Information Theory in Modem Practice

Paolo Prandoni

January 26, 2004

1 Introduction

The design of voiceband modems is an extremely intriguing area in which many related but different disciplines have converged to yield on of the most successful hi-tech consumer products. A first point of interest, for instance, is how previously existing communication techniques like Quadrature-Amplitude Modulation have been adapted to this particular problem. Another captivating aspect is represented by the remarkable improvements in data throughput which were made possible by the adoption of sophisticated information theoretical ideas. And, last but not least, the signal processing techniques which allow the entire machinery to run full-duplex, real time, and above all, *robustly* over a wide variety of telephone connections are simply fascinating.

In these notes we will try to look at the interesting interplay between information theory and electrical communication practice which takes place inside modern modems, and which is arguably the fundamental reason behind their impressive performance. We will first examine the features of the telephone channel, and review some intuitive ideas in data transmission. We will then deal in some more detail with three fundamental modulation techniques which are used in practice, namely PAM, PSK, and QAM. QAM, in particular, is the cornerstone of data transmission in modern modems, and we will thus study it in greater detail, taking into account the differences between possible constellations and constellation-shaping schemes. We will finally reach trellis coded modulation, the heart of modem design: this is a modulation technique in which error control codes and constellation design mix together to yield remarkable transmission gains. Experience has shown that it is always risky to speak of "conclusions" when it comes to modems; we will however try in the end to at least sum up what we have learnt. Enjoy!

2 Modem Design and the Telephone Channel

The history of modem design has been characterized by repeated and embarassingly wrong attempts to define the "ultimate" performance allowed for data transmission over the telephone channel. This upper limit has been again and again pushed forward by the newer modem standards defined over the years, and its current *de facto* value is set by the V.34bis specifications at a data rate of 33,600 bps (bits per second). It is interesting to note that in [1], a paper which does not fail to point out the hastiness of those who considered 4800 bps as an unsurpassable limit, this same limit is tentatively placed at 19,200 bps. All this was on the wake of the 14,400 bps V.32bis standard, roughly twelve years ago. What happened then in the meantime? For sure, the one thing which did *not* happen was the reaching of the Shannon limit (letting alone its breaking!): the achieved gains in throughput are indeed the outcome of more and more elaborate signal processing and coding techniques but, just as importantly, they are a consequence of the general improvement in the quality of the public switched telephone network (PSTN). When this improvement is taken into account, the Shannon limit will still look out of hand, as it "ought" to be (even though by not too much).

And, there is another very important point. Today's modems do not *guarantee* a given specific date rate; rather, they are defined by their manifacturers as *adaptive* systems which adjust their signaling capabilities to make the most of the particular channel they find themselves communicating through. A V.34 modem for instance possesses an extremely rich set of different signaling configurations which, among other things, enables the system to communicate reliably at data rates starting from 2400 bps up to 28,800 bps in increments of 2400 bps. If the particular connection during a data session happens to exhibit characteristics which are more typical of the telephone lines of the '60s, a V.34 modem will not do *so* much better than its older ancestors. But some better, it will. After all, we've got DSP's nowadays! This ongoing chase between modem designers and the properties of the telephone channel is summarized in figure 1; the diagram points out the major landmarks in modem design in the last 30 years or so, together with an approximate estimate of the Shannon Limit for the telephone channel of the time. We're getting closer and closer.

2.1 The physical telephone channel

The telephone channel can be quite accurately described as a low-noise, strictly bandlimited channel. It is also power limited, of course, otherwise we wouldn't be building modems, or complicated ones at least. The power limitation of the channel is however a relatively mild constraint, since a SNR of at least 28 dB can be assumed in all cases and one of 32-34 dB can be reasonably expected on a large percentage of individual connections. The main effects of power limitation are usually reflected in nonlinear distortion problems, so they will not be taken into account for the moment.

Most of the older textbooks will define the passband of the telephone channel as limited between 300 and 3000 Hz, for a total bandwidth of 2700 Hz. Things used to be so not because of the low capacity of copper wires (twisted pairs can be utilized to transmit data up to tens of megabits per second) but because the telephone companies used to

SNR	Bandwidth			
	2400	2800	3200	3429
28 dB	22176	25872	29568	31684
30 dB	23760	27720	31680	33947
32 dB	25344	29568	33792	36210

Table 1: Approximate capacity of the telephone channel.

introduce what are called *loading coils* in the line to compensate for the attenuation introduced by the capacitive effects of long wires. A side effect of these coils is to act as a lowpass filter with cutoff frequency around 4 kHz. Nowadays, most of the lines are PCM (Pulse Coded Modulation) digital loops, and the bandwidth limitations are imposed only by the antialiasing filters at the input of the necessary A/D converters, for a maximum bandwidth in excess of 3400 Hz. That's where all those new bits per second pass through. This also enables us to cast our own prediction on the ultimate capacity for data transmission over the PSTN: since current AD/DA codecs operate at 8000 Hz with 8 bits per sample, the maximum data throughput should be 64,000 bps. Makes sense, but better be careful.

2.2 Capacity of the telephone channel

Shannon's capacity formula for the bandlimited Gaussian channel with a Gaussian input signal is

$$C = W \log_2(1 + \frac{S}{N}) \tag{1}$$

where W is the channel's bandwidth, S is the power of the signal and N is the power of the Gaussian noise. Gaussianity of either the channel or the signal is not exactly met a condition in the case of data transfers over the telephone channel but, with this approximation, the above formula can be seen as an upper bound. The high SNR of the telephone channel allows us to simplify things a bit and we can rewrite the capacity formula as

$$C \approx 0.33 \times W \times SNR \tag{2}$$

with, as usual, $SNR = 10 \log_{10}(S/N)$. Table 1 presents several upper bounds on the capacity (in bits per second) for various available bandwidths and SNR's; note that 2400 Hz is the bandwidth assumed by the 14,400 bps V.32bis standard while 3429 Hz is the maximum bandwidth V.34 will attempt to use.

As a final remark, noting how close actual performance of today's modems is to the Shannon limit, one might ask how come we are so good at the task of transmitting data over telephone lines, which is not really the case for other types of communication channels. The answer is that, for one thing, the more bandlimited the channel, the lower the symbol rate for the signaling system, as we will see; this means that we

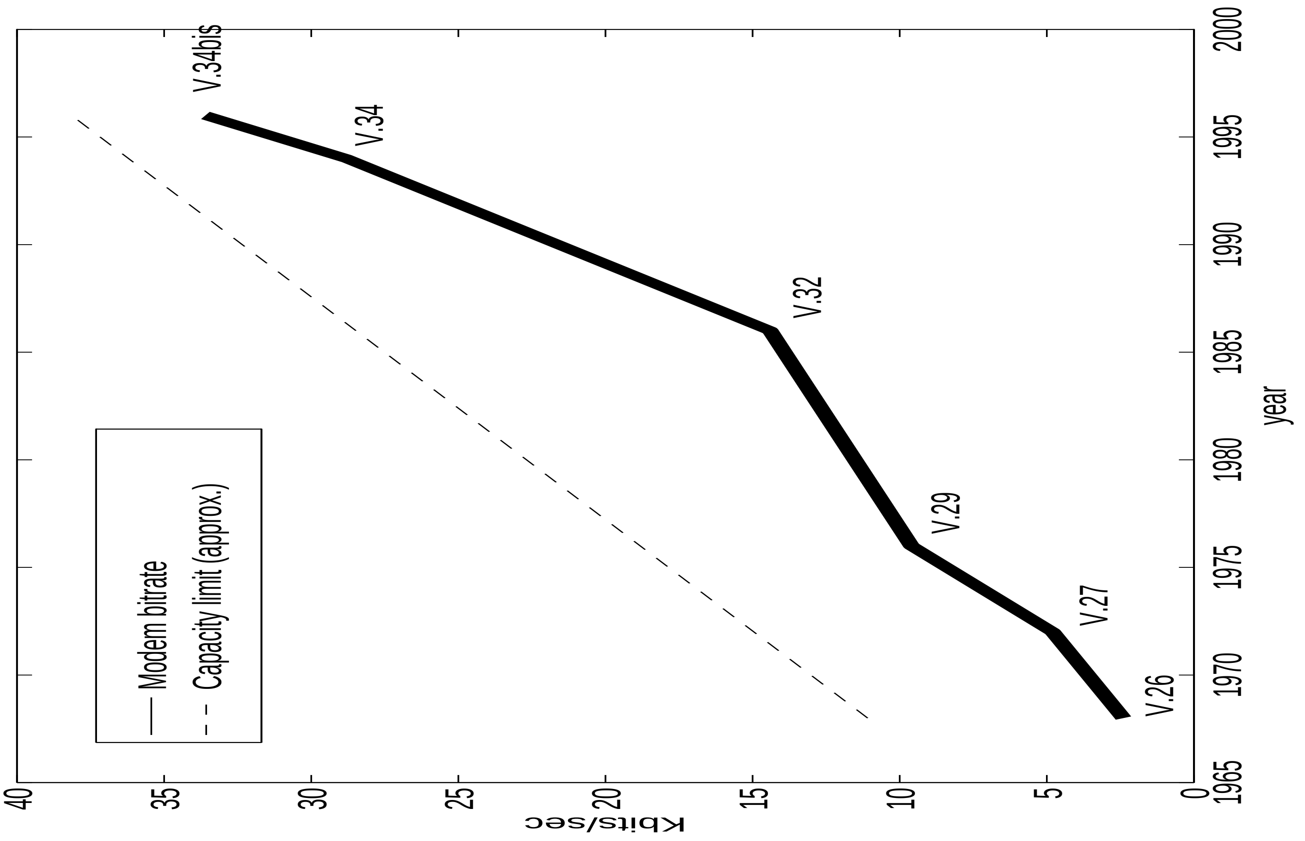

Figure 1: Evolution of modem capabilities.

have a lot of time to spend doing DSP on each incoming symbol, and so we can afford a lot of complicated techniques to achieve more coding gain. On the other hand, the telephone channel is really a very well-behaved thing; it can be assumed linear with good approximation, it practically *is* time invariant, and it has a relatively easily equalizable characteristic. In many ways, it is a dream channel.

3 Digital Modulation and its Tradeoffs: Some Informal Remarks

The fundamental problem in digital modulation can be stated as such: given a physical communication channel and a bit stream of, say, R bits per second, can we reliably transmit the bitstream over the channel, and how ? There is no unique answer to these questions, and possible solutions rely on the study of the transmission channel's features, of the tradeoffs between different signaling schemes, and of possible coding techniques; this analysis encompasses a whole range of topics drawn from communication theory to information theory the description of which is - fortunately! - outside the scope of these notes. There are however a few general guidelines which can be used to develop an intuitive feeling for the problem.

3.1 Hackers vs. Shannon

On one hand, information theory can help us frame the problem in a theoretical setting. Once we know the channel's bandwidth W and expected signal to noise ratio S/N, we also know that the channel's capacity C (for a Gaussian channel - but let us assume the channel *is* Gaussian) is given by equation (1). Now, this tells us that for $R < C$ there indeed exists a method to transmit R arbitrarily well; it does not tell us, however, what the method is or how to find it – but we can suspect that, as usual, the thing will involve enormous blocksizes, delays, and unrealistic complexity.

On the other hand, for the "practical" engineer, the problem has a very physical and instantaneous flavor: we've got this channel, and we want to close an electrical circuit around it to transmit signals; the trasmission methods which are sought do not take the data into account. Historically, the design of these circuits has been the task of electrical communication engineers and, following a patently wrong separation principle, their work has often been quite independent from that of their information theory colleagues. The result has been a series of "recipes" for communication with names such as PSK, DPSK, OOK, QAM, FSK, and so on, each with its own "efficiency figure", which is the expected probability of error at the receiver given bandwidth and allowed power; this is called *uncoded modulation.* System designers would use one of this methods and tag error control schemes to the data in an independent fashion.

Fortunately enough, the two perspectives have been brought together in the case of modem design with a technique called *trellis coded modulation.* It is thanks to this that the latter part in the curve of figure 1 is steeper. But we will see this in more detail later. For the time being, however, it is useful to look more closely at the communication engineer's perspective because, for one thing, in this case what is important is the practicality of the scheme and without that no working modem would have been possible.

