional to draw the trellis taking all the inputs bits (in our case A and B) into account, although only some of them (B) enter the encoder and the others (A) do not. TCM trellis diagrams thus have ‘parallel transitions’, that is, multiple paths between the same states. For our encoder the new

state	input (BA)	output (cba)	new state
0	0(00)	0(000)	0
0	1(01)	1(001)	0
0	2(10)	2(010)	1
0	3(11)	3(011)	1
1	0(00)	4(100)	0
1	1(01)	5(101)	0
1	2(10)	6(110)	1
1	3(11)	7(111)	1

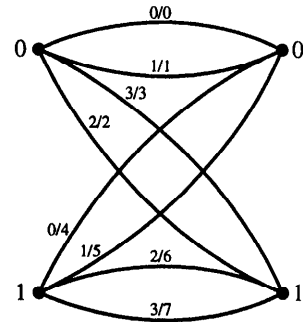


Figure 18.32: The trellis table and diagram for the simplest TCM system. The table gives the output and new state given the present state and the input bits. The diagram depicts a single time step. Between each state and each new state there are two parallel transitions, labeled BA/cba.

state will be whatever B is now. So assuming we are presently in state zero, we will remain in state zero if the input bits AB are either $0 = 00$ or $2 = 10$; however, if the inputs bits are $1 = 01$ or $3 = 11$ the state will change to one. It is straightforward to derive the table and diagram in Figure 18.32 which connect all the relevant quantities.

Now let's see if this TCM technique is truly more resistant to noise than the original QPSK. What we would really want to compare are the energies of noise that cause a given bit error rate (BER). However, it is easier to calculate the ratio of the noise energies that cause a minimal error event. For this type of calculation we can always assume that a continuous stream of zeros is to be transmitted. For the QPSK signal the minimal error is when the 00 constellation point is received as one of the two neighboring symbols (see Figure 18.33). This corresponds to a noise vector \underline{n} of length 1.414 and of energy $|\underline{n}|^2 = 2$. What is the minimal error for the TCM case?

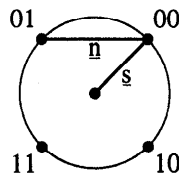


Figure 18.33: The minimal error event for uncoded QPSK. The symbol \underline{s} was transmitted but the symbol $\underline{s} + \underline{n}$ was received. The noise vector is of length $\sqrt{2}$ and hence of energy 2.

Without error we would stay in state zero all the time and receive only the 0 constellation point. Since we are in state zero there is no way we could receive an odd-numbered constellation point, so an error must result in receiving one of the points 2, 4, or 6, with 2 and 6 being the closest. Both 2 and 6 correspond to the same noise energy as in the QPSK case, but also switch the state to one, so that the next constellation point received will be odd numbered. The odd numbered points of minimal distance from the true point of zero are 1 and 7, both with distance about 0.765 and energy 0.5858. Thereafter the state reverts to 0 and the proper constellation point may be received again. So the combined noise energy of the two errors that make up this error event is about 2.5858. If the constellation point labeled 4 is mistakenly received the state does not change, but this corresponds to noise of energy $2^2 = 4$. So the minimal noise energy is 2.5858 as compared with 2 for the uncoded case. This corresponds to an improvement of a little over 1.1 dB.

By using more complex ECCs we can get more significant gains. For example, the four-state code of Figure 18.34 is described in Figure 18.35. It is not hard to see that the minimal energy error event occurs when we substitute the constellation point 001 for 000, taking the parallel transition and remaining in state 0. The energy of this event is $2^2 = 4$ rather than 2 for a coding gain of 3 dB. By using even more complex ECCs we can achieve further coding gains, although the returns on such computational investment decrease.

