Introduction to Modem Design

Paolo Prandoni, LCAV - EPFL

January 26, 2004

Part 1

Modulation

In order to transmit a signal over a given physical medium we need to adapt the characteristics of the signal to the properties of the medium. In the case of electromagnetic signals, the main object is to fit the spectrum of the signal into a prescribed bandwidth, called the *passband*, and this is accomplished by means of a technique called *modulation*. Modulation is performed by multiplying the original signal by a sinusoidal signal called *carrier*; the gist of the modulation theorem is that, in so doing, we are actually traslating the spectrum of the original signal in frequency, recentering it over the frequency of the carrier. This frequency is chosen according to the physical medium: copper wires, optical fibers, water, deep interstellar space, all require different modulation frequencies since their useful passbands are located in different portions of the spectrum. The passband of a communication channel is, roughly speaking, the part of the spectrum which behaves linearly for transmission; there, we can rely on the fact that a properly modulated sinusoidal signal will be received with only phase and amplitude distortions, and these are "good" types of distortion that we know very well how to handle.

The passband of a physical channel is of finite width, so we must make sure that the bandwidth of the original signal prior to modulation is "of the same size as the channel's passband. In other words, we must build a signal with a finite, prescribed spectral support. Intuitively speaking, the wider the support, the more information we can send over the channel. A big effort in designing a modem is trying to squeeze as much information as possible over the relatively narrow passband of the telephone channel, as we will see. The operation of limiting the bandwith of a digital communication signal goes under the name of *pulse shaping* and is basically a linear filtering operation.

To illustrate what modulation is all about, take the example of AM radio. The AM band extends from 530 KHz to 1700 KHz and each radio station is allowed by law to transmit over an 8 KHz frequency slot in this range. Assume we want to transmit speech

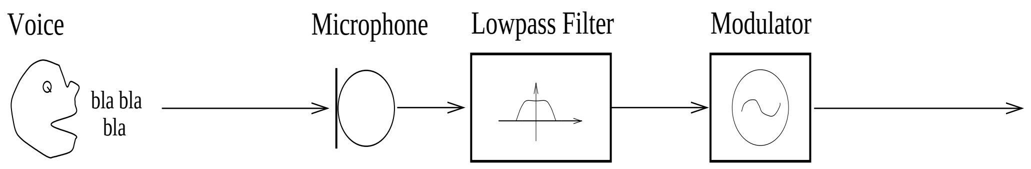

Figure 1.1: A simple AM radio transmitter.

over AM and we are given a slot from $f_{\min} = 650$ KHz to $f_{\max} = 658$ KHz, with the bandwidth $W = f_{\max} - f_{\min}$ equal to 8 KHz; the speech signal $s(t)$, obtained with a microphone, has a wideband spectrum which spans several KHz; we can however filter it through a lowpass filter with cutoff frequency 4 KHz without losing too much quality and thus reduce its spectral width to 8 KHz. The filtered signal has now a spectrum extending from -4 to 4 KHz; by multiplying it by a sinusoid at frequency $f_c = (f_{\max} + f_{\min})/2 =$ 654 KHz, we can shift it to the allotted AM band according to the *modulation theorem*:

$$s(t) \overset{\mathcal{F}}{\longleftrightarrow} S(f) \qquad \Longrightarrow \qquad s(t)cos(2\pi f_c t) \overset{\mathcal{F}}{\longleftrightarrow} \frac{1}{2}(S(f - f_c) + S(f + f_c)). \tag{1.1}$$

The scheme of the redio transmitter is displayed in Figure 1.1.

A third constraint on the transmitted signal limits its actual power. This constraint stems from the physical limitations under which the medium behaves linearly (imagine cranking the volume up on your stereo); in the case of telephone lines, it is also strictly regulated by the telephone companies who don't like their wires to be burned. Power constraints are going to affect the transmission scheme always with respect to the ubiquitous *noise* which degrades the transmitted signal. The keyword here is the *signal to noise ratio* (SNR), which is a measure of how good the channel is: the higher this figure, the better the received signal. We only have control over the power of the signal, the noise being a fact of nature which simply is there. We can therefore try to improve the SNR by augmenting the transmission power but, due to the power limitations inherent to the channel, we will ultimately have to stop and settle for a given SNR and design our receiver around that figure.

1.1 The telephone channel

The usable bandwidth of a physical communication channel can be determined by several factors; it can be fixed by law, as is the case for the different bands of the electromagnetic spectrum allocated to wireless communications (AM/FM radio, TV, mobile telephony); it can be limited by the sheer physical properties of the medium (as in the case of fiber

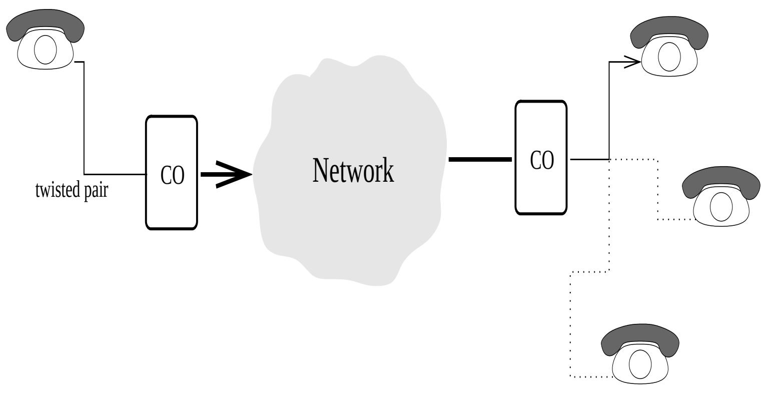

Figure 1.2: The Plain Old Telephone System.

communications); or, in the case of telephone lines, it can be dictated by a combination of (sometimes outdated) practical and economical considerations.