3.2 Uncoded transmission

In what follows, R as we said is the raw data stream, expressed in bits per second. Transmission efficiency over a bandlimited channel will however require some type of *multilevel* signaling, in which the transmitter will send out signals corresponding to groups of m bits, referred to as *symbols*. Symbols belong to a finite alphabet of size 2^m which, just to give the picture, we can for the moment consider as signal amplitude values (the decimal value of the m-bit groups in Volts, for instance). Other meaningful choices for the alphabet are sets of different phases, phase-amplitude combinations, orthogonal waveforms and so on; each choice of the symbol alphabet defines a different signaling scheme. The duration of each symbol is $T = m/R$, and $1/T$ is called the *baud rate* of the system. In general the transmission can be assumed to be passband, so that the bitstream is going to be sent as a sinusoidal carrier modulated (in amplitude in our example) by the symbols in the alphabet. The positive frequency spectrum of the transmitted signal is going to be centered around the carrier frequency and its bandwidth W is going to be roughly equivalent to the "frequency" at which the carrier is modulated by the stream of symbols, that is:

$$W \approx \frac{1}{T}$$

(remember the sampling theorem). Of course this is not an accurate description because we know that, independently of the chosen modulation scheme, a finite duration for the symbol time T entails an infinite bandwidth for the modulated signal (remember the Fourier pair rect $\leftrightarrow$ sinc); by appropriately filtering the signal, however, the tradeoff between T and W can be adjusted so that the above relation is practically valid, as we will see later. With a somewhat crude approximation, the energy of the generated signal is proportional to the size of the symbol alphabet. Therefore, in the presence of a maximum power limit on the physical channel, if we use a large alphabet we will have to "scale down" the actual transmitted signal; as a consequence, the symbols at the receiving end will appear more "packed" together, or in other words, their relative distance (in some metric) will decrease. This in turn means that in the presence of noise, it will be more difficult for the receiver to identify the symbols correctly and the symbol error rate will increase.

This is in a nutshell the fundamental tradeoff in practical uncoded modulation schemes, which links reliability, bitrate, bandwidth, and power. Suppose we want to achieve a given reliability level in transmission: if we want to transmit a fixed given number of bits per second we can reduce the bandwidth by increasing T, but then we will have to increase the alphabet size and, to maintain the same error rate, the power level as well; conversely, if we want to reduce the power level we will have to reduce the alphabet size, which means transmitting more symbols per second or, equivalently, using a larger bandwidth. Another tradeoff strategy is to adjust the bitrate to the given constraints on bandwidth and power, which is what modems do. Reliability is hardly ever traded for, of course. There is no way out this impasse if we are not willing to take a step further and move on to coded modulation, which means employing some information theoretical tools and try to comply with the guidelines laid out by Shannon's capacity theorem.

In the end, the hacker's recipe for data transmission over the telephone channel (which is nothing but the philosophy built in V.34) is the following: choose the baud rate as to maximally fill the available bandwidth ($T \approx 1/W$), signal at the maximum power level allowed, and see how big an alphabet can be used while keeping a sufficiently low error rate. Then step aside and let Information Theory do the necessary improvements to bring us close to capacity (and take all the merit ... oh, but that's another story).

4 Introduction to Digital Modulation Techniques

As we have already hinted at, we are in the need of transmitting digital information over an arbitrary portion of the frequency spectrum. This is accomplished by representing the transmitted information as the *modulation* (that is, the modification over time) of a sinusoidal carrier; the frequency of the carrier determines the center frequency of the spectrum of the transmitted signal. The modulating signal, which represents the data, is usually referred to as the *baseband* signal and the modulated waveform as the *passband* signal; since modulation is in practice just a frequency translation of the baseband signal, the only requirement for the carrier frequency f_0 is $f_0 \geq W/2$, where W is the bandwidth of the baseband signal (see figure 2). Conceptually, a passband signal is any signal which can be thought of as the output of a modulation process. In radio transmission and in most other cases $f_0 \gg W/2$, but there is no requirement for the passband signal to occupy any particular region of the spectrum: modem signals, for instance, are well within the acoustic range, while being technically passband.

We will now briefly examine some techniques to actually attach the digital data to a sinusoidal carrier.

4.1 Pulse Amplitude Modulation (PAM)

Pulse Amplitude Modulation is one of the most basic signaling techniques, and we have already introduced it in its essence in section 3. It is simply the digital equivalent of old-time radio AM; at each symbol interval, m bits from the data stream are mapped onto one of 2^m scalars taken from the set

$$A = \{\pm 1, \pm 3, \pm 5, \ldots, \pm(2^m - 1)\}; \tag{3}$$

the selected value determines the local amplitude of the transmitted carrier. The given set of points is chosen as such because it forms a *balanced* set of coordinates; balanced means that its mean value is zero and that it is symmetric around the zero amplitude value.

If we want to express the transmitted waveform in formulas we run into a little awkwardness because we are basically mixing the digital domain (the sequence of symbols) with the continuous time domain (the carrier). The symbol sequence can be written independently of the modulation technique as a discrete time series $a[k]$, where $a[k] \in A$ is the k-th symbol in the data stream; in the remainder we will consider this as a random

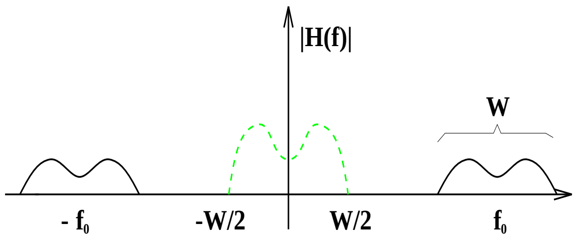

Figure 2: Baseband and passband signals.

sequence. A good representation for the transmitted signal signal can be obtained in the following way: let's consider a data rate of $1/T$ symbols per second and let's start by expressing the symbol stream as a weighted sum of Dirac pulses:

$$a(t) = G_0 \sum_k a[k]\delta(t - kT);$$

where G_0 is a gain factor which determines the relative "spacing" between modulation levels; this representation clearly shows how the symbols are spaced T seconds apart (see figure 3(a)). We can think of this as of a system which shoots out a symbol value (for an infinitesimal time) every T seconds; now we have to tag these values in some way to the sinusoidal carrier. The first, simplest idea is to set the amplitude of the carrier equal to $a[k]$ for the whole duration of the k-th symbol interval. The Dirac pulses representation introduces neatly the idea that this can be achieved by filtering the signal $a(t)$: a copy of the filter's impulse response $s(t)$ is placed at each impluse location, scaled by the value of the corresponding symbol. If the impulse response is $s(t) = \text{rect}(t/T)$[1] , we obtain the filtered signal of figure 3(b), which is what we were looking for. The filtered pulse stream has the form

$$b(t) = G_0 \sum_k a[k]s(t - kT) \tag{4}$$

and modulation is achieved multiplying this signal with a sinusoidal carrier at frequency f_0, yielding the transmitted signal

$$c(t) = b(t)\cos(2\pi f_0 t + \theta_0);$$

θ_0 accounts for possible phase offsets which can be introduced both at the transmitter and over the channel. Ok, now the troubles begin. The "attentive reader" will have already frowned at the waveform of figure 3(b), which is patently discontinuous[2]. Indeed,

[1](One never remembers this quite exactly:) the rect function is defined as

$$\text{rect}(x) = \begin{cases} 1 & \text{if } |x| \leq 1/2 \\ 0 & \text{if } |x| > 1/2 \end{cases}$$

The sinc function is defined as

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x};$$

note that $\text{rect}(x/T)$ has a support of width T and that $\text{sinc}(x/T) = 0$ for $x = \pm T, \pm 2T, \pm 3T, \ldots$. The functions $\text{sinc}(\cdot)$ and $\text{rect}(\cdot)$ are related through the Fourier transform as

$$\begin{aligned} \mathcal{F}[\text{rect}(x/T)](f) &= T\text{sinc}(Tf) \\ \mathcal{F}[\text{sinc}(x/T)](f) &= 2\pi T\text{rect}(Tf) \end{aligned}$$

where

$$\mathcal{F}[s(t)](f) = \int_{-\infty}^{+\infty} s(t)e^{-2\pi ft}$$

[2]Of course such a waveform cannot be realized in practice, but it can be approximated arbitrarily well by very fast (and expensive) circuitry. This would be a waste of money, as we are about to show.

this proves to be a problem, and to show that we only have to take a look at the spectrum of the transmitted signal. Now, $c(t)$ is a random process, therefore some theoretical justifications are needed before we can apply the usual tools of spectral analysis. For a rigorous derivation, the reader is referred to [2, pag. 65] or to [3, pag. 191]; here let it suffice to say that with the approximation that the $a[k]$'s form a sequence of independent, uniformly distributed random variables, the power spectrum of $c(t)$ is

$$P_c(f) = G_0^2 \frac{S_A}{2T} \left[|S(f - f_0)|^2 + |S(-f - f_0)|^2\right] \tag{5}$$

where $S(f)$ is the (continuous time) Fourier transform of $s(t)$ and S_A is an energy factor which, for the set A defined above, is $S_A = (4^m - 1)/3$ (more about this energy factor later). If $s(t) = \mathrm{rect}(t/T)$, we obtain

$$P_c(f) = G_0^2 \frac{S_A T}{2} \left[\mathrm{sinc}^2(T(f - f_0)) + \mathrm{sinc}^2(T(-f - f_0))\right]$$

which indicates where the problem lies with using rectangular pulses for signaling: the frequency support of the waveform is infinite (which we would expect from a time-limited $s(t)$) but, more seriously, since the sinc function decays very slowly (as $1/f$), a large bandwidth is required to contain most of its energy. Ideally, we would like to have a power spectrum for the transmitted signal strictly confined to the finite bandwidth W which the bandlimited channel offers. Practically, we do not really mind if there's a little bit of out-of-band spillage, provided it is sufficiently small so that almost all of the waveform's energy is in-band; but the $1/f$ decay of the sinc waveform means that we have to have *at least* $W = 2/T$ to have 90% of the signal's energy in band. Practically speaking, this means that for highly bandlimited channels the baud rate should be kept very low.

We could take the reverse approach however, and look for a filter $s(t)$ whose Fourier transform is bandlimited. With the channel having a bandwidth of W Hz, the goal could be achieved by taking $T = 1/W$ and $s(t) = \mathrm{sinc}(t/T)$, so that $S(f) = (2\pi/W)\mathrm{rect}(f/W)$ and in this case 100% of the energy would be in-band. The downside this time is that the corresponding impulse response has an infinite duration in time. We can rewrite (4) as

$$b(t) = G_0 \sum_k a[k]\mathrm{sinc}(t/T - k)$$

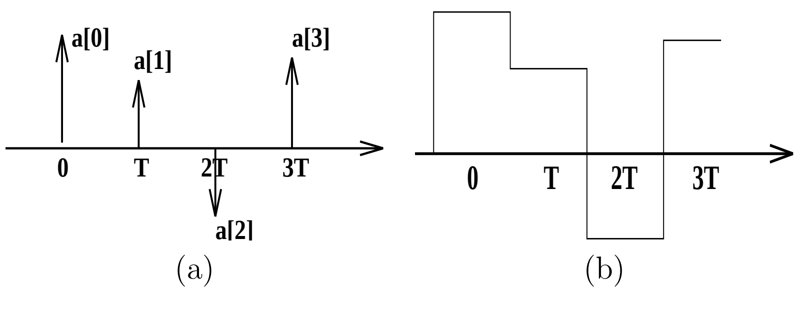

Figure 3: Pulse sequence and rect modulation.