The first standard modem to use TCM was the CCITT V.32 modem at 9600 b/s. In order to make TCM practical, the trellis code must be made

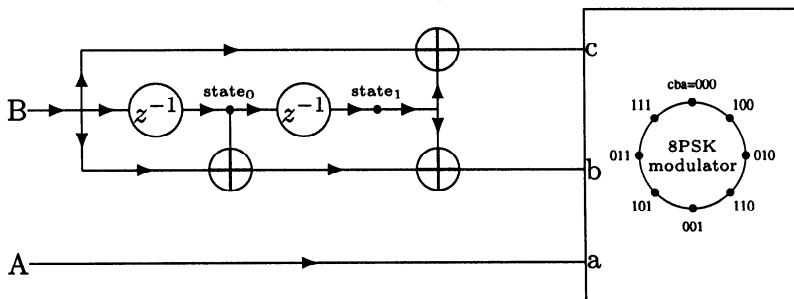


Figure 18.34: The modulator for a four-state 8PSK TCM system. One of the input bits is passed directly to the 8PSK modulator, and the other is encoded by a two state $1/2$ convolutional code. The 'state' consists of the previous two bits and the outputs are formed by binary additions (xor) of the present bit and the state bits.

state	BA	cba	new state
0(00)	0(00)	0(000)	0(00)
0(00)	1(01)	1(001)	0(00)
0(00)	2(10)	6(110)	2(10)
0(00)	3(11)	7(111)	2(10)
1(01)	0(00)	6(110)	0(00)
1(01)	1(01)	7(111)	0(00)
1(01)	2(10)	0(000)	2(10)
1(01)	3(11)	1(001)	2(10)
2(10)	0(00)	2(010)	1(01)
2(10)	1(01)	3(011)	1(01)
2(10)	2(10)	4(100)	3(11)
2(10)	3(11)	5(101)	3(11)
3(11)	0(00)	4(100)	1(01)
3(11)	1(01)	5(101)	1(01)
3(11)	2(10)	2(010)	3(11)
3(11)	3(11)	3(011)	3(11)

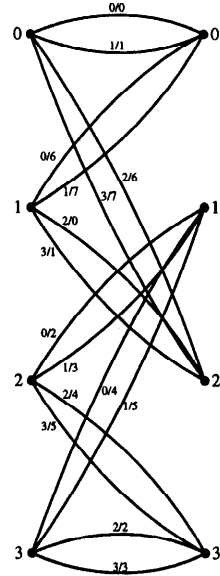


Figure 18.35: The trellis table and diagram for the four-state 8PSK TCM system. Once again there are two parallel transitions between each two states, labeled by BA/cba.

invariant to rotations by 90° . This feat was accomplished by Wei, although it required a nonlinear code.

EXERCISES

- 18.19.1 The simplest TCM described in the text uses the replicating convolutional code with a particular phase, outputting at a given time the present input bit and the previous one. What would happen if we output the present bit twice instead?
- 18.19.2 Calculate the noise energies for the different possible error events for the four-state 8PSK TCM system and show that the minimal event is indeed that mentioned in the text.
- 18.19.3 Devise a set partition for the 16 QAM constellation. To do this first partition the 16 points into two subconstellations of 8 points each, with each subconstellation having maximal minimal distance. What is this distance and how does it relate to the original minimal distance? Now continue recursively until each point is labeled by four bits. By how much does the minimal distance increase each time?
- 18.19.4 Why does TCM simplify (or even eliminate the need for) the slicer?

18.20 Telephone-Grade Modems

The history of telephone-grade modems is a story of rate doubling. The first telephone modem of interest was the Bell 103 and its internationally recognized standard version called V.21. This modem used FSK and allowed full-duplex operation of up to 300 b/s. The V.21 originating modem (called channel 1) uses frequencies 980 and 1180 Hz while the answering modem (channel 2) uses 1650 and 1850 Hz. V.21 channel 2 is still in widespread use today as the medium for negotiation between fax machines; such information as the maximum speed supported, the paper size, the station telephone numbers, and other identification are sent at 300 b/s before the higher-speed modem sends the image. This same FSK signal then cuts in again with 'end of page' and 'end of transmission' messages. You can recognize the V.21 as the 'brrrr' sound before and after the 'pshhhhh' sound of the higher-speed transmission.