The POTS (Plain Old Telephone System) can be schematized as in Figure 1.2. The user's telephone is connected via a *twisted pair* (i.e. two plain copper wires) to the nearest Central Office (CO), usually at a distance in the kilometer range; dialing instructions imparted by the caller are interpreted by the CO and relayed over via the (world-wide) telephone network to the CO nearest to the called telephone. To understand the limitations of a telephone channel's bandwidth we have to step back to older, analog times when CO's were made of electromechanical switches and the connections inside the cloud labeled "Network" were mostly analog lines spiced up by a lot of operational amplifiers. The first link of the chain, the twisted pair, has no major bandwidth limitations, given the short distance it covers[1]; telephone companies however, used to introduce what are called *loading coils* in the line to compensate for the attenuation introduced by the capacitive effects of longer wires in the network. A side effect of these coils was to turn the first link into a lowpass filter with cutoff frequency around 4 kHz, so that most of the older textbooks will define the passband of the telephone channel as limited between $f_{\text{min}} = 300$ Hz and $f_{\text{max}} = 3000$ Hz, for a total usable bandwidth $W = 2700$ Hz. Later on, the network became mostly digital (and so the CO's), yet telephone companies found it convenient not to expand the nominal bandwidth of the user's channel so that many more conversations could be multiplexed over the same cable or satellite link. The standard sampling rate for a telephone channel is nowadays 8 KHz and the bandwidth limitations are imposed only by the antialiasing filters at the CO, for a maximum bandwidth in excess of $W = 3400$ Hz. The upper and lower ends of the band are not usable due to possible great attenuations

[1]This is the principle behind ADSL, by the way

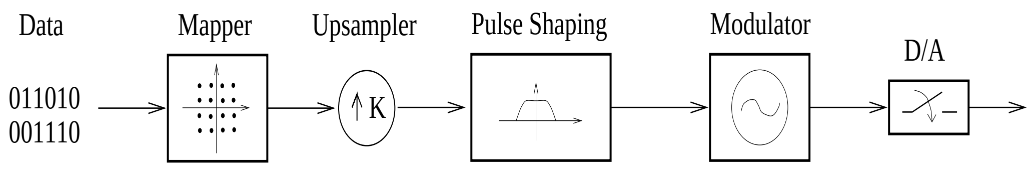

Figure 1.3: The modem's digital modulator.

which may take place in the transmission.

Under these bandwith constraints, the telephone channel can be quite accurately described as a linear, low-noise, strictly bandlimited channel. It is also power limited, of course, but this 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.

1.2 Digital Modulation

1.2.1 Preliminaries

Data trasmission over a physical medium is by definition analog: the universe *is* analog, after all[2]. However, most of the modulation techniques can and are performed in the digital domain, and this is the road we will follow, up to a final D/A converter which is our interface with the outside world.

The scheme of the modulator is shown in Figure 1.3 (please note the similarity with the AM transmitter of Figure 1.1). We initially have a data source (e.g. a computer) which produces a raw data stream of, say, R bits per second. This bit stream is split in consecutive groups of m bits and a circuit called a *mapper* uniquely maps each of the possible 2^m values of each group into a *symbol* $\boldsymbol{a}$. Symbols are basically signal amplitude values (a simple choice could be the decimal value of the m- bit group in Volts, for instance) and they belong to a finite *alphabet* A of size $M = 2^m$. The mapper produces a new symbol every is $T = M/R$ seconds, and $1/T$ is called the *baud rate* of the system.

Call the sequence of symbols at the output of the mapper $\boldsymbol{a}_i$. This sequence is not yet suitable for transmission; first we need to design its spectral characteristics so that it fits into the available bandwidth of the channel, and then we have to translate it in frequency to place it right in the passband of the channel. These functions are performed by a *pulse shaper*, which is basically a lowpass filter, and by a *modulator*. Before we consider the details of these two components, let's have a look at the constraints imposed by the channel in the case of digital modulation.

[2]Or maybe not...

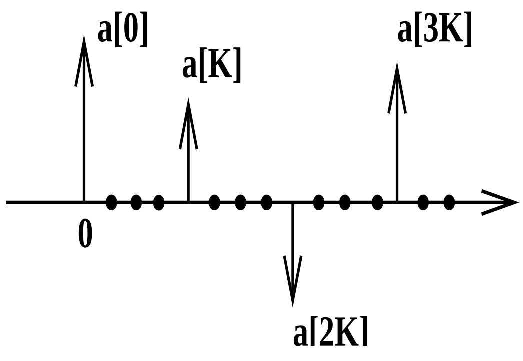

Figure 1.4: The digital symbol sequence.

Since, as we said, in the end we have to produce an analog signal, the last component of a digital modulator is a D/A converter at a sampling frequency of, say, f_s Hertz. All the digital processing taking place in the modulator is therefore happening at an internal sampling frequency f_s and therefore we can convert the analog specifications for transmission over the channel into digital frequencies. In particular the upper and lower bounds of the passband will become $\omega_{\mathrm{max}} = 2\pi(f_{\mathrm{max}}/f_s)$ and $\omega_{\mathrm{min}} = 2\pi(f_{\mathrm{min}}/f_s)$; the resulting width of the passband will be $\Theta = 2\pi((f_{\mathrm{max}} - f_{\mathrm{min}})/f_s)$ and the center-band carrier frequency will be $\omega_c = \pi((f_{\mathrm{max}} + f_{\mathrm{min}})/f_s)$.