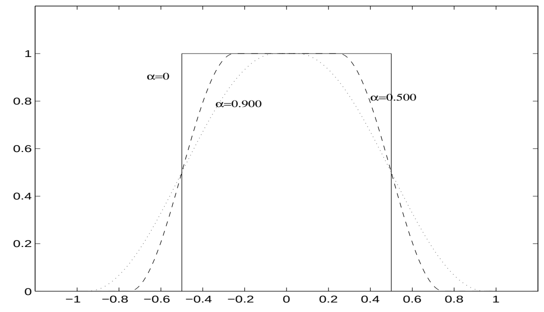

Figure 4: Raised cosine pulses: spectral characteristics.

which is also the waveform at the receiver after demodulation (minus a possible phase offset term introduced by the whole process). The good news is that if we could sample this signal at *exactly* $t = kT$ for all k we would get the $a[k]$'s back exactly: pulses with this property are called *Nyquist pulses.* The bad news, however, is that it's really impossible to obtain and keep such a level of synchronization at ther receiver and, as a consequence, each sample of the received, demodulated waveform will contain "traces" of all the other symbols in the form of the sampled sum of the "tails" of the surrounding pulses; this is called *intersymbol interference* (ISI). The amount of ISI in the case of a small jitter at the sampling instants depends on how fast the tails of the pulse approach zero, of course. The bandwith-greedy rect pulse we have seen before has always zero ISI; the spectrally efficient sinc pulse however, with its $1/t$ decay rate, is a very bad choice. Tradeoffs, tradeoffs, tradeoffs.

We have thus seen the two possible extremes; are there intermediate possibilities amongst the family of Nyquist pulses which can efficiently trade spectral occupancy for intersymbol interference ? Fortunately yes, and one of the most popular is called the *raised cosine pulse.* The main charm of this pulse is that it has a "knob" in the form of a parameter $0 \leq \alpha \leq 1$ with which to operate this tradeoff; in practical systems this knob is set such that the spectral occupancy of the waveform is only 10% greater than the minimum value $W = 1/T$ corresponding to a rect pulse. For an analytical formulation of the raised cosine pulse, the reader is referred again to [2, pag. 32]; in figure 4 some of the obtainable spectra are displayed. The decay rate of the corresponding pulses in the time domain is $1/t^3$ for $\alpha > 0$; this means that in actual implementations the theoretical infinite duration of the pulse can be well approximated by a limited number of samples.[3]

4.2 Phase Shift Keying (PSK)

Phase Shift Keying is another modulation technique in which the phase of the carrier, rather than its amplitude, is changed over time to transmit information. Now the

[3]To understand in full the implications of a poorly designed pulse shaping function, it is enough to consider the famous example of the first transoceanic cables for telegraphy which, at their best, offered a throughput of *two* words per minute; the interested reader is encouragingly referred to the intriguing chapters in [4, pagg. 332-337].

symbols are drawn from a set of the form

$$A = \{k\frac{2\pi}{2^m},\ k = 0, \ldots, 2^m - 1\};$$

the transmitted waveform can be expressed as

$$c(t) = \cos(2\pi f_0 + \theta(t))$$

where $\theta(t)$ is the time-varying phase determined by the symbol stream; as an example think of $\theta(t) = a[k]$ for $(k-1)T < t \leq kT$, $a[k] \in A$. We will not go into details as far as PSK is concerned, but basically all the bandwidth-ISI tradeoffs which we saw in relation to PAM apply here just as well. An interesting derivation which not only shows that, but which will also lead us nicely into the next modulation scheme is the following. The transmitted signal can be expanded as

$$\begin{aligned} c(t) &= \cos(\theta(t))\cos(2\pi f_0 t) - \sin(\theta(t))\sin(2\pi f_0 t) \\ &= I(t)\cos(2\pi f_0 t) - Q(t)\sin(2\pi f_0 t) \qquad (6) \end{aligned}$$

from which we see that PSK can be seen as a special kind of PAM, in which two carriers 90 degrees apart are modulated in amplitude by the functions $I(t)$ and $Q(t)$, which now have the same role of $b(t)$ in equation (4). The frequency occupation of the signal is determined only by the power spectrums of $I(t)$ and $Q(t)$, which occupy the same bandwidth. A point worth noting is that now, at each instant t, we have:

$$I^2(t) + Q^2(t) = 1;$$

after some reflection one can see that by eliminating this constraint, twice as much information could be sent over the same bandwidth at the same baud rate in the form of two independent I and Q streams[4]. How to do this is the topic of the next section.

4.3 Quadrature Amplitude Modulation (QAM)

Quadrature Amplitude Modulation is a modulation technique which combines PAM and PSK, in the sense that the sinusoidal carrier is now modulated both in amplitude and in phase. There are several ways one could approach QAM, but the most practical one is what we already saw in equation (6), that is, considering the QAM signal as a couple of orthogonal carriers independently modulated in amplitude. This also reflects the most efficient way to demodulate a QAM signal at the receiver, which consists in splitting the received signal into two orthogonal signals and demodulating them as independent PAM waveforms. The two PAM signals are called the *in-phase* (I) and *quadrature* (Q) components. From this point of view, modulation could proceed just as in the case of PAM: we choose an alphabet set for the I and the Q carriers, proceed to build the modulating waveform as in (4), multiply one of them by $\cos(2\pi f_0 t)$ and the other by $\sin(2\pi f_0 t)$, and sum them together to obtain the transmitted signal. The number of bits per symbol is just the sum of the bits per symbol in the I and the Q signals.

[4]Ok, this could be achieved also by doubling the alphabet size of PAM or PSK. But going this way offers advantages which will be apparent in the next sections

Complete independence of the I and Q alphabets is not desirable, however. To stress the fact that each QAM symbol is a truly two-dimensional entity, we are better off by switching to complex notation. This is just a notational convention, of course, but it simplifies things a lot and it highlights some very important properties of QAM signals.

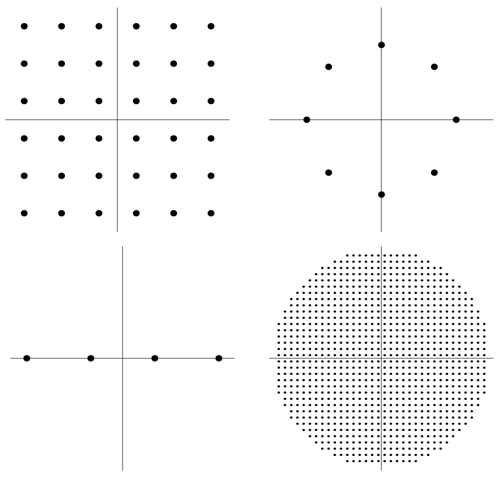

Figure 5: Examples of QAM constellations.

If we want to transmit m bits per QAM symbol ($m \geq 2$) we will define the alphabet set as a set of complex points of cardinality 2^m; this set can be represented on the complex plane as a *constellation*, and some examples are shown in figure 4.3. The first constellation corresponds to the case of two identical alphabet sets for the I and Q signals, each of them defined as in equation (3), but of half the size; the QAM alphabet set is

$$A = \{a + jb;\ a, b \in C\} \tag{7}$$

with $C = \{\pm 1,\ \pm 3,\ \pm 5,\ \ldots,\ \pm(2^{m/2} - 1)\}$; this type of constellation leads to what is called "narrow-sense" QAM. The second example uses the alphabet

$$A = \{e^{jk\frac{2\pi}{2^m}},\ k = 0, \ldots, (2^m - 1)\}$$

while the third uses

$$A = \{\pm 1,\ \pm 3,\ \pm 5,\ \ldots,\ \pm(2^m - 1)\};$$

these two examples correspond to PSK and PAM respectively, which shows how QAM is an extension of the two. The last example shows a type of non-separable circular constellation (as used by V.34 for instance) which cannot be expressed as the cartesian product of two independent symbol streams.

With this notation at hand, we can write the equations for the modulating signal in the usual way:

$$a(t) = G_0 \sum_k a[k]\delta(t - kT),$$

where now the $a[k]$'s are complex, and

$$b(t) = G_0 \sum_k a[k]s(t - kT)$$

with the usual properties for the pulse $s(t)$. The transmitted waveform can be expressed as

$$c(t) = \mathrm{Re}[b(t)e^{j(2\pi f_0 t + \theta_0)}]$$

which is nice since it brings us back to the fact that, after our complex- valued machinery, all we really transmit is a real signal as it ought to be. The reader is encouraged to compare the previous expression to (6), and to find the expressions for $I(t)$ and $Q(t)$ in this case. It is also easy to verify that the power spectrum of the QAM signal is expressed by equation (5): the frequency occupation is the same, and depends only on the pulse $s(t)$, while the type of constellation affects only the energy parameter S_A. This last fact will be extensively studied in the following sections.

4.4 Demodulation

In these short notes, we will not go into the details of all the demodulation techniques for the signaling schemes we have introduced. As a general remark, however, a demodulator is a system which tries to strip the incoming waveform of all the features which were introduced at the source to facilitate transmission (the carrier, the pulse shape, and so on) to obtain a local version of the original time series $a[k]$. In an ideal demodulation system, with perfect equalization and perfect synchronization, all the side effects of transmission like ISI and channel-induced distortion will be eliminated. There is one thing however which the demodulator cannot get rid of, and that is the ubiquitous noise introduced by the channel; as a consequence the "regenerated" time series will have the form

$$\hat{a}[k] = G_0 a[k] + n[k] \tag{8}$$

where $n[k]$ is a sampled random process (the noise samples) whose statistics are dependent on the particular characteristics of the channel. The simplest technique to recover $a[k]$ from $\hat{a}[k]$ is to associate to each $\hat{a}[k]$ the alphabet symbol scaled by G_0 closest in the Euclidean metric (*hard decoding*). An error will occur if the value of $n[k]$ is greater than half the distance between $a[k]$ and its closest neighbor. The greater this distance, the lower the probability of error; of course this distance can be augmented by increasing the gain factor G_0, and therefore the average power of the transmitted signal, but only up to the power limit imposed by the channel.

We can substantiate these arguments in the case of ideal QAM transmission over the bandlimited Gaussian channel. From an operational point of view this channel can be seen as an ideal brickwall filter $H(f)$ of bandwidth W, plus additive white noise (see fig. 6). The noise power is constant over the channel's bandwidth and equal to $N_0/2$;

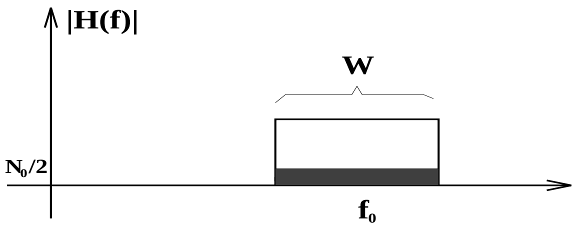

Figure 6: Representation of the Gaussian bandlimited channel.

the total noise power N is obtained by integrating the corresponding two-sided power spectrum

$$N = \frac{N_0}{2}\int_{-\infty}^{+\infty} |H(f)|^2 = WN_0.$$

Just as we represented the QAM signal as two orthogonal signals, we can split the additive gaussian noise in two orthogonal components, affecting the I and Q signals independently. This can be neatly expressed at the demodulator in the same form as in equation (8), where in the present case $n[k] = n_I[k] + jn_Q[k]$ is a sequence of *complex* iid random Gaussian variables of zero mean and of variance $\sigma^2 = N_0/2$ in each component. If the symbol alphabet for the transmitted QAM signal is drawn from a square lattice such as the one described by (7) and regardless of the shape of the constellation's boundary, the probability of correctly decoding a symbol is equal to the probability that the corresponding complex noise sample lies in a square of side $2G_0$ centered around the transmitted point; for computational ease, this probability can be approximated well enough by integrating the noise probability density function (pdf) over a circle C of radius G_0 (see figure 7). The pdf for the noise samples is that of a two-dimensional Gaussian distribution

$$p_n(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

so that the approximate error probabilty is

$$p_e = 1 - \iint_C p_n(x, y)\, dx\, dy = e^{-\frac{G_0^2}{N_0}} \tag{9}$$

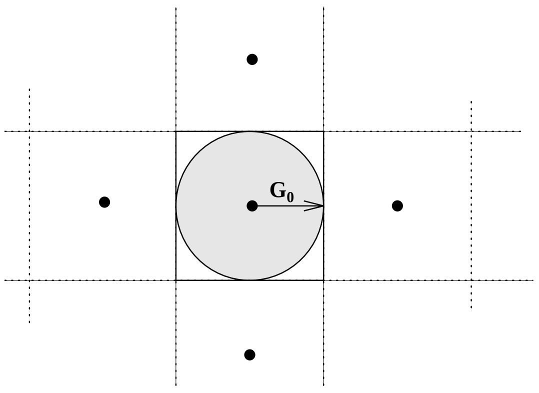

Figure 7: Correct decision region at the demodulator: optimal region in the nearest-neighbor sense (the square) and its circular approximation.