A breakthrough came with the introduction of the Bell 202 and its ITU version, V.23. The V.23 FSK modem attained full-duplex 600 b/s and 1.2 Kb/s but only over special four-wire lines; on regular dial-up two-wire circuits these rates were for half-duplex use only. The rate increase as compared to V.21 was due to the use of an equalizer, although it was a fixed compromise equalizer designed for a 'typical line' and implemented as an analog filter. It is an amazing statement about conservativeness that the Bell 202 signal is still in common use. In many places it appears before the phone is picked up, carrying the number of the calling party for display to the called party.

The Bell 201 modem came out in 1962, and was later standardized by the ITU as V.26. This modem was QPSK and could attain half-duplex 2.4 Kb/s by using 1200 baud and 2 bits per symbol. The Bell 201 was the last of the modems built of analog components; it had a carrier of 1800 Hz and a fixed compromise equalizer. Its innovation was the use of carrier recovery.

Logically, if not historically, the first of the new breed of DSP modems was the V.22 QPSK modem, which reached 1.2 Kb/s full-duplex over regular dial-up lines. By upgrading the constellation from QPSK to square 16QAM, V.22bis was able to reach 2.4 Kb/s. The baud rate for these modems is 600, with one side using a carrier of 1200 Hz and the other 2400 Hz.

By using 8PSK, Milgo was able in 1967 to extend the half-duplex bit rate to 4.8 Kb/s. The baud rate was 1600, the carrier 1800 Hz, and the original version had an adjustable equalizer. Unfortunately, the adjusting had to be done by hand using a knob on the modem's front panel. This modem was standardized in 1972 as V.27.

The next major breakthrough was the 1971 introduction by Codex and the 1976 standardization of V.29. This modem achieved a half-duplex rate of up to 9.6 Kb/s by using an unusual 16-point QAM constellation with carrier 1700 Hz and baud rate 2400. Another innovative aspect of this modem was its adaptive equalizer. This modem is still popular as the 9600 fax, where half-duplex operation is acceptable.

Surprisingly, more than two decades after Shannon had predicted much higher rates, technology appeared to stop at 9.6 Kb/s. Popular wisdom believed Shannon's predictions to be overoptimistic, and efforts were devoted to implementational issues. Then in 1982, Ungerboeck published his paper revealing that an eight-state TCM code could provide a further 3.6 dB of gain, and the race toward higher rates was on again. The next step should have been to double the 9.6 Kb/s to 19.2 Kb/s, but that leap wasn't achieved. At first the V.33 modem achieved 14.4 Kb/s full-duplex on a four-wire line, and its two-wire half-duplex version (V.17) is the standard 14.4 fax used today. Next, with the introduction of DSP echo cancelling techniques, V.32bis achieved that same rate for full-duplex on two wires. All of these modems use $f_c = 1800$ Hz, $f_b = 2400$ Hz and a 128-point cross-shaped constellation. Since one of the seven bits is used for the coding, the remaining six bits times 2400 baud result in 14.4 Kb/s. These modems also provide 64-point square and 32-point cross constellations for 12 Kb/s and 9.6 Kb/s respectively.

V.32bis had attained 14.4 Kb/s, so the next major challenge, dubbed V.fast, was to attempt to double this rate (i.e., to attain 28.8Kb/s). For several years different technologies and signal processing techniques were tried, until finally in 1994 the V.34 standard was born. V.34 was a quantum leap in signal processing sophistication, and we will only be able to mention its basic principles here. Due to the complexity of the signal processing, most V.34 modems are implemented using DSP processors, rather than special-purpose DSP hardware.

The original ITU-T V.34 specification supported all data rates from 2.4 Kb/s to 28.8 Kb/s in increments of 2.4 Kb/s, and an updated version added two new rates of 31.2 and 33.6 Kb/s as well. Two V.34 modems negotiate between them and connect at the highest of these data rates that the channel can reliably provide.