In view of the derivations which follow, the actual value of f_s is not important at this time as long as:

- $f_s > 2f_{\mathrm{max}}$; i.e. the sampling frequency is greater than twice the largest frequency of the channel's passband;

- $f_s = K(1/T)$ with K integer; i.e. f_s is a multiple of the baud rate.

So, for instance, for a telephone channel's passband from 300 Hz to 3800 Hz and a baud rate of 2400, we could choose $f_s = 9600$ Hz.

The data symbols produced by the packetizer come out at a rate of $1/T$ and, in order to use them in the signal design system, we must bring them up to the right sampling rate. Thanks to our "smart" choice for f_s this is accomplished simply by a K-time upsampler, with $K = f_sT$, as shown in Figure 1.3. In the previous example, for instance, the upsampling factor would be $K = 4$. We will denote by $\boldsymbol{a}[n]$ the upsampled symbols sequence, an example of which is given in Figure 1.4.

1.2.2 The baseband signal: pulse shaping

The baseband signal, so called to tell it apart from the *passband signal* which is the baseband signal after modulation, is the output of the pulse shaping filter. Before we examine what the baseband signal looks like, let's take a closer look at the input to the system, i.e., at the upsampled data symbols sequence $\boldsymbol{a}[n]$. Since we don't know a priori what kind of data are generated by the computer, all we can do is consider the sequence as a realization of a M-valued discrete random process. We don't really know the statistics of this process, but it is a reasonable assumption to model it as an iid process with uniform distribution over the set of symbols $\boldsymbol{\alpha_i} \in A$. We will see later that the alphabet A is designed so that the process has zero mean and variance σ_A^2. Since the symbols are iid, it can be shown that the power spectrum of $\boldsymbol{a}[n]$ is

$$A(e^{j\omega}) = \sigma_A^2. \tag{1.2}$$

This power spectrum is flat over all frequencies and therefore its support is 2π. In order to reduce it to the required value Θ we will try to filter it via a pulse shaping filter.

The pulse shaping filter will be a L-tap FIR filter of impulse response $g[n]$ and frequency response $G(e^{j\omega})$. The filtered symbol sequence is therefore

$$\boldsymbol{b}[n] = \sum_{m=0}^{L-1} g[m]\boldsymbol{a}[n-m] \tag{1.3}$$

and, by a fundamental result of random signal processing theory, its power spectrum is

$$B(e^{j\omega}) = \sigma_A^2 |G(e^{j\omega})|^2. \tag{1.4}$$

How do we choose the shape of $S(e^{j\omega})$? First we have to look at what requirements we have in both the time and the frequency domain.

In the frequency domain, since we want the signal's support to be at most Θ, a reasonable request is that $G(e^{j\omega})$ acts as a lowpass filter with cutoff frequency $\Theta/2$.

In the time domain, remember that as a result of the convolution operator, equation (1.3) tells us that each sample of the filtered signal can be seen as a weighed average of L neighboring samples of the symbol sequence. We would like that at least every K samples, i.e. when the symbols sequence takes the value the symbol value to be transmitted, this value was left untouched, so we can easily retrieve it at the receiver. In other words, we would like $\boldsymbol{b}[Kn] = \boldsymbol{a}[Kn]$. If this was not the case, every sample value would be influenced by several symbol values and we would be in the presence of *intersymbol interference* (ISI). To see how we can be ISI-free, we can exploit the commutativity of the convolution operator and write

$$\boldsymbol{b}[n] = \sum_{m=-\infty}^{+\infty} \boldsymbol{a}[m]g[n-m]; \tag{1.5}$$

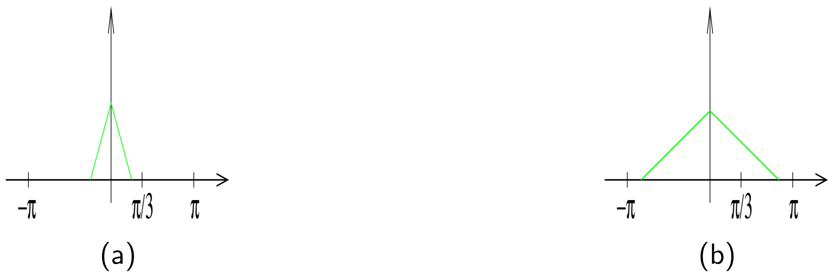

Figure 1.5: Pulse shaper minimal bandwith theorem.

since $\boldsymbol{a}[n] \neq 0$ only for n multiple of K, a sufficient condition on $g[n]$ to achieve $\boldsymbol{b}[Kn] = \boldsymbol{a}[Kn]$ is thus

$$g[Kn] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0. \end{cases} \tag{1.6}$$

This condition has an important consequence, since it implies that the minimum frequency support of $G(e^{j\omega})$ cannot be smaller than $[-\pi/K, \pi/K]$ (Nyquist's theorem). A simple proof of this fact can be outlined using a standard result from multirate signal processing. Assume the spectrum $G(e^{j\omega})$ is nonzero only over $[-\omega_b, \omega_b]$, for $\omega_b < \pi/K$; then we know that we can subsample $g[n]$ by at least a factor of K without aliasing, and the resulting spectrum is going to be the spectrum of the original $g[n]$ stretched by K. In other words, the spectrum of the sequence $g[Kn]$ is going to be nonzero over $[-K\omega_b, K\omega_b]$ and zero elsewhere (see Figure 1.5). But the sequence $g[Kn]$ is by definition simply an impulse in zero and we know that its spectrum is identically one over $[-\pi, \pi]$, therefore we have a contradiction.