5 QAM Revisited

Let us sum up the situation so far; we have a bandlimited Gaussian channel (bandwidth W) with a power constraint which limits the maximum SNR to $10\log_{10}(S/N)$ dB, and we decided to use QAM modulation on it with baud rate $1/T = W$ and maximum allowed power. As we have seen, Shannon's capacity formula for this channel yieds

$$C = W\log_2(1+\frac{S}{N}) \approx W\log_2(\frac{S}{N})$$

for high SNR channels; this means that there exists a signaling method which allows us to transmit C bits per second with arbitrarily low probability of error. We want to see how far from this limit we are using our simple QAM modulation scheme.

The average power of the QAM signal can be obtained by integrating the power spectrum of equation (5); if we assume that the the shaping pulse $s(t)$ has been designed so as to have unit energy, the signal power is

$$S = G_0^2\frac{S_A}{T}.$$

The "energy factor" S_A depends on the type of constellation and on its size; it can be expressed as the average power of a discrete probability distribution defined over the alphabet set A and associating a "probability of usage" p_k to each of the symbols:

$$S_A = \mathrm{E}[|a_k|^2] = \sum_k |a_k|^2 p_k.$$

In the case of equiprobable points, as in uncoded modulation, $p_k = 1/2^m$ (m bits/symbol); the square constellation of equation (7) has for example an average power of (exploiting symmetries)

$$S_A = \frac{4}{2^m}\sum_{i=1}^{2^{m/2-1}}\sum_{j=1}^{2^{m/2-1}}[(2i-1)^2+(2j-1)^2] = \frac{2(2^m-1)}{3}.$$

Given a desired probability of error p_e, equation (9) yields

$$G_0^2 = -N_0\ln p_e;$$

substituting for these values, we can express the SNR of the QAM signal as

$$\begin{aligned} \frac{S}{N} &= (-N_0\ln p_e)\frac{S_A}{T}\frac{1}{WN_0} && (10)\\ &= -\frac{2\ln(p_e)}{3}(2^m-1) && (11)\\ &\approx \kappa 2^m \end{aligned}$$

so that the maximum number of bits per symbol we can reliably transmit is

$$m = \log_2(\kappa^{-1}\frac{S}{N}).$$

The "effective" capacity obtained by uncoded QAM is thus

$$C' = W \log_2(\kappa^{-1} \frac{S}{N}) \quad \text{bits/sec}$$

which, for a practical probability of error $p_e \approx 10^{-6}$, corresponds approximately to that of a channel with a SNR smaller by 9 dB than what we actually have ($\kappa = -(2/3)\ln(10^{-6}) \approx 9.21 \approx 9.6$ dB). This means that, by using some smart coding techniques, there are more than 9 dB's worth of room for possible improvement – that's where information theorists come into play.

5.1 Shape gains

Let's switch perspective for a second: we have an estimate of how many bits per symbol we can transmit, we know what the maximum SNR is, and we want to minimize the probability of error within these limits. We have seen in (11) that

$$\frac{S}{N} = G_0^2 \frac{S_A}{T} \frac{1}{WN_0} = G_0^2 \frac{S_A}{N_0}.$$

Consider the two constellations shown in figure 8: it is immediate to show that their energy factors are $S_A^2 = 6G_0^2$ and $S_A^2 = 4.73G_0^2$ respectively (in the second constellation the points on the axis are at a distance of $1 + \sqrt{3}$ from the origin). This means that, for a given SNR, we can have a bigger scaling factor G_0 if we use the second arrangement of points, which in turn reduces p_e. In other words, with respect to the first one, the more efficient constellation makes the SNR appear higher by a factor $(6/4.73) \approx 1$ dB; this kind of gain is called *shape gain*. Unfortunately, in the second constellation the points are not drawn from a uniform grid, which complicates decoding and makes this type of arrangement impractical and unused.

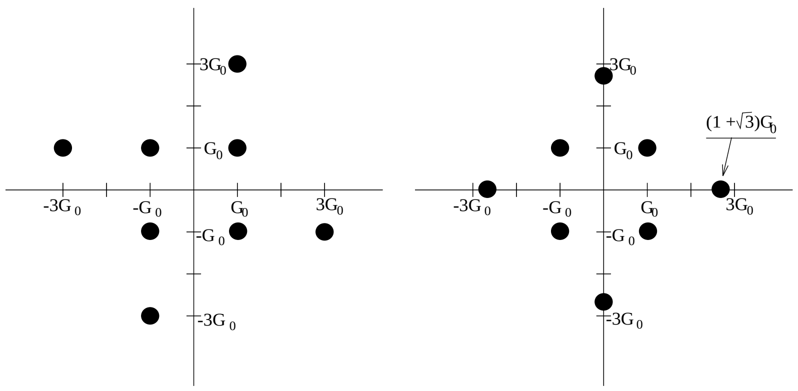

Figure 8: Two 8-point constellations.

During the early times of QAM, a lot of researchers set out to find the "optimal" constellations in terms of shape gain; for small orders this task can be carried out by

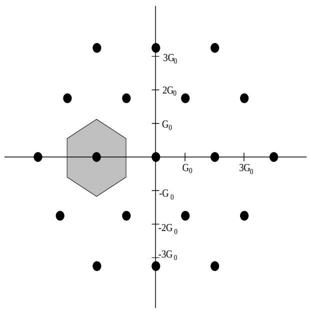

Figure 9: Example of constellation based on a hexagonal lattice; the grey shaded area is the corresponding Voronoi region

exhaustive search, and it can yield some worthwhile results, as we have just seen. As the number of point increases, however, the possible shape gains become smaller and smaller; this can be shown invoking the so called *continuous approximation* [5]. For a large constellation, the number of points n is well approximated by the ratio of the area of the constellation (which depends only on its boundary) and the area of the *Voronoi region* for the points (the Voronoi region is defined as the set of points in the plane closer to a given constellation point than to any other constellation point - it is basically the "correct decision" square shown in figure 7). Furthermore, the energy factor of the constellation is also well approximated by the average energy of a uniformly distributed probability function defined over B, the area of the constellation:

$$S_A = \frac{1}{2B} \iint_B (x^2 + y^2)\, dx\, dy \tag{12}$$

Since the circle is the shape which has the least such average energy amongst all shapes with the same area, a circular boundary is the best possible boundary for a constellation. It is left as an exercise to prove that the shape gain of a circular constellation over an equivalent square constellation is approximately 0.2 dB. V.34 can use a wealth of different constellations, corresponding to different possible bitrates, and they are all subsets of a 960-point circular constellation drawn from the odd-indexed integer lattice.

As a last note, suppose we arrange the constellation points on a hexagonal grid while keeping their distance (and thence p_e) the same; the area of the corresponding Voronoi region is going to be scaled by 0.866 (see figure 9. This in turn means that a smaller area will be required to enclose a given number of constellation points, with all the relative shape gain benefits: 0.866 is approximately 0.6 dB after all. Unfortunately, the hexagonal arrangements exceedingly complicates the demodulation process, and we will see that there are better ways to get this gain back (even though some modems in the past did use such constellations); trellis coding together with set partioning, which will be introduced in Section 6, are indeed a way to indirectly change the constellation's underlying lattice. When these coding techniques are employed, the problem of populating the constellation space with signal points can be efficiently addressed from a pure channel coding perspective while preserving the simplicity of the physical modulation/demodulation processes.

5.2 Shape Gains for Multidimensional Constellations

Narrow sense QAM (in which the 2-D constellation is separable) is no more efficient than equivalent-order PAM[5]. The first advantages of QAM signaling come when we consider the signal as truly two-dimensional and nonseparable, in which case, as we have just seen, a maximum gain of 0.2 dB can be obtained from circular constellations (plus another 0.6 dB if we are willing to go for hexagonal lattices). We can push this strategy further up to higher-dimensional constellations, with diminishing returns and with an asymptotic limit for the gain of an hypersphere over the corresponding hypercube of approximately 1.53 dB. Using the continuous approximation, in N dimentions equation (12) generalizes to

$$S = \frac{1}{N\,V(C)} \int_C \| \mathbf{x} \|^2 \, d\mathbf{x} \tag{13}$$

where $\| \mathbf{x} \|^2 = (x_1^2 + \ldots + x_N^2)$, C is the region bounding the multidimensional constellation and $V(C)$ is the volume of this region. For a hypercube C_c of side L

$$\int_{-L/2}^{L/2} \cdots \int_{-L/2}^{L/2} (x_1^2 + \ldots + x_N^2)\, dx_1 \ldots dx_N = N \frac{L^{N+2}}{12}$$

so that

$$S_c = \frac{L^2}{12}.$$

For a hyperspere C_s of radius R, the integral calculation is better done in polar coordinates[6]; the Jacobian for the coordinate change in N dimensions is

$$J = \rho^{N-1} \prod_{i=2}^{N} \sin^{N-i} \theta_{N-i+1} = \rho^{N-1} \Theta(\theta_2, \ldots, \theta_{N-1})$$

so that

$$\int_{C_s} \| \mathbf{x} \|^2 \, d\mathbf{x} =$$
$$\int_0^{2\pi} \int_0^{\pi} \cdots \int_0^{\pi} \Theta(\theta_2, \ldots, \theta_{N-1}) d\theta_1 \, d\theta_2, \ldots, \theta_{N-1} \int_0^R \rho^{N+1} \, d\rho$$

and

$$V(C_s) = \int_{C_s} d\mathbf{x} =$$
$$\int_0^{2\pi} \int_0^{\pi} \cdots \int_0^{\pi} \Theta(\theta_2, \ldots, \theta_{N-1}) d\theta_1 \, d\theta_2, \ldots, \theta_{N-1} \int_0^R \rho^{N-1} \, d\rho$$

[5]Narrow-sense QAM at 2^m bits per symbols can be carried by 2^m-level PAM; the corresponding signal, if transmitted using single-sideband modulation techniques [3], occupies the very same bandwidth as the original QAM signal and offers the same p_e.