We have already seen in Figure 18.26 the constellation used by V.34 for 33.6 Kb/s operation. For lower rates subconstellations of this one are used. This constellation is by far the most dense we have seen.

Recall from equation (18.20) that the information transfer rate in any given channel is maximized by matching the PSD to the channel characteristics. One of the problems with the modems up to V.32bis is that they

baud rate (Hz)	low carrier (Hz)	high carrier (Hz)	maximum data rate (Kb/s)
2400	1600	1800	21.6
2743*	1646	1829	26.4
2800*	1680	1867	26.4
3000	1800	2000	28.8
3200	1829	1920	31.2
3429*	1959	—	33.6

Table 18.3: The basic parameters for V.34. The baud rates marked with an asterisk are optional. Each baud rate, except the highest, can work with two possible carrier frequencies.

have a single predefined carrier and single baud rate, and hence their PSD is predetermined. The PSD of V.32bis stretches from $2400 - 1800 = 600$ Hz to $2400 + 1800 = 3000$ Hz irrespective of the channel characteristics. V.34 provides six possible baud rates (three mandatory and three optional) as well as nine carrier frequencies. A V.34 modem starts by probing the channel with a probe signal that creates the distinctive ‘bong’ noise you hear when trying to connect with a V.34 modem. This probe signal (see exercise 2.6.4) consists of a comb of sinusoids spaced 150 Hz apart from 150 Hz to 3750, except that the 900, 1200, 1800 and 2400 Hz tones have been removed. Using the probe signal the receiving modem can determine the frequency-dependent SNR, decide on the maximum data rate that can be supported with reasonably low bit error rate, and inform the transmitting modem which carrier and baud rate best match the channel. The possible baud rate and carriers are given in Table 18.3.

The second half of the channel capacity theorem specifies how the signal that maximizes information transfer rate should look. It should appear as white noise other than the water-pouring filtering. The suboptimality of V.32bis can be easily ascertained by observing its spectrum. With V.34 techniques were added to whiten the modem’s spectrum.

Looking at Figure 18.3 we note that the data rate is not always an integer multiple of the baud rate (i.e., there is a noninteger number of bits per symbol). For example, the 33.6 Kb/s maximum bit rate requires 8.4 bits per symbol at 3429 baud. This feat is accomplished using a *shell mapper*.

In ordinary QAM the constellation points are used with equal probability, so that the received (imperfectly equalized) I-Q plot is homogeneous inside a disk. A noise signal would have its I-Q plot distribution decrease as a Gaussian function of the radius. We can imitate this behavior by dividing the constellation into concentric circles called shells, and, based on the data to be transmitted, first choose a shell and then the point within the

shell. By using an algorithm that prefers interior shells we can transmit the constellation with a more Gaussian distribution.

V.34 also uses more powerful TCM codes than the Wei code used in V.32. The standard specifies three codes, a 16-state code (also invented by Wei) with 4.2 dB gain, a 32-state code with 4.5 dB gain and a 64-state code with 4.7 dB gain. All three of these codes are *four-dimensional*, meaning that they are based on four-dimensional symbols built up from two consecutive two-dimensional ones. Why should we want to group two transmitted symbols into a more complex one? The reason has to do with the geometry of n -dimensional space. Note that in one-dimensional space we can only place two points at unity distance from a given point, while in two-dimensional space there can be four such, and in n -dimensional space, $2n$ nearest neighbors. Thus for a given amount of energy, we can place more constellation points and thus carry more information, in higher-dimensional space. Of course the four-dimensional symbols are actually transmitted as two two-dimensional ones, but not every combination of consecutive two-dimensional symbols is possible.

In order to widen the usable bandwidth V.34 uses a more powerful equalization technique. Although DFE is capable of attaining close to the Shannon capacity, it has several drawbacks, the most important being that it is hard to combine with TCM. For V.34 a Tomlinson type equalizer was chosen instead. During the initialization a DFE is trained and the feedback coefficients sent to the modulator, where they are used as a 'precoder'. Taking the decision element out of the receive data path now makes integration of the equalizer with the TCM possible. A new mechanism called flexible precoding was invented to specifically integrate the precoder with the rest of the V.34 engine.