Now let's backtrack a little bit: we have just seen that in order to achieve ISI-free transmission at a baud rate of $1/T$, the required support of the baseband signal is *at least* $[-\pi/K, \pi/K]$, with $f_s = K(1/T)$, for a total bandwidth of at least $2\pi/K$. However, the bandwidth is not a design parameter, but a constraint of the channel! Remember that we have to fit the signal to the channel, not the other way around. So, in practice, given a maximum bandwith of Θ, we will have to select a suitable baud rate so that the support of the pulse fits in $[-\Theta/2, \Theta/2]$; in other words, we must have

$$\pi/K < \Theta/2. \tag{1.7}$$

Now, regardless of the values for f_s and K, $\Theta = 2\pi(W/f_s)$ and therefore, as long as we choose $f_s = K(1/T)$, the above relation becomes

$$1/T < W. \tag{1.8}$$

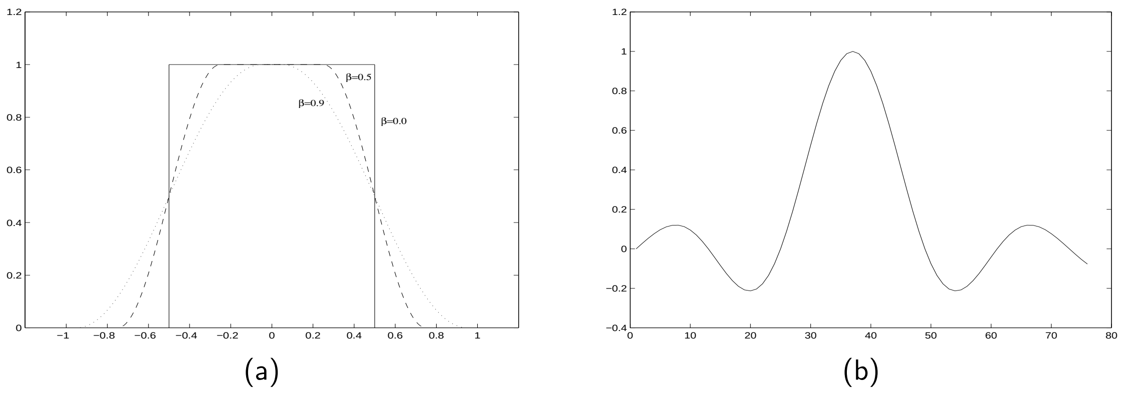

Figure 1.6: Raised cosine pulse shaper: (a) frequency response, (b) impulse response.

This is the first and most important rule of modem design: *the baud rate is always less than the width of the passband.*

So now we know the maximum baud rate; how do we choose a $G(e^{j\omega})$ which satisfies (1.6) and acts as a lowpass with support within $[-\pi/K, \pi/K]$? Initially we could think of choosing $G(e^{j\omega})$ as close as possible to a brickwall filter over $[-\pi/K, \pi/K]$ (note that the brickwall filter is ISI-free). There is however a very important tradeoff in filter design, which we can sum up by saying that the sharper the transition bands are, the longer the relative impulse response. A long impulse response is a response which dies out very slowly in time; this is bad for two reasons: first, since we must use FIR filters for stability reasons, we would need to use a lot of coefficients, which increases the computational load. Secondly, a lot of FIR coefficients imply a long processing delay at both transmitter and receiver, which is something we should strive to reduce, instead. Fortunately there is a type of pulse shape which can help, called the *raised cosine pulse.* We will not go into the details of its derivations, but this pulse, while fulfilling (1.6), has also a “knob”, in the form of a parameter $0 \leq \beta \leq 1$, with which to adjust the sharpness of the transition bands; the resulting minimal bandwidth is increased by a factor of $(1 + \beta)$. In practical systems this knob is set such that the total support is only 10% more than the minimal support $[-\pi/K, \pi/K]$, i.e. $\beta \approx 0.1$. Some of the possible frequency responses are displayed in figure 1.6, together with the typical bell shape of the impulse response. The decay rate of the corresponding impulse responses in the time domain is $1/n^3$ for $\alpha > 0$; this means that in actual implementations the theoretical infinite duration of the pulse can be very well approximated by a small number of samples.

1.2.3 The symbol alphabet

The constellation

So far we haven't said much about the alphabet itself, except that it is composed by $M = 2^m$ symbols. The first piece of news about it, and possibly a surprising one at first, is that the symbols are actually *complex values*. One might initially object that this does not makes sense since there is no such thing as a phisical complex entity – and the telephone line is a very physical thing. On the other hand remember that we are designing a *digital* modulator, which means that all operations are going to be performed on a DSP chip, i.e. a computer. A DSP has no problem handling complex variables, since a complex variable is just a pair of floating point (or fixed point) numbers. So, as long as the final input to the D/A converter is a real discrete time signal, we are fine using complex-valued internal processing.