[6]A N-dimensional point is represented by the N-tuple $(\rho, \theta_1, \ldots, \theta_{N-1})$, where $\rho \in [0, +\infty)$, $\theta_1 \in [0, 2\pi)$, and $\theta_2, \ldots, \theta_{N-1} \in [0, \pi)$.

so that (13) simplifies to

$$S_s = \frac{\int_0^R \rho^{N+1}\, d\rho}{N \int_0^R \rho^{N-1}\, d\rho} = \frac{R^2}{N+2}. \tag{14}$$

At this point we want to compare the two values for the energy factor in the case of two constellations which contain the same number of points, and we have seen that the continuous approximation tells us that the number of points in a constellation is given by the ratio between its volume and the volume of the Voronoi region for the undelying lattice. The lattice for both the hypercube and the hypersphere is the same, so we just need their volumes to be equal. For the hypercube,

$$V(C_c) = L^N$$

while for the hypersphere we can use the "well-known" formula

$$V(C_s) = \frac{R^N \pi^{N/2}}{\Gamma(N/2+1)};$$

by setting $V(C_c) = V(C_s)$ we obtain R in terms of L, and by substituting in (14) we have

$$\frac{S_c}{S_s} = \frac{\pi}{12} \frac{N+2}{(\Gamma(N/2+1))^{2/N}};$$

Finally, Stirling's approximation for the Gamma function is $\Gamma(N/2+1) \approx (N/2e)^{N/2}$ for large N, and in the end

$$\lim_{N\to\infty} \frac{S_c}{S_s} = \frac{\pi e}{6} \approx 1.53\text{dB}.$$

Now, after all this work, one still has to notice that while the two-dimensional representation of QAM signals is pretty intuitive and has a well defined physical counterpart in the way we modulate the signal, higher- dimensional constellations possess no such ease of representation; in practice they are dealt with by transmitting the coordinates of each multidimensional point in pairs, as a sequence of 2-D QAM points. This prompts the following observation: consider (for simplicity) a two-dimensional uniform probability distribution defined over the unit circle C; its density is going to be

$$p_u(x,y) = \begin{cases} \frac{1}{\pi} & \text{if } (x,y) \in C \\ 0 & \text{if } (x,y) \notin C \end{cases}$$

The marginal probabilities defined over the axis are however not uniform:

$$p_u(x) = \begin{cases} \frac{2\sqrt{1-x^2}}{\pi} & \text{if } |x| \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

This shows that the net effect of using a circular QAM constellation is to induce a non-uniform probability distribution on the underlying one-dimensional I and Q signals; analogously, high-dimensional constellations induce non-uniform probabilities on the sequences of 2-D points which are transmitted as their coordinates. As the number of dimensions goes to infinity, this distribution tends to (guess what!) a Gaussian distribution – the function in eq. (15), while not a Gaussian, is certainly closer to it than a uniform distribution. Now, this is precisely the kind of signal Shannon's theorem calls into play when capacity is to be attained, and this is where the shaping gain comes from. To see this, remember that the information capacity C for a memoryless channel is [6]

$$C = \max_{p(x)}\{I(X;Y)\}$$

where as usual $I(X;Y)$ is the mutual information between the channel's input X and its output Y, and $p(x)$ is the pdf for X. For a Gaussian channel $Y = X + Z$, where Z is an iid Gaussian random variable of variance $N_0/2$, as we have seen; this allows us to write

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - H(Z) \\ &= H(Y) - \frac{1}{2}\log \pi e N_0 \end{aligned}$$

The channel is however power-limited too; this puts a constraint on the channel's input variance, so that it must be $\mathrm{E}[X^2] \leq P$, and therefore $\mathrm{E}[Y^2] \leq P + N_0/2$. Now, we know that for a given variance the entropy is maximized by a Gaussian distribution, so that a Gaussian distribution for X (and thence for Y) maximizes the mutual information under these conditions. High-dimensional signaling can thus be seen as an *indirect* way of biasing the transmitted signal to a Gaussian signal (or to an approximation thereof) in order to maximize channel's capacity. There are more direct and controllable ways to achieve this, though, which we will examine in the next section.

5.3 Biasing Gain

So far we have assumed that the symbols in the constellation are used with the same probability, which makes sense since we are willing to suppose that the data bitstream is a maximum entropy signal and thus all m-bit combinations are equally probable (otherwise we would be wasting bandwidth, wouldn't we?). However, a significant gain can be achieved by non-equiprobable signaling. Let's consider the first constellation shown in figure 8 and let's divide its point in two subsets C_i and C_o, where C_i is the subset of the four "inner" points closer to the origin, and C_o is the subset of the four "outer" points. Assume to utilize a symbol-to-point mapping scheme so that subsets C_i and C_o are selected with probabilities p and $1-p$ respectively; the points within C_i and C_o are used equiprobably with probability $1/4$. It is easy to see that for this constellation the energy factor is

$$S_A = 2p + 10(1-p) = 10 - 8p;$$

for $p = 1/2$ we are back to the equiprobable signaling case, and $S_A = 6$ like before, but for $p = 2/3$ we obtain $S_A = 4.67$ which is a 1 dB gain over the equiprobable constellation.

Yes, it is the same gain offered by the second constellation in figure 8 but this time with the important difference that the arrangements of points *is* drawn from a rectangular grid. This doesn't however come for free. If we simply map received points to alphabet symbols and compute the entropy of the resulting sequence Z, we obtain

$$H(Z) = 2 + p\log_2 p + (1-p)\log_2(1-p); \tag{16}$$

for $p = 2/3$, $H(Z) = 2.9$, which is less than for a maximum entropy information sequence for the constellation (which is $H(Z) = 3$, since we can map 3 bits per point). This means that the effective biasing gain is less than the value we have just computed since, intuitively, it would take less energy to send a bitstream with less information. In the case of this small constellation it is not immediate to quantify this reduction precisely; for larger constellation we can however invoke the continuous approximation, and we will examine this general case by describing in a much simplified way the actual biasing method used by V.34 [7].

5.3.1 Biasing fundamentals

We start from a constellation of circular shape and we divide it into M rings of *equal area*, as in figure 10; this means that each ring contains the same number of constellation points. If r_0 is the radius of the innermost ring, it is easy to show that for $i = 1, \ldots, M-1$

$$r_i = r_0\sqrt{i+1};$$

since the points within each ring will be used equiprobably, the power associated with each ring is

$$\begin{aligned} S_i &= \frac{1}{2\pi(r_i^2 - r_{i-1}^2)} \int_0^{2\pi} d\theta \int_{r_{i-1}}^{r_i} \rho^2 \rho d\rho & (17) \\ &= \frac{r_i^2 + r_{i-1}^2}{4} & (18) \\ &= (2i+1)S_0 & (19) \end{aligned}$$

where S_0 is the power of the innermost ring, which depends only on the number n_0 of inner points and which can be approximated as $S_0 = (2/\pi)\, 2^{n_0}$ for the odd-indexed square lattice. Each ring is to be selected during transmission with probability p_i, so that the signal power will be

$$S = S_0 \sum_{i=0}^{M-1} (2i+1)p_i.$$

The information carried by the signal can be divided in two parts; the first is associated with the index of the point within a ring and, since the points are equiprobable, its

[7]In the following, we will treat all quantities as real numbers; not to worry, though: fractional bitrates can be dealt with by block processing, and non-integer constellation sizes by nearest- integer approximations which, in the case of large constellation sizes, will not exceedingly affect the final results.

entropy will be $H_1 = n_0$; the other is associated with the index of the ring and its entropy is

$$H_2 = -\sum_{i=0}^{M-1} p_i \log_2 p_i. \tag{20}$$

If we were to transmit this total information with equiprobable signaling we would need a circular constellation of average energy

$$S_{eq} = (2/\pi)2^{(H_1+H_2)} = S_0\, 2^{H_2}.$$

The net biasing gain is then:

$$\begin{aligned}\gamma^2 &= \frac{S_{eq}}{S} \\ &= \frac{2^{\left(-\sum_{i=0}^{M-1} p_i \log_2 p_i\right)}}{\sum_{i=0}^{M-1}(2i+1)p_i};\end{aligned} \tag{21}$$

minimization of (21) under the constraint $\sum p_i = 1$ yields a parametric formulation for the p_i's [5]:

$$p_i = \frac{1-\lambda}{1-\lambda^M}\lambda^i; \tag{22}$$

with $0 \leq \lambda \leq 1$; this will not particularly surprise the "attentive reader" since, for large M, these p_i's approximate a binomial distribution, which is just the "discrete approximation" of a Gaussian. Substituting (22) into (21), the maximum attainable shaping gain for a chosen M can be found by numerical evaluation; it can be analytically proved however that, for a circular constellation, this maximum will tend to $e/2$ or 1.33 dB as M grows to infinity [5] (but $M = 16$ is pretty much all one needs). Just as promised, biasing gain plus the two-dimensional shaping gain yields $1.33 + 0.2 = 1.53$.

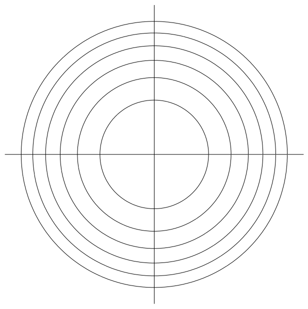

Figure 10: Constellation partitioning.

5.3.2 A practical example: the V.34 biasing scheme

Let's go back to our dear world of practice and let us examine what happens inside a V.34 modem: the user's maximum entropy data stream arrives at R bits per second and it is segmented in *mapping frames*, each of which corresponds to 8 consecutives QAM symbols; each frame is thus $L = 8RT$ bits, where $1/T$ is the baud rate of the connection. Of these L bits, K go into a coder which produces 8 ring indices according to some nonuniform probabilities p_i, $i = 0, \ldots, M-1$, while the rest is used to uniformly select a point within the subconstellations. The bitstream which goes into the coder is a maximum-information sequence; since the coder preserves this information, the number of bits K' at its output will have to be strictly greater than K, otherwise the bits would be uniformly distributed and the corresponding ring probabilities p_i would be uniform. This is independent of the particular addressing scheme for the rings, and it determines an expansion of the constellation with respect to the equiprobable case. The number of points within each subconstellation will be $n_0 = 2^{0.5+(L-K)/8}$ (the half bit in the exponent derives from the fact that we are using trellis coded modulation; it will be clear later).

If we are not faint of heart we can look at the V.34 specifications and pick for example the following case:

- baud rate $1/T = 3200$ symbols per second,
- user's bitrate $R = 28800$ bps,
- $M = 14$ rings,
- $K = 28$ bits;

for these values, $L = 72$, $n_0 = 2^{0.5+(L-K)/8} = 64$, so that the total number of points in the constellation is $n = Mn_0 = 896$. Since each frame is 8 symbols, there are $14^8 \approx 2^{30.45}$ possible sequences of ring indices to choose from each frame, and we will have to select $2^K = 2^{28}$ of them to obtain an invertible mapping between data bits and ring sequences. The philosophy behind the coder in V.34 is to order all the possible sequences of 8 ring indices according to their cost in terms of power: the sequence $(0, 0, 0, 0, 0, 0, 0, 0)$ for instance would be the least expensive and $(13, 13, 13, 13, 13, 13, 13, 13)$ the most expensive; after this, the first 2^K are selected for the mapping. However, this is easier said than done: given the number of possible sequences, table lookup methods are out of the question and an (involved) algorithmic procedure becomes necessary; the one implemented in V.34 is called *shell mapping*, but we will not describe it in detail. Returning to the practical example we are considering, numerical evaluation of the power of the nonequiprobable constellation as determined by the shell mapper yields $S = 401.57$. To transmit the same information through equiprobable signaling we would need a constellation of size $n = 2^{RT+0.5}$ points, and the closest viable value for this is $n = 768$ points; the gain for this signaling combination is thus:

$$\gamma^2 = \frac{401.57}{(2/\pi)\,768} \approx 0.86\text{dB}.$$

Numerical evaluation of (21) for $M = 14$ yields $\gamma^2_{\max} \approx 1.28$ dB.

5.3.3 Final remarks

There are several possible implementations for the coder which provides the ring indices. For a given number of rings, we know what the optimal probabilities are; deploying a coder which provides just that is however an entirely different matter. For the transmission parameters of the previous example, for instance, we can see that the coder acts in practice as a 28/32 block code; table lookup techniques are not viable for these blocksize values, and algorithmic procedures which yield effective code rates close to the optimal rate of equation (20) are very well known to be hard to design. The shell mapper implemented in V.34 [7] is a remarkably efficient and, just as importantly, very flexible algorithm which fits very well into the multirate framework of V.34 . It is very interesting to note that its original formulation was in relation to vector quantization for lossless compression of stationary sources, which is another example of the general duality between compression and coding.

5.4 What Next?

We started this section showing how there is a 9 dB margin for improvement over uncoded QAM modulation. So far we have gone through pretty complicated techniques and, let's face it, we've managed to scrap a tiny 1-1.3 dB gain out of all that (modem engineers are not *that* enthusiastic about having to implement shell mapping, let me tell you). So where should one go for the big gains? Well, in 1982 Dr. Ungerboeck showed the world, and in the next section we will follow his steps introducing the killer coding technique for QAM signaling: trellis coded modulation.

6 Trellis Coded Modulation

All the techniques we have seen so far can be grouped under the common heading of *channel coding techniques*. Their aim is to represent the data stream by means of a signal which best fits the available communication channel, which in our case is the telephone line. Other types of channel (such as satellite links or fast-fading radio channels) possess different physical characteristics which render some of the tricks we explained before either useless or downright bad.