The logarithmic encoding used in the digital telephone system (μ -law or A-law) compresses the outer constellation points, making decisions difficult. V.34 has an option called 'nonlinear encoding' or 'warping' designed to combat these distortions. When enabled, the constellation is distorted, increasing the distance between outer constellation points, at the expense of decreasing that of more interior points.

The extremely sophisticated signal processing of the V.34 standard took years to develop and several years more to agree upon in standards committees. Yet, paradoxically, although for all intents and purposes V.34 at last approached the Shannon limit, it reigned supreme for only about a year. The next step, the step that would once again double the transmission speed from 28K to 56K, was just around the corner.

EXERCISES

- 18.20.1 Plot the PSDs of V.22bis, V.32, and the various modes of V.34 and compare spectral utilization.
- 18.20.2 Obtain diagrams of the initialization phases of V.32bis and V.34. Can you explain what happens at each stage?

18.21 Beyond the Shannon Limit

Can we beat the Shannon limit? No, there is no way of reliably communicating over an analog channel with the characteristics of the telephone channel at rates significantly higher than those of V.34. So how do V.90 (56Kb/s) modems work? What about G.lite (1 Mb/s), ADSL (8 Mb/s), and VDSL (52 Mb/s)?

These modems do not exceed the Shannon limit; they simply use a different channel. Even though they may be connected to the same phone lines that previously used a V.34 modem, what they see is different.

All of the previous modems assumed that the telephone network is comprised of (analog) twisted pairs of wire, with (analog) filters restricting the bandwidth. This was indeed once the case, but over the years more and more of the telephone system has become digital, transmitting conversations as 8000 eight-bit samples per second. Of course at every stage of this transformation of the telephone system the new equipment has emulated the old as closely as possible, so the new digital system looks very much like the old analog one; but in many places the only truly analog portion left is the 'last mile' of copper wire from the telephone office to the subscriber's house.

Were we able to transcend that last mile of copper we should be able to provide eight bits 8000 times per second, that is, an information transfer rate of 64 Kb/s. This is not surprising since this is the rate used internally by the telephone system for its own digital signals. The problem is that we are that mile or two away.

What would happen if someone at the telephone central office (CO) were to send us digital data at 64 Kb/s (i.e., a 256 level signal at 8000 samples per second)? Our telephone would interpret this *PCM modem* as a very loud noise, too loud in fact. In order to reduce crosstalk between neighboring cables, restrictions are placed on the average power that one can put onto the telephone wires. When voice is sent using these same 256 levels the

lower levels are more probable than the higher ones; when digital data is sent all levels are equally probable, resulting in a higher average power. There is another problem with this attempt at transmitting 64 Kb/s. In some parts of the world not all 8 bits are available. Every now and then the least significant bit is 'robbed' for other uses. This does not degrade voice quality very much, but would be most undesirable for data. In order to reduce the average power we can use shell mapping, and because of this, together with overcoming the robbed-bit phenomenon, we should not expect more than 7 bits 8000 times a second, for a grand total of 56 Kb/s.

What would happen if someone at the CO were to send us digital data at 56 Kb/s with an appropriate shell mapping? Would we be able to distinguish between these closely spaced levels? There would be ISI, but that could be overcome by an equalizer. We would need an echo canceller to remove our own transmission, but that too is well-known theory. It turns out that if we send digital levels directly on the pair of wires going to the other modem, then it is possible to recover the original levels. The data source need not sit physically in the telephone office, as long as its connection to that office is completely digital.

This is how the V.90 56 Kb/s modem works. A V.34 modem is used in the upstream direction, that is, from the consumer to the service provider. In the downstream direction a shell-mapped digital signal of up to 56 Kb/s is sent. This asymmetry is acceptable for many applications (e.g., for Internet browsing where the downstream often consumes ten times the data rate as the upstream). In a newer version dubbed V.92 even the upstream transmission tries to overcome the last mile and jump onto the digital link.