The M complex symbols $\boldsymbol{\alpha}_i \in A$, $i = 1, \ldots, M$, can be represented graphically as points in the complex plane. The resulting plot is called the modem's *constellation*. One of the basic constellations is the square grid, displayed in Figure 1.7 for $M = 2^4$; in this alphabet the real and imaginary parts of the symbols take on the values of the odd integers between $-(\sqrt{M} - 1)$ and $+(\sqrt{M} - 1)$ (remember that $M = 2^m$ so that $\sqrt{M} = 2^{(m/2)}$. In formulas,

$$\boldsymbol{\alpha} = \alpha_I + j\alpha_Q \tag{1.9}$$

with

$$\alpha_{I,Q} \in \{\pm 1, \pm 3, \pm 5, \ldots, \pm(\sqrt{M} - 1)\}; \tag{1.10}$$

α_I and α_Q, the real and imaginary parts of a symbol value, are commonly called the *In-phase* and *Quadrature* components. The complex sequence $\boldsymbol{a}[n]$ at the ouptut of the upsampler is filtered via the pushe shaping filter (digital filters work beautifully for complex signals too) to yield the complex sequence $\boldsymbol{b}[n]$. If we need to, we can always separate real and imaginary parts of the complex signal and write

$$\boldsymbol{a}[n] = a_I[n] + ja_Q[n] \tag{1.11}$$

and, via the linearity of the filtering operation,

$$\boldsymbol{b}[n] = g[n] * a_I[n] + j(g[n] * a_Q[n]) = b_I[n] + jb_Q[n]. \tag{1.12}$$

The fundamental property of a constellation is its *average power*. It is defined the variance of a uniformly distributed random variable with probability $1/M$ taking values on the alphabet symbols. The variance of a random variable can be interpreted as the average energy of a symbol, and it is important because the power of the transmitted

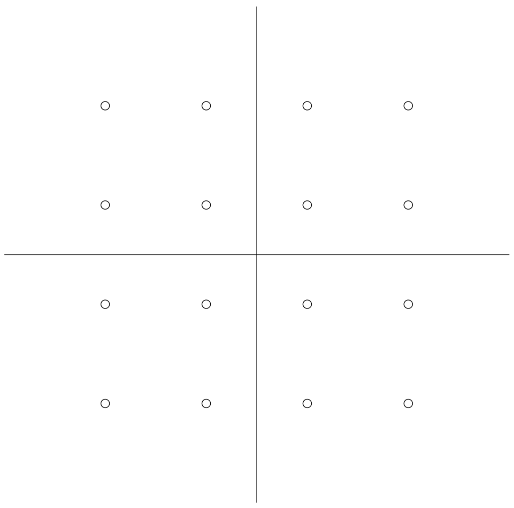

Figure 1.7: Square constellation, $M = 16$.

signal is proportional to it as shown in (1.2). Define then a complex random variable $\boldsymbol{\alpha}$ taking values over the square constellation alphabet defined in (1.10) and uniformly distributed:

$$P[\boldsymbol{\alpha} = \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_i \in A] = 1/M; \tag{1.13}$$

it is a simple exercise to show that the mean of $\boldsymbol{\alpha}$ is zero and that its variance is

$$E[\boldsymbol{\alpha}^2] = \sigma_A^2 = \frac{2}{3}(M-1). \tag{1.14}$$

Alphabet size and SNR

To determine the maximum number of symbols in the constellation we have to take into account the third constraint of modulated transmission, the power constraint. So far we have considered the spectral properties of the transmitted signal as a random process. Now we will concentrate on the single symbols one at a time to see what is the influence of noise on the system and how this affects our ability to send data. We will see in the notes about demodulation that the receiver attempts to reconstruct a local copy of the symbol sequence $\boldsymbol{a}_i$ as it comes out of the mapper. The transmission however introduces noise in the process so that, without going into too many details, the copy of the symbol sequence at the receiver will be

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i + \boldsymbol{n}_i \tag{1.15}$$

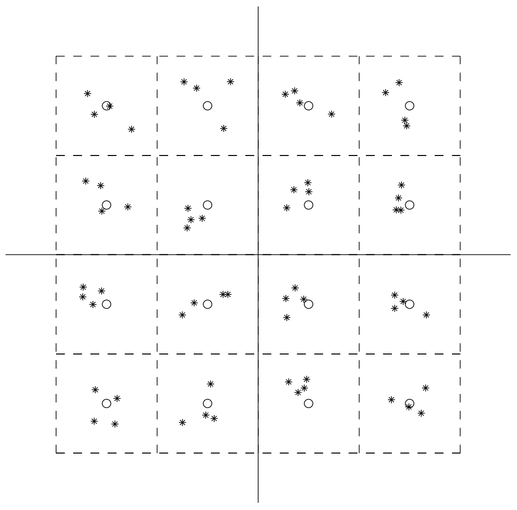

Figure 1.8: Square constellation with the decoding boundaries.

where $\boldsymbol{n}$ is a sampled complex random process (the noise samples) whose statistics are dependent on the particular characteristics of the channel. The simplest technique to try to recover the original values $\boldsymbol{a}_i$ from $\hat{\boldsymbol{a}}_i$ is to associate to $\hat{\boldsymbol{a}}_i$ the alphabet symbol $\boldsymbol{\alpha} \in A$ which is closest to $\hat{\boldsymbol{a}}_i$ in the Euclidean metric (*hard decoding*). Visually, this can be illustrated by superimposing a grid to the costellation points as in Figure 1.8; the values for the noisy $\hat{\boldsymbol{a}}_i$ are also plotted with stars, and all the values that fall in a square are decoded as the value for the center of the square. An error will occur if either the real or imaginary part of $\boldsymbol{n}_i$ is greater than half the distance between the minimum distance between two alphabet values; for the alphabet in (1.10) this minimum distance is 2.