Under the same heading are most of the tools which are concerned with the coding of the data stream before modulation proper. The coding deliberately introduces redundancies in the data in the form of error control codes; even though error control codes increase the net bitrate we have to modulate, they also decrease the probability of error at the receiver, so that in the end (hopefully) a net coding gain arises. This is actually where Information Theory comes into play in all its elegance: we have already said in Section 3 that we can trade reliability for bandwidth and/or power; channel coding allows us to do that in a very sophisticated way, limited only by the signal processing resources we have at hand.

The breakthrough in modem performance after 1984, reflected in the steeper portion of the curve in Figure 1, coincided with the adoption of an error control code *as a part of the modem specifications.* More precisely, the error control code is not separated from the modulation process but is developed as an inherent part of it, therby removing the artificial separation between modulation and channel coding scheme designs. This transmission technique goes under the name of *trellis coded modulation* and it originated from the insightful combination of three fundamental ideas: set partitioning, trellis encoding, and soft Viterbi decoding. In the reminder of this section we will examine these three ideas from a intuitive point of view and we will try to bring them together to complete our exploration of the inner working of a modem.

6.1 The intuition behind soft decoding

Let us pick a very simple error control scheme. Suppose, to make things simple, that we want to use a size-2 PAM system to send a binary stream at one bit per symbol; the corresponding alphabet for the transmitter is

$$A = \{+1, -1\}$$

and in the following we will identify binary 0 with -1 and binary 1 with +1 for convenience. The original bitstream is processed by a rate 2/3 parity check coder prior to transmission: the coder is just a box which takes data bits two at a time and associates each pair of input bits to a triple of output bits according to the following scheme:

$$\begin{aligned}
(-1, -1) &\rightarrow (-1, -1, -1)\\
(-1, +1) &\rightarrow (-1, +1, +1)\\
(+1, -1) &\rightarrow (+1, -1, +1)\\
(+1, +1) &\rightarrow (+1, +1, -1);
\end{aligned}$$

the minimum Hamming distance between codewords is 2, and it is easy to verify that this code is capable of detecting (but not correcting) any single error.

In Section 4.4 we said that the demodulator converts the analog symbol stream to a digital data stream by associating each received and noise-corrupted symbol $\hat{a}[k] = a[k] + n[k]$ to the nearest alphabet symbol. This process (which is inherently non-linear) is performed by a "circuit" in the receiver called a *hard slicer*. We will now show that, if the original data stream has been processed by an error control coder, this upfront slicing is not really the best possible idea. We assume perfect system synchronization, so that in the end, after decoding, the receiver will produce 2 "true" data bits every $3T$ seconds, where $1/T$ is the baud rate as usual. A hard slicer for the demodulator is simply $h(t) = \mathrm{sgn}(t)$, and the decoding of the code can be done by table lookup: the Hamming distance between the triple of hard-sliced bits and each of the codewords is computed and, if none of these distances is zero, an error is signaled. Now, suppose the transmitted triple is $(-1, -1, -1)$ while the received triple before slicing is $(-1.2, 0.15, -0.9)$; if we hard-slice the single bits we obtain $(-1, +1, -1)$, which is not a legitimate codeword, and we have an error. However, if we compute the squared Euclidean distances between the received triple and the four codewords we obtain

$$\begin{aligned} d_1^2 &= 1.3725, \\ d_2^2 &= 4.3725, \\ d_3^2 &= 9.7725, \\ d_4^2 &= 5.5725; \end{aligned}$$

it is clear that the codeword with minimum distance is the correct one, and this process of selecting the most likely codeword in terms of minimum Euclidean distance is called *soft decoding*. We could in fact show that for this 2/3 parity code, soft decoding offers a 3 dB gain over hard decoding; intuitively, it now takes a noise sample with squared amplitude twice as big to cause an error, and if all squared distances are greater than 4 an error is signaled.[8] In some cases, however, we might be paying a price in terms of system complexity: obviously for a hard-wired decoder it is much harder to compute Euclidean distances rather then Hamming distances. But this is really not an issue if we are using a programmable DSP, like in a modern modem.

Let's look at the situation from another point of view. The use of error control code replaces the bit-by-bit decoder of a plain 2-PAM decoder with a size-3 codeword decoder; the efficiency of the scheme arises from the fact that *not all* the possible 3-bit sequences are allowed. We can visualize this geometrically by representing the codewords as points in 3-D space (see Figure 11: all length-3 sequences are located on the 8 vertices of a cube of side 2; the minimum distance (Hamming or Euclidean) between any two sequences is still equal to the minimum distance between single 1-bit symbols; the four legitimate codewords however lie on alternate vertices only, so that their minumum distance is increased (look at the black dots in the figure: their square Euclidean distance is greater by a factor of 2). At this point the type of slicing comes into play. Hard slicing would

[8]Please note: this is not the overall coding gain with respect to uncoded signaling, because now we are sending 3 symbols in the same time we were sending 2 before; it is just a gain over hard decoding.

force each received triple to the nearest vertex of the cube *before* a decision on the triple is made; if the vertex is not a codeword an error can be signaled, but the decoder cannot resolve the ambiguity. Soft decoding on the other hand associates the sequence to the nearest *admissible* vertex, thereby taking full advantage of the increased minimum distance.

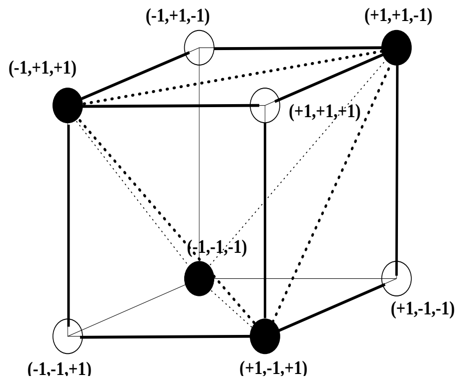

Figure 11: Geometrical representation of the parity check code.

6.2 The intuition behind trellis coding and Viterbi decoding

The geometrical interpretation we have just seen can be generalized as follows. A first step is to group symbols together into M-symbol sequences so that the the decision process from one-dimensional becomes a M-dimensional problem (for instance, $M = 3$ in the previous example); this however does not lead to any gain per se since if the sequence space is just the Cartesian product of M times the symbol space or, in other words, if all M-symbol sequences are codewords, the minimum distance between different sequences is the same as the minimum distance between symbols (in the previous example this would be the distance between any two vertices of the cube). The trick is then to "weed out" some of the possible sequences so that the codeword set is a proper subset of that Cartesian product but with an increased minimum distance between set elements; this process is performed by the chosen coding scheme (the parity encoder in the previous example). In the case of QAM modulation, we would encode the data at the transmitter so that not all possible sequences of QAM points are allowed; at the receiver, instead of deciding on each received symbol by associating it to the nearest point in the constellation, we would compute some distance measure between the received sequence and all allowed sequences. The net effect of this scheme, as we anticipated while talking of shape gains, is to create a "virtual" multidimensional constellation (of dimensionality NM, if the basic constellation is N-dimensional) whose underlying lattice contains "holes" which correspond to the unlegitimate symbol sequences; that is to say, the points in the new lattice are more spaced apart than the basic odd-indexed grid we have seen so far, which increases their minimum distance. The practical problem with this approach is that the number of sequences grows as 2^{mM}, where as usual m is the number of bits per symbol in the basic constellation; even for moderate values of M, sequence mapping and decoding rapidly require the use of some smart coding techniques instead of exhaustive table-lookups.

One class of codes which can be used to this aim is the class of *block codes*, of which the parity encoder we saw previously is just a simple example. Famous members of this class are the various Hamming codes, Golay codes, and BCH codes; the Reed-Solomon codes used in audio CD are for instance a particular type of BCH codes. The general principle behind block codes is to segment the input bitstream into k-bit blocks, and to map each of these blocks onto a n-bit codeword, with obviously $n > k$; such a code will be labeled a k/n block code. For a survey of block codes, see for instance [2] or [3].

Another class of codes which has found widespread application especially in conjunction with narrowband signaling is the class of *trellis codes*. In these codes, the bitstream is not split into blocks but used seamlessly as the input to a *trellis encoder*, which for the time being we can imagine as a sort of IIR filter; the output bitrate of the trellis encoder is greater than the input bitrate, so that trellis codes can also be labeled with a parameter of the form k/n, which is the ratio between input and output bitrates. Since in this type of coder the current output depends on all previous inputs (as opposed to block codes), trellis codes are also called *convolutional codes*.

There are two fundamental advantages with trellis codes with respect to block codes. First, they can be designed much more flexibly in relation to the particular transmission problem at hand. Second, for trellis codes there exists an optimal decoding procedure which is also easy to implemented in practice and perfectly scalable, and this is the famous Viterbi algorithm. Even though the computational requirements of the Viterbi algorithm are quite substantial, the particular case of narrowband transmission and the availability of programmable DSP's allow for the use of such a relatively time-consuming technique; "narrowband" indeed implies that we have a lot of time for symbol processing.[9]

6.2.1 Representation of trellis codes

Each author has his own favorite way of explaining trellis coding and Viterbi decoding, which is usually pretty wordy[10] – in these notes we will just go through the basics without any attempts at rigor, while the interested reader is once again referred to [2] and to [3]. A trellis encoder is probably best viewed as a *finite state machine*; in this perspective it is characterized by the following parameters:

- the number of distinct internal states, 2^L; in the implementation this usually corresponds to a L-bit shift register (memory);
- the coder's rate k/n: at each time interval the encoder takes k bits as an input, determines a state transition (there are 2^k possible transitions for each state), and

[9]Compare the situation with, for instance, what happens inside a CD player where the data rate is around 4 Mbit per second: DSP processors this fast did not quite exist in the '70s, and hard-wired implementations of the Viterbi algorithm are difficult and costly; for block codes, however, there exist hard-slicing decoding techniques which can be efficiently and cheaply implemented in hardware.

[10]I think the quickest way to fully understand the Viterbi algorithm is just to write an actual piece of code implementing it – just a hint.

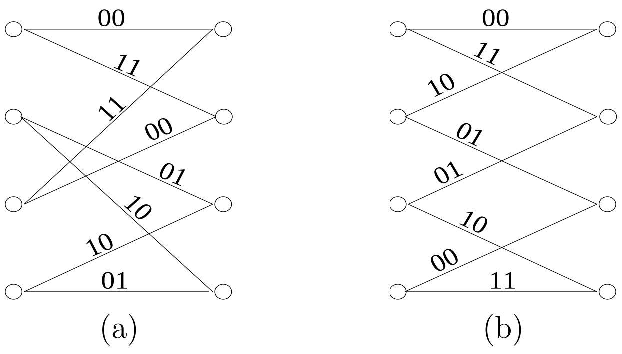

Figure 12: Two possible rate 1/2 trellises.

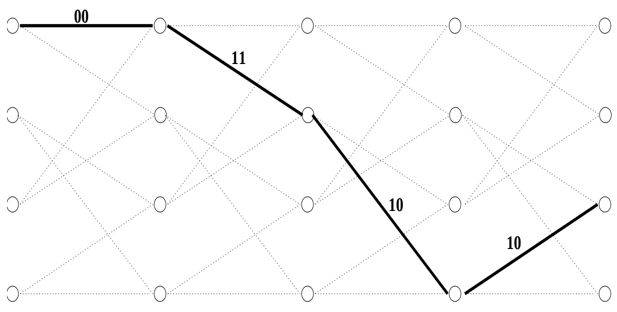

Figure 13: Trellis coding for the sequence $\{0, 1, 1, 0\}$, using the trellis of figure 12(a).

produces n bits as an output;

- the state machine's transition table, with relative outputs, which will consist of $2^{(L+k)}$ entries.