V.90 exceeds Shannon by exploiting the fact that the telephone system is no longer a 4 KHz analog channel, and thus the maximum possible rate is the 64 Kb/s used by the telephone system itself. Getting even higher than 64 Kb/s requires an even more radical departure from our model of the telephone system.

We have mentioned that the telephone network remains analog only in the 'last mile' to the subscriber, more formally called the 'subscriber line'. Now if we look at the frequency response of such subscriber lines, we find behaviors such as those of Figure 18.36. Although there is strong attenuation at high frequencies, the bandwidth is definitely higher than 4 KHz.

The 4 KHz restriction is actually enforced by filters at the telephone office, in order to enable multiplexing of multiple telephone conversations on a single carrier. There is nothing inherent in the subscriber line that recognizes this bandwidth restriction. So if we can place our modem before the filters and are allowed to use the subscriber line as a general-purpose

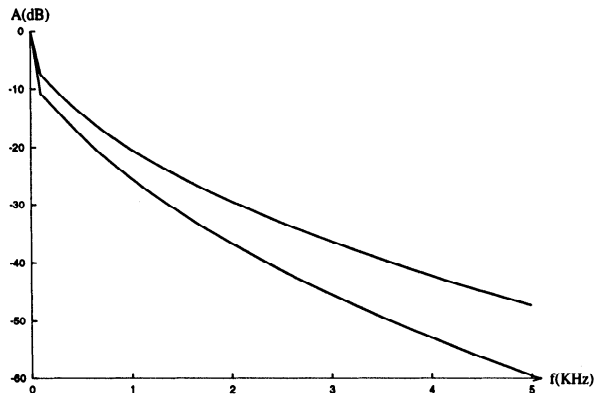


Figure 18.36: The attenuation for unshielded twisted-pair lines. Depicted is the line attenuation in dB for one kilometer of standard telephone-grade 24-gauge (upper curve) and 26-gauge (lower curve) unshielded cable. For two kilometers of cable the attenuation in dB is doubled.

cable, the so-called **Digital Subscriber Line (DSL)** modems can reach much higher capacities. Of course we may desire to continue to use the same subscriber line for our regular phone conversations, in which case a ‘splitter’ is placed at the end of the line. A splitter is simply a low-pass filter that passes the low frequencies to the phone, and a high-pass filter that delivers the high frequencies to the DSL modem.

The longer the subscriber’s cable, the higher the attenuation and thus the lower the capacity. Long lengths can support G.lite or ADSL rates, short lengths VDSL rates. The maximum capacity can be estimated using water-pouring calculations. The strong difference in attenuation between low frequencies and higher ones can be compensated for by an equalizer.

The DSL environment is more complex than we have described so far. In addition to the attenuation there is the acute problem of crosstalk. At high frequencies a significant portion of the signal energy leaks between adjacent cables, causing one DSL modem to interfere with another. The interferer may be located close by, as in the case of a bank of DSL modems at the telephone office, or remotely located but transmitting to a co-located DSL demodulator. The former case is called NEXT (**N**ear **E**nd **X**Talk) and the latter FEXT (**F**ar **E**nd **X**Talk). In addition, signals such as AM broadcast transmissions can be picked up by the subscriber line and cause narrow bands of frequencies to be unusable. DSL modems must be able to cope with all these types of interference.

Multicarrier modems were proposed, but not accepted, for V.fast. A multicarrier scheme called **Discrete MultiTone (DMT)** has become the recognized standard for G.lite and ADSL. These modems transmit a large number of independent equally spaced carriers, each with an nPSK or QAM constellation, and all with the same baud rate. This baud rate is very low compared to the bit rate, and so each carrier has a narrow bandwidth. These narrow bandwidth transmissions remind us of those used in the proof of the second half of Shannon's capacity theorem, and indeed the multicarrier approach is successful partly due to its ability to approach the water-pouring limit. Furthermore, we can assume that the channel attenuation and phase delay are approximately constant over the narrow bandwidth of these transmissions, hence equalization in the normal sense is not required. All that is needed is a single gain to compensate for the attenuation at the carrier frequency, and a single phase rotation to bring the constellation to the proper angle. This **F**requency **E**qualizer (FEQ) can be performed by a single complex multiplication per carrier. The coefficient can be found as in exercise 6.14.8.