The minimum distance between symbols can be increased without altering the shape of the constellation by multiplying the values of all symbols by a *gain factor* G_0; this is the digital equivalent to a linear amplifier boosting the level of the modem's output signal. The minimum distance becomes then $2G_0$ and by increasing the gain factor we can therefore reduce the probability of error. In so doing we are also increasing the average power of the transmitted signal which, however, can be done only up to the power limit imposed by the channel.

We can substantiate these arguments in the case of transmission over an ideal channel which introduces white Gaussian noise only, and we will illustrate the main intuitive points. If the noise is white and Gaussian, it is going to affect equally and independently both real and imaginary parts of $\boldsymbol{a}_i$; the probability density function (pdf) of each complex noise sample can therefore be described by a two-dimensional Gaussian pdf with zero mean and

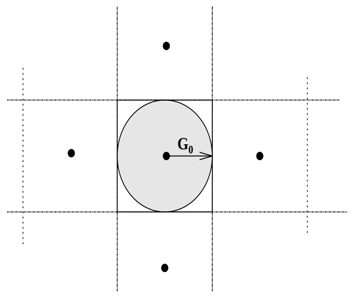

Figure 1.9: Correct decision region: optimal region in the nearest-neighbor sense (the square) and its circular approximation.

variance equal to $\sigma^2 = N_0/2$

$$p_n(\boldsymbol{n}) = \frac{1}{\pi N_0} e^{-\frac{|\boldsymbol{n}|^2}{N_0}}. \tag{1.16}$$

For the square alphabet as in (1.10), 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; this probability can be approximated well enough by integrating the noise pdf over a circle C of radius G_0 (see figure 1.9) so that the approximate error probabilty is

$$p_e = 1 - \int\!\!\int_C p_n(\boldsymbol{n})\, d\boldsymbol{n} = e^{-\frac{G_0^2}{N_0}} \tag{1.17}$$

Let's see how this affects the amount of data we can transmit. Fisrt of all, we must settle on a *reliabilty figure* which represents the tolerates probability of error of the system; in practical modems, for instance, a probability of error on the order of $p_e \geq 10^{-6}$ is usually the norm. To achieve this probability of error, equation (1.17) tells us that half the minimum distance between symbols has to be at least

$$G_0^2 \geq -N_0 \ln(p_e); \tag{1.18}$$

this gives us the minimum amplification gain we must use for the transmitted signal. On the other hand, assume that the maximum *average* power we can transmit over the channel is $P_{\max}$; the average power P of the transmitted signal is basically the average power of the symbol sequence scaled by the gain factor (squared since it's a power measure):

$$P = G_0^2 \sigma_A^2. \tag{1.19}$$

We must ensure that $P \leq P_{\max}$, and therefore, by substituing for G_0, we have

$$\sigma_A^2 \leq \frac{P_{\max}}{-\ln(p_e)N_0}. \tag{1.20}$$

Now, N_0 is the averge power of a noise sample (real and imaginary part) and therefore $(P_{\max}/N_0)$ is the average signal to noise ratio (SNR) of the channel. Since $\sigma_A^2 = (2/3)(M-1)$, we have that for a square constellation it must be

$$M \leq (\frac{3}{2})(\frac{\text{SNR}}{-\ln(p_e)}) + 1 \tag{1.21}$$

and the total number of bits per second will be

$$R \leq (1/T)\,\log_2(M). \tag{1.22}$$

Modulation

The method for building the baseband signal we have explained so far is called Quadrature Amplitude Modulation (QAM) and is completed by a last building block, the modulator, which takes the amplified baseband signal $G_0\boldsymbol{b}[n]$ and multiplies it by a complex sinusoidal signal at a frequency ω_c (the carrier) to create the passband signal. We know that the baseband signal is a filtered random process whose power spectrum is given in (1.4), so we should use results from random signal processing to derive the spectrum of the passband signal. To make things simpler, however, from now on we will just consider $\boldsymbol{b}[n]$ as a *deterministic* signal with Fourier transform $B(e^{j\omega})$. This is only partially cheating since a modem always transmits a finite-length signal (or else you would have to pay an astronomical phone bill!) and we can always take Fourier transforms of finite-length sequences.

The modulated signal is

$$s[n] = G_0\boldsymbol{b}[n]e^{j\omega_c n}; \tag{1.23}$$

modulation allows us to shift in frequency the support of the signal so as to fit it into the slot offered by the channel since, in the frequency domain, the complex version of the modulation theorem states:

$$S(e^{j\omega}) = G_0 B(e^{j(\omega-\omega_0)}). \tag{1.24}$$

The signal $\boldsymbol{s}[n]$ is still a complex signal, and therefore we can't transmit it as it is. What we actually feed to the D/A is, instead,

$$c[n] = G_0\text{Re}\{\boldsymbol{b}[n]e^{j\omega_c n}\}. \tag{1.25}$$

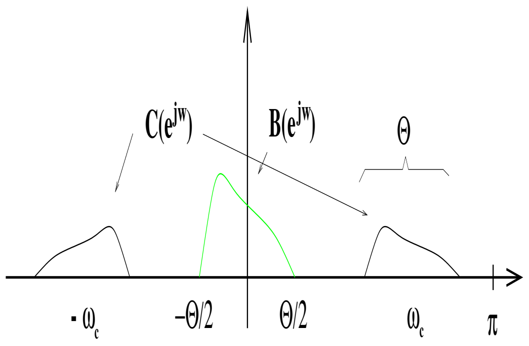

Figure 1.10: Spectral relationships between baseband and passband signals.