All this information can be easily represented by a *trellis diagram* (thence the name of the encoder). Figure 12 shows two such diagrams for two possible rate 1/2 trellis encoders with $L = 2$; the dots on the left and on the right represent the four possible internal states of the encoder and the branches connecting them are the state transitions determined by the input, ordered from top to bottom in ascending order (that is to say: the uppermost branch sprouting from a state corresponds to a value of zero for the input, while the lowest branch corresponds to the largest input value, which is 1 in this example). The labels over each branch represent the encoder's output bits for each transition. The encoding of a l-point input sequence can be graphically represented by replicating the basic trellis unit l times, and by drawing the path corresponding to the successive state transitions. Figure 13 illustrates the process for the input sequence $\{0, 1, 1, 0, \ldots\}$ using the trellis in figure 12(a); the initial state for the coder is state zero by convention.

6.2.2 Decoding

Decoding at the receiver takes place by trying to reconstruct the transmitter's trellis path from the noise-corrupted data using a maximum-likelihood approach. This is carried out efficiently by means of dynamic programming, and it all goes under the name of *Viterbi decoding*. The Viterbi algorithm is thus an efficient way of computing the distance between the received sequence of symbols and all the possible sequences allowed by the transmitter's trellis encoder; the minimum-valued distance sequence is then picked by the decision algorithm as the real transmitted data. The metric used by the Viterbi algorithm can be any: of course, hardware implementations will tend to prefer something like a Hamming distance, for instance. But in conjunction with QAM signaling, soft decoding with an Euclidean metric is the choice of rigor, we should say, yielding at least a 2dB gain over hard slicing.

Let us try to explain the basics of Viterbi decoding by means of an example using the trellis code of Figure 12(a). The input sequence is $\{0, 1, 1, \ldots\}$, so that the transmitted sequence is $\{0, 0; 1, 1; 1, 0; \ldots\}$, where the semicolon separates the trellis symbols.[11] Suppose that, due to the noise, the received sequence becomes $\{0.3, -0.2; 0.6, 0.45; 0.7, 0.42; \ldots\}$. At the very beginning the receiver knows the transmitter is in state zero by convention. There are thus only two possible symbols which could have been sent at time zero: the actual received symbol is compared to these two and the relative squared Eucidean distances are computed. The receiver starts building its own version of the trellis by drawing the two possible branches and labeling them with the just-computed distances (fig. 14(a)). At the second step, we have two hypotheses on the previous state for the encoder, so we consider in turn all the branches which can stem out of these two hypothetical states, compare the corresponding potentially transmitted symbols to the received one, compute the squared Euclidean distances and label with these the corresponding branches on the local copy of the trellis (fig. 14(b)). From now on we will have four hypotheses on the encoder's previous state and we will go on just as before computing squared Euclidean distances associated to the branches for each new incoming symbol and extending the trellis to the right. At the end (and we will see in a second when the end is) we will just select the path in our local replica of the trellis which has the minimum cumulative metric of all (that is, the minimum sum of all the branch labels). We can do this because the squared Euclidean distance is additive: remember that each received symbol is just one coordinate of our hypothetical M-dimensional point, so that the cumulative metric of the sequence is just the squared Euclidean distance between the received series of coordinates and the candidate M-dimensional symbol. The running cumulative distance is printed at the end of the paths in figures 14(a) to (d).

It seems at first sight that proceeding this way leads to an exponential growth in the number of paths in the local trellis, but fortunately it is not so. Consider figure 14(c), which is the third step of the decoding example we've been seeing so far; we see that two paths converge momentarily onto state number 2 at iteration number 2. In the end,

[11]Here a *symbol* corresponds to two bits: this could be transmitted as a single channel symbol if we use multilevel modulation, or as two channel sybols in a row if we use a simpler, two-level modulation scheme

when we are selecting the minimum metric path, suppose we find that the optimal path passes indeed through state 2 at iteration 2; well, since the optimal path is minimum metric, its “tail” between iteration 0 and 2 must be minimum metric too, so even at iteration 2, without waiting for the end, we can already discard the path with the larger metric of the two that converge at node 2 (the lower path in the figure). This is really just dynamic programming and, as a consequence, at any time during decoding only 2^L paths survive. Figure 14(d) represents the receiver’s trellis at the end of the third iteration after the unnecessary paths have been pruned (thin lines).

One final word about when the “end” of the process occurs: since there is no block partitioning of the incoming bitstream, theoretically we should wait till all the available data has been transmitted. Practically, since we can’t do that of course, we just wait for a while before carrying out the first real slicing ($5L$ iterations is the rule-of-thumb delay) and then, at each new time interval lT, we do the following things:

- select the minimum-metric path;
- declare the first symbol of the path as the real transmitted symbol $5L$ symbols ago;
- shift left the whole trellis by one by dropping the fist stage and renormalize the metrics;
- extend the trellis to the right with the new incoming data.

The net effect of this is that we are now using a slightly suboptimal version of the Viterbi algorithm, but we are getting one symbol out of the decoder at each time interval. The fundamental tradeoff is between allowed decoding delay and reliability on the symbol decision.

The computational requirements of the Viterbi decoder are roughly proportional to the number of states both in memory requirements and in number of operations per symbol; in V.34 there are three possible encoders with 16, 32, and 64 states respectively. With the current, baseline DSP’s available today, however, the 64-state encoder requires a somewhat simplified form of the Viterbi algorithm in order to execute in real time.

6.2.3 How does it work ?

We said earlier that the trick to coding is to move the slicing process to a space of symbol sequences where not all possible sequences are allowed. In a trellis coder it is always $k < L$ so that from a given state not all the other states can be reached; this limits the number of possible paths in the trellis, and therefore the number of possible output sequences. This said, we next have to look at how the minimum distance between allowed sequences is affected by the encoding process. The worst-case scenario is that of two paths in the trellis which are the same up to a certain point, which then diverge, then merge again, and from then on keep on being the same; what is the minimum distance in this case ? The first thing one should try to maximize is the minimum number of transition that have to pass before the two paths can merge again after they have diverged. It is easy to verify for instance that for the trellis in figure 12(a) this

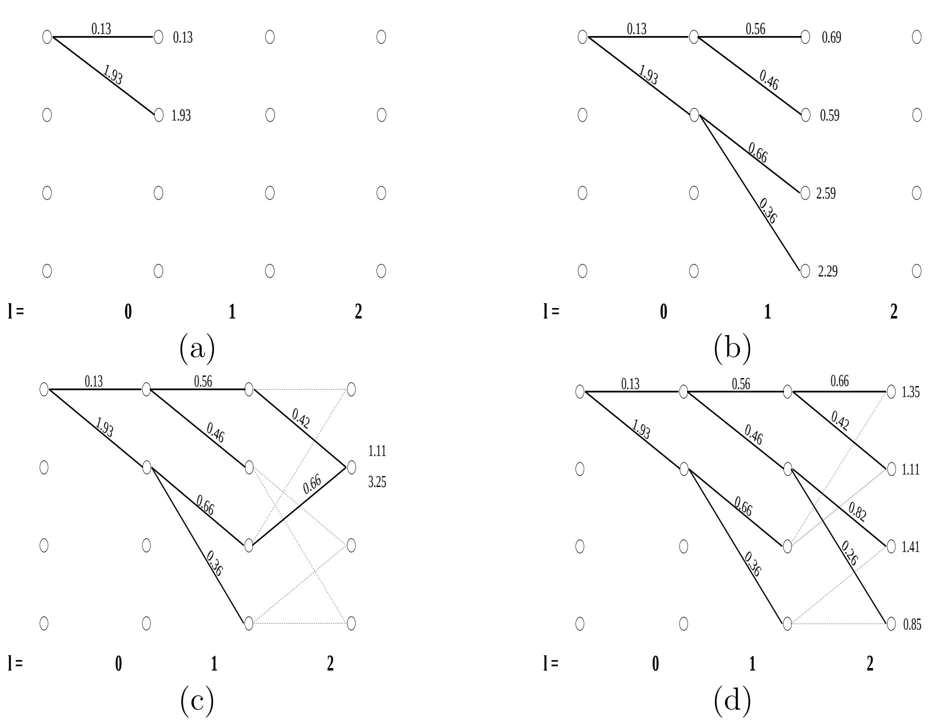

Figure 14: First three stages of decoding with the Viterbi algorithm. Input sequence: $\{0, 1, 1, \ldots\}$; transmitted sequence $\{0, 0; 1, 1; 1, 0; \ldots\}$; received sequence: $\{0.3, -0.2; 0.6, 0.45; 0.7, 0.42; \ldots\}$.

number is three, while for the trellis in figure 12(b) it is only two; figure 15 shows a case of two minimally-diverging paths. The next thing is to assign output symbols to the transitions so that the minimum distance between re-merging paths is maximized. A rule of thumb to assign the symbols is to assign maximally distant symbols to the k branches which stem from the same state and maximally distant symbols to the k branches which converge to the same state, if possible. This way, the minimum distance between sequences is at least twice the distance between maximally distant symbols. In figure 12(a), for instance, this principle is followed and it can be proved that the minimum Hamming distance between any two sequences is 5; in figure 12(b) it is not possible to do that, and the minimum Hamming distance is only 3. It is clear that this trellis is poorly structured and inefficient. The design of trellis codes is sort of an art, especially when other properties, which we will not examine but which are higly desirable, are taken into account; for basic trellis codes, however, systematic computer searches have been carried out and a set of "recipes" for the optimal trellises for most rates is available in the literature ([3]).

6.3 The intuition behind set partitioning

At this point we have seen which coding (and decoding) technique could be used to obtain sequences of QAM points with desirable intersequence distance properties from the input bitstream. There are two problems worth noting, though. From a practical point of view a convolutional encoder which takes m bits as an input when m is large requires an exceedingly costly Viterbi decoder – remember, m is the number of bits per symbol and for instance, at a baud rate of 3200 at 28800 bps, $m = 9$; this would require at least a 1024-state encoder ($k = m$ and $L > k$), while in "cost-efficient" V.34 implementations a 64-state Viterbi decoder is already an upper limit in computational requirements. Furthermore, from a theoretical point of view, there is nothing in trellis encoding per se which makes it suitable to QAM signaling: in the previous sections, we have proven a series of properties for the uncoded constellation which relied on the fact that the constellation points were being used according to certain probabilities (uniform in the case of shaping gains, gaussian in the case of biasing gains); the set of sequences obtained by trellis encoding the data bitstream might be such that these probabilities can not be attained. There is a way around both problems, however. Suppose we take our basic constellation, let us take it two-dimensional for the sake of simplicity, and we alternatingly label its points with A's and B's as shown in figure 16(a). The original constellation is thus split into two subconstellations of half the size, rotated by 45 degrees; given the regularity of the labeling scheme, all the shaping properties we so painstakingly endowed the original constellation with are gracefully passed on to the two smaller subconstellations. However note that now, for the A and B subconstellations, the minimum squared distance between points has doubled. This process can (and is) of course carried further, by alternatingly relabeling the points in each of the subconstellations (see figure 16(b)); after n such steps we obtain 2^n subconstellations, and the minimum squared distance between points in each of these is $2^n d_{min}^2$, where d_{min} is the minimum Euclidean distance between points in the original costellation – for the QAM systems we saw in the previous sections, $d_{min} = 2G_0$. It is also important to note that the minimum squared distance between points which belong to any two different subsets goes from a maximum value of $2^{n-1} d_{min}^2$ (see for examples subsets A and C in figure 16(b)) to a minimum value of d_{min}^2 (subsets A and B).