The narrow bandwidth and slow baud rate make the ISI less important; however, if the carriers are close together we would expect **I**nter**C**hannel **I**nterference (ICI) to become a problem. ICI is removed in DMT by choosing the intercarrier spacing to be precisely the baud rate. In this fashion each carrier sits on the zeros of its neighbor's sincs, and the ICI is negligible. Multicarrier signals with this spacing are called **O**rt**H**ogonal **F**requency **D**ivision **M**ultiplexing (OFDM) signals, since the carriers are spaced to be orthogonal.

How do we demodulate DMT signals? The straightforward method would be to use a bank of band-pass filters to separate the carriers, and then downmix each to zero and slice. However, it is obvious that this bank of filters and downmixers can be performed in parallel by using a single FFT algorithm, making the DMT demodulator computationally efficient. Indeed, the modulator can work the same way; after dividing the bit stream into groups, we create complex constellation points for each of the constellations, and then perform a single iFFT to create the signal to be transmitted!

EXERCISES

- 18.21.1 What is the SNR needed to achieve 56 Kb/s using every other PCM level and assuming 3.8 KHz of bandwidth and that the consumer's modem has a 16-bit linear A/D? Is this reasonable? Why is it harder to transmit 56 Kb/s upstream?

- 18.21.2 Why is the POTS splitter implemented using passive analog filters rather than digital filters?
- 18.21.3 A DMT modem still has some ISI from previous symbols. This ISI is removed by using a cyclic prefix. Explain. The overhead of a long cyclic prefix can be reduced by using a **T**ime **E**qualizer, which is a filter whose sole purpose is to decrease the length of the channel impulse response. What is the connection between the TEQ, FEQ, and a regular equalizer?
- 18.21.4 DMT modems suffer from high **P**eak to **A**verage **R**atio (PAR). Explain why. Why is this an undesirable feature? What can be done to lower the PAR?

Bibliographical Notes

The general theory and practice of digital communications systems is covered in many texts [242, 95], and modems in particular are the subject of [144, 199]. [262] covers real-time DSP programming (using a floating point processor) for communications, including AM, FM, SSB, PAM, QAM, and echo cancellation for full-duplex modems.

Harry Nyquist published in 1928 a precursor to information theory [183]. Shannon's separation theorems appear in [237], which later appeared as a book. The first part of the channel capacity theorem first appears in [238], an article that very much deserves reading even today. The water-pouring criterion is due to Gallager, and appears in his book [67].

A good modern textbook on information theory is [46], while error correcting codes are covered in many books, e.g., [194]. This latter is an updated version of one of the first texts on the subject. A fascinating mathematically oriented book on topics relevant to error correcting codes is [42]. Reed and Solomon published their code in [218]. Viterbi presented his algorithm for decoding convolution codes in [265], but the classic overview is [63].

A dated, but still useful, reference on constellation design is [119]. Multidimensional constellations are covered in [70].

Timing recovery is reviewed in [64] and a suggested original article is [76].

TCM was first presented by Ungerboeck in [263], and Wei [270] discovered how to make it rotationally invariant, leading to the trellis code used in V.32. For TCM in multidimensional constellations consult [271].

Since standard texts go only as far as V.32, it is worthwhile consulting the V.34 review in [117]. Tomlinson and flexible precoding is explained in [118].

The classic, but dated, reference for multicarrier modulation is [18].

Readers interested in a nontechnical introduction to DSL modems should consult [82], while in-depth coverage is provided in [217, 36, 251].