Now, remember that for any complex signal $x[n]$ we can write $\text{Re}\{x[n]\} = (1/2)(x[n] + x^*[n])$, where the star denotes the complex conjugate; also remember that the Fourier transform of $x^*[n]$ is $X^*(e^{-j\omega})$. With this, the spectrum of $c[n]$ is

$$\begin{aligned} C(e^{j\omega}) &= \frac{1}{2}[S(e^{j\omega}) + S^*(e^{-j\omega})] \\ &= \frac{G_0}{2}[B(e^{j(\omega-\omega_c)}) + B^*(e^{j(-\omega-\omega_c)})]. \end{aligned} \tag{1.26}$$

Since $B(e^{j\omega})$ has support $[-\Theta/2, \Theta/2]$, $C(e^{j\omega})$ has support $\{[-\omega_c - \Theta/2, -\omega_c + \Theta/2] \cup [\omega_c - \Theta/2, \omega_c + \Theta/2]\}$; the spectral relationship between baseband and passband signals is depicted in Figure 1.10.

1.3 The Final Recipe

Ok, let' review the rules of thumb in designing a modem, and then let's apply them to a real case scenario.

First of all, we need to know the details of the passband offered by the channel in terms of it minimum frequency $f_{\min}$ and $f_{\max}$ in Hertz and its width $W = f_{\max} - f_{\min}$. This will automatically give a bound for the baud rate $(1/T) < W$. We will choose an operative baud rate a little smaller than W to leave room for the raised cosine expansion; this expansion is usually about 10-20%. We will then choose an internal sampling rate f_s for our transmitter such that $f_s \geq 2f_{\max}$ and $f_s = K(1/T)$ for some integer K; usually K is about 3 or 4 and is dictated by the speed of the DSP chip: the larger K is, the better we can design our internal filters, but also the larger the number of operations per sample. Finally, we need to determine the modulation frequency; the typical value for that is the center frequency of the passband, i.e.:

$$\omega_c = 2\pi \frac{f_{\text{max}} + f_{\text{min}}}{2f_s}. \tag{1.27}$$

The amplifyer's gain and the maximum number of symbols in the constellation is then determined from the average SNR using equation (1.21) and, with that, we're all set. Easy, isn't it.

For example...

As a practical example assume a telephone line where $f_{\text{min}} = 450$ Hz and $f_{\text{max}} = 3150$ Hz, with a SNR of about 22 dB. The baud rate will have to be less than 2700 symbols per second, since $W = 2700$ Hz, and say we choose the value 2400. We also choose $K = 3$ to keep computational costs down, so that a good internal sampling frequency satisfying all requirements is $f_s = 7200$. With this, the maximum support of the baseband signal is $\Theta = 2\pi(2700/7200) = 0.75\,\pi$. We then design a raised cosine filter with cutoff frequency $\pi/3$ and 12.5% expansion factor, i.e. $\beta = 0.125$; the resulting support of the baseband signal is therefore $(1+\beta)(2\pi/K) = 0.75\,\pi$, which is just OK. The carrier frequency will be $\pi/2$, corresponding to a analog frequency of 1800 Hz. For a probability of error $p_e < 10^{-6}$ we find that the maximum value for M is approximately 18. For a square constellation, M should be a perfect square and so we choose $M = 16$. We have just designed a modem which, over the given telephone channel, can transmit 9600 bits per second; this is actually one of the possible operating modes of the V.32 modem standard.

Part 2

Demodulation

The signal created at the modulator is converted to a continuous time signal $c(t)$ by a D/A converter operating at a sampling frequency f_s and sent over the telephone channel. With reasonably good approximation the channel behaves like a linear filter $D(e^{j\omega})$ and also introduces a certain amount of additive noise so that the signal appearing at the receiver's input looks like

$$v(t) = d(t) * c(t - t_p) + \nu(t) \tag{2.1}$$

where t_p is the *propagation delay*, dependent on the distance between transmitter and receiver, $d(t)$ is the equivalent impulse response of the channel and $n(t)$ is the noise. The first thing the digital receiver does is sampling the incoming signal (see Figure 2.1); for convenience we will assume that the sampling rate used by the receiver matches the internal sampling frequency f_s of the transmitter so that the signal processed by the receiver is simply

$$v[n] = v((1/f_s)n - t_p) \tag{2.2}$$

A fundamental building block of any modem is an *adaptive equalizer* whose task is to estimate the distortion introduced by the channel in order to eliminate it. Since we assume that the channel acts as a linear filter (with frequency response $D(e^{j\omega})$ in the digital domain) the equalizer's job is to estimate $D(e^{j\omega})$, compute the inverse filter $D^{-1}(e^{j\omega})$ and filter the incoming signal with it. In these notes we will not study how the equalizer works, but we will simply assume that there is one and that it achieves its goal, so that the processed signal after the equalizer is distortion-free. Also, we assume that the adaptive equalizer amplifies or attenuates the incoming signal to a normalized level which we can assume without loss of generality to be unity. In other words, we are undoing the gain amplification by G_0 introduced at the transmitter so that the signal to be demodulated is