Now, suppose we knew at the receiver which subset each of the incoming symbols belongs to; then the probability of erroneously slicing the symbol would be very very

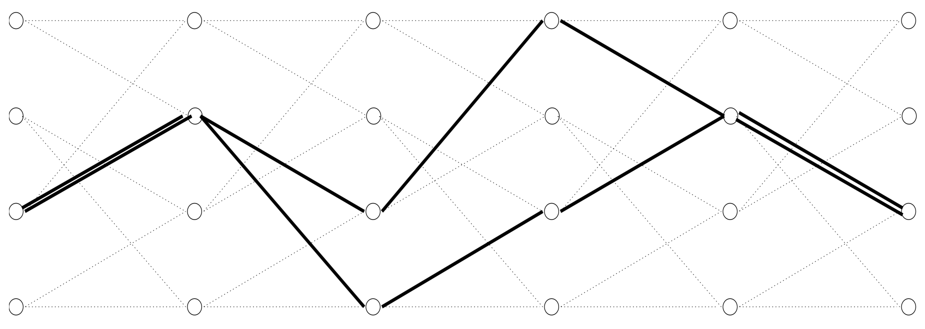
Figure 15: Two diverging and remerging paths in the trellis

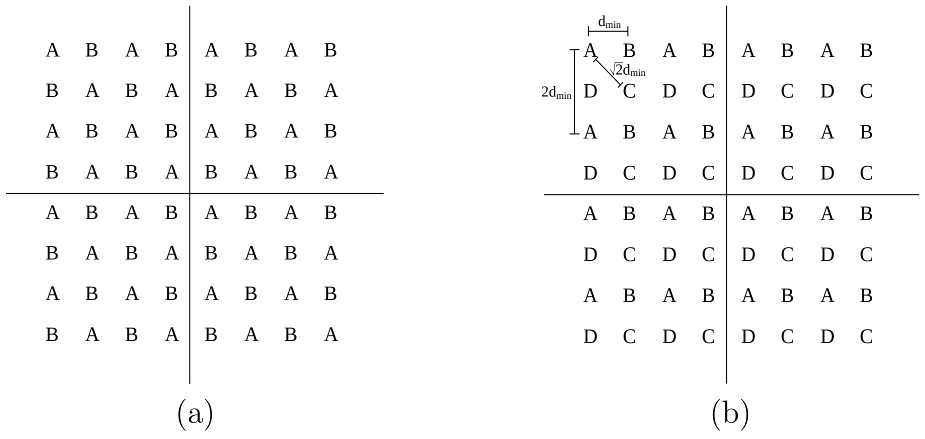

Figure 16: Example of set partitioning.

small, since the squared Euclidean distance between points in the given set is 2^n times the original minimum squared Euclidean distance. In practice of course we don't want to know the subconstellation index a priori because we want to use it to carry information as well (it is like what we saw for the ring indices in the biasing scheme). But, and this is Ungerboeck's brilliant idea, we can encode the transmitted data so that *not all sequences of subconstellation indices are possible*; if the receiver manages to correctly identify the sequence of subconstellations, then the probability of erroneously slicing the single QAM points will be negligible.

6.4 Putting it all together

The global coding scheme is thus the following. At each symbol interval the m incoming bits are split in two parts: k bits enter a rate $k/(k+1)$ trellis encoder which selects one of 2^{k+1} possible subconstellations, while the remaining $m-k$ bits are used to select a point within the subconstellation either uniformly or according to a given biasing strategy.[12] Starting from a constellation of 2^{m+1} points, we determine 2^{k+1} subconstellations by subset partitioning; we then build the trellis encoder labeling the branches with subconstellation indices according to Ungerboeck's rules:

- branches coming out of and going into the same state are assigned maximally distant subsets;
- all the subcostellation indices must appear with equal frequency and regularity in the trellis;

Figure 17 shows for instance the trellis of figure 12(a) relabeled according to these rules.

At the receiver, decoding takes place in two steps. First, for each incoming, noise-corrupted symbol we select the nearest 2^{k+1} constellation symbols. Given the way the

[12]Other encoder rates than $k/(k+1)$ can be used, of course, but this is the rate used in both V.32 and V.34.

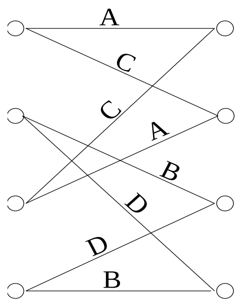

Figure 17: Trellis relabeled with subconstellation indices

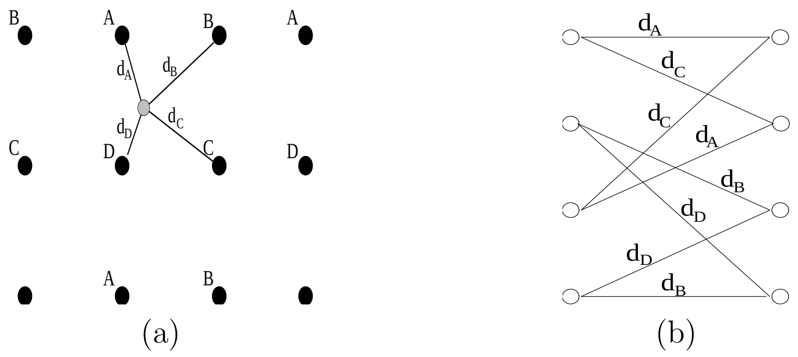

Figure 18: (a) Nearest-point distances – (b) Corresponding branches in the Viterbi algorithm trellis.

constellation is partitioned, each of these will be univocally associated to a different subset. We then compute the squared Euclidean distances between the received symbol and the above points: we take these as the distances between the true and received subconstellations and we use them as the branch metrics for the Viterbi decoder. After the usual decoding delay, we will have a reliable decision on the subconstellation for the symbol: the output of the decoding process will then be that symbol, among the 2^{k+1} selected above, which corresponds to the just-determined subconstellation. Figure 18(a) shows the distances between the received point (the grey circle) and the four nearest candidate points for the constellation of figure 16(b); figure 18(b) shows how the Viterbi algorithm would label the branches of the trellis if the transmitter's encoder is that of figure 17.

Trellis coded modulation increases the minimum distance between the points the receiver has to tell apart; on the other hand, since we now have to map a total of $m+1$ bits onto the constellation, we will have to use a constellation of twice the size compared to the uncoded case. The total coding gain can be estimated as

$$\gamma^2 = 10 \log_{10} \frac{d^2_{min,coded}}{d^2_{min,uncoded}}. \tag{23}$$

With no coding, we would need a constellation of 2^m points, with minimum distance $d_{min,uncoded} = 2G_0$. With coding, we need a constellation of twice the size; since we must signal at the same power, we will have to scale down the expanded constellation so that the minimum distance between its points will be $d_{min,coded} = \sqrt{2}G_0$ (remember that, with the continuous approximation, the number of points is proportional to the area of the constellation, and so is its power). Coding, however, increases the actual minimum distance by a factor of K so that

$$\gamma^2 = 10\log_{10}\frac{K^2\,2G_0^2}{4G_0^2} \approx 10\log_{10}K^2 - 3 \text{ dB}. \tag{24}$$

Since K depends only on the structure of the tellis encoder we can see the coding gain as the "pure" trellis coding gain minus the energy taken by the fact that we doubled the size of the constellation (a factor of two is 3dB).

Let us combine by way of example the trellis of fiugre 17 to the partitioned constellation of figure 16(b). The minimum squared distance between sets A and C and between sets B and D is $2d_0^2$; given the way the trellis is defined, every time two paths diverge and then merge again their square distance increases by *at least*

$$2d_{min}^2 + 2d_{min}^2 = 4d_{min}^2;$$

in other words, $K^2 = 4$ and (24) yields

$$\gamma^2 = 6 - 3 = 3 \text{ dB}.$$

This is not a bad result for such a simple scheme, and that's why trellis coded modulation is the "standard" for voiceband data transmission. More complex but still practical trellises can yield up to 6 dB of coding gain, at the sole price of an increased number of multiplies/adds per decoded symbol.

6.4.1 Trellis coded modulation in V.34

The main difference between the simple trellis coding schemes we have seen so far and the actual system used in V.34 is that V.34 uses a four-dimensional constellation. The four-dimensional symbols are sent as pairs of two-dimensional QAM points, and the four-dimensional constellation is just the Cartesian product of a circular, 960-point 2D-constellation with itself. Looking at the specifications, the whole thing appears a little bit confusing because the various building blocks in V.34 sometimes operate on the sequence of 4D symbols, sometimes on the sequence of 2D symbols; the baud rate of the modem is however always referring to the 2D symbol sequence, so that at a baud rate of 3200, for instance, we are actually sending 3200 two-dimensional symbols per second or, equivalently, 1600 four-dimensional symbols per second. Oh well.

On the other hand, the rate $k/(k+1)$ trellis encoder always operates on the four-dimensional symbols; as a consequence, the additional bit introduced by the coder is split between two consecutive 2D-symbols (this is the half-bit we saw in section 5.3.2). The advantage of using a 4D constellation is then clear: the 3 dB penalty we were paying before for the constellation expansion is now halved. To see this more clearly, let us consider our familiar odd-indexed 2D lattice with minimum distance between points $d_{min} = 2G_0$; the Voronoi region for this lattice is just a square of area d^2_{min}. When we move to four dimensions by taking the Cartesian product of the 2D constellation with itself, the Voronoi region becomes a hypercube of volume d^4_{min}. The volume of any constellation depends just on the number of points P and on the volume of the Voronoi region, so that for the 4D constellation we have:

$$V = P d^4_{min}.$$

Now remember once again that, for a fixed given shape, the energy of the constellation is just proportional to its volume, with the factor S in equation (13) being the proportionality constant. Trellis coding requires us to double the number of points in the 4D constellation; to keep its energy constant we then have to scale down the minimum distance between points as

$$d^4_{min,coded} = \frac{V}{2P} = \frac{d^4_{min,uncoded}}{2}$$

or, equivalently,

$$d^2_{min,coded} = (1/\sqrt{2}) d_{min,uncoded}. \tag{25}$$

Trellis encoding augments the minimum distance by a factor of K like before, and by plugging equation (25) into (23) we obtain

$$\gamma^2 \approx 10 \log_{10} K^2 - 1.5 \text{ dB}.$$

Of course things are not so straightforward, and there are some subtler tradeoffs which have to be taken into account in the case of trellis coded modulation with multidimensional constellations. For the interested reader, a good exposition of the problems involved can be found in [8].

Finally, V.34 offers the choice between three different trellis encoders, with rates 2/3, 3/4, and 4/5 corresponding to 16, 32, and 64 state encoders respectively. The appropriate trellis is selected before the initiation of data transmission, after a probing of the telephone channel has been performed; poorer connections will require higher-order coders, with a maximum coding gain for the 64-state encoder in excess of 5.6 dB.

7 Conclusions

It should be clear by now that it is very difficult to reach any conclusive statement about modem performance. The fundamental concept is that there is no such thing as *the* capacity of the telephone channel: each connection has its own characteristics and, should the telephone network undergo a major general upgrade, the capacity of the average link would increase and modems would be designed anew to take advantage of this. V.34 has been designed as an adaptive system to exploit at best the majority of lines it is employed onto; from this point of view, V.34 reaches a data throughput which is remarkably close to capacity, up to a point in which the cost of improvements in performance would doubtfully pay back. But, after all, only time will tell.

As a final remark, it is important to say that in these notes we haven't had time to talk about the signal processing techniques which are used to implement a modem in practice. This is a very fascinating topic per se which, once again, pulls several disciplines together: adaptive filtering, to implement equalizers and echo cancelers; system theory, to implement feedback timing recovey circuits; discrete-time filter design, to implement the modulators and the demodulators; plus all the optimization techniques needed to achieve real-time performance on general purpose DSP processors. The difficulties inherent in this aspect of modem design are many, and should not be forgotten. In the end, the references will provide interesting material to those (still) interested.

References

[1] G.D. Forney, *et al.*, “Efficient Modulation for Band- Limited Channels”, *IEEE Journal on Selected Areas in Communications*, vol. SAC-2, pp. 632-647, Sept. 1984.

[2] R.E. Blahut, *Digital Trasmission of Information*, Addison-Wesley, 1990.

[3] J.G. Proakis, *Digital Communications*, McGraw-Hill, 1989.

[4] T.W. Körner, *Fourier Analysis*, Cambridge University Press, 1988.

[5] A.R. Calderbank, *Bandwidth Efficient Communication*, unpublished lecture notes.

[6] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[7] R.Laroia *et al.*, “On optimal shaping of multidimensional constellations”, *IEEE Transactions on Information Theory*, July 1994, pp.860-870.

[8] L. Wei, “Trellis-Coded Modulation with Multidimensional Constellations”, *IEEE Transactions on Information Theory*, vol. IT-33, no. 4, July 1987.

[9] J.A.C. Bingham, *The Theory and Practice of Modem Design*, Wiley, 1988.