$$v[n] = c((1/f_s)n - t_p) + \nu[n]. \tag{2.3}$$

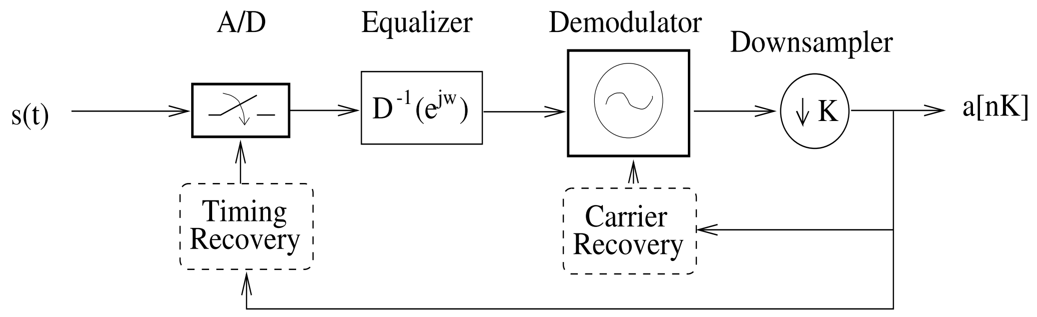

Figure 2.1: Demodulator.

2.1 Basic Demodulation

To study the basics of demodulation we will assume at first that modulator and demodulator are connected *back to back*, i.e., that there is no channel between them and therefore no need to convert the signal to and from continuous time. This means that the "received signal has no delay and no added noise or, in other words, that it is identical to the transmitted signal $c[n]$:

$$r[n] = c[n] = \mathrm{Re}\{\boldsymbol{b}[n]e^{j\omega_c n}\}; \tag{2.4}$$

2.1.1 Demodulation

The function of a demodulator is to remove the modulation, obviously, and the first thing to do is to strip the carrier off the received signal. Now, if we had the full complex signal $s[n] = \boldsymbol{b}[n]e^{j\omega_c n}$ rather than just its real part, carrier removal would be very easy: just multiply $s[n]$ by $e^{-j\omega_c n}$. The question is wether we can reconstruct the complex signal from its real part.

Consider the sequences $\boldsymbol{b}[n]$ and $e^{j\omega_c n}$; we can separate their real and imaginary parts and write

$$s[n] = (b_I[n] + jb_Q[n])(\mathrm{Re}\{e^{j\omega_c n}\} + j\mathrm{Im}\{e^{j\omega_c n}\}). \tag{2.5}$$

If we look in turn at the real and imaginary part of $s[n]$ we have

$$s_r[n] = \mathrm{Re}\{s[n]\} = b_I[n]\,\mathrm{Re}\{e^{j\omega_c n}\} - b_Q[n]\,\mathrm{Im}\{e^{j\omega_c n}\} \tag{2.6}$$

$$s_i[n] = \mathrm{Im}\{s[n]\} = b_I[n]\,\mathrm{Im}\{e^{j\omega_c n}\} + b_Q[n]\,\mathrm{Re}\{e^{j\omega_c n}\}; \tag{2.7}$$

it is obviously $s_r[n] = c[n]$. Let's now have a look at the above relations in the frequency domain. To do that just remember three things:

- for any complex number z it is

$$\begin{aligned} \mathrm{Re}\{z\} &= (1/2)(z+z^*) \\ \mathrm{Im}\{z\} &= (-j/2)(z-z^*) \end{aligned}$$

  where the star denotes the complex conjugate;

- for a complex exponential
$$(e^{j\alpha})^* = e^{-j\alpha};$$

- If $X(e^{j\omega})$ is the Fourier transform of $x[n]$, then the Fourier transform of $x[n]e^{j\alpha n}$ is simply $X(e^{j(\omega-\alpha)})$ (modulation theorem).

With this it is easy to see that

$$\begin{aligned} S_r(e^{j\omega}) &= \frac{1}{2}[B_I(e^{j(\omega-\omega_c)}) + B_I(e^{j(\omega+\omega_c)}) + jB_Q(e^{j(\omega-\omega_c)}) - jB_Q(e^{j(\omega+\omega_c)})] \\ S_i(e^{j\omega}) &= \frac{1}{2}[-jB_I(e^{j(\omega-\omega_c)}) + jB_I(e^{j(\omega+\omega_c)}) + B_Q(e^{j(\omega-\omega_c)}) + B_Q(e^{j(\omega+\omega_c)})]. \end{aligned}$$

Now, $B_{\{I,Q\}}(e^{j\omega})$ are the baseband spectra corresponding to the real and imaginary part of the symbol sequence $\boldsymbol{a}_k$, so their translates in frequency at $\pm\omega_c$ do not overlap (see Figure 1.10). In this case, it is easy to verify that we can write

$$S_i(e^{j\omega}) = H(e^{j\omega})S_r(e^{j\omega}) \tag{2.8}$$

with

$$H(e^{j\omega}) = \begin{cases} -j & 0 \le \omega < \pi \\ j & -\pi \le \omega < 0 \end{cases} \tag{2.9}$$

The filter $H(e^{j\omega})$ is called a *Hilbert transformer* and its impulse response is

$$h[n] = \begin{cases} \dfrac{2\sin^2(\pi n/2)}{\pi n} & n \neq 0 \\ 0 & n = 0 \end{cases} \tag{2.10}$$

The Hilbert transformer is an ideal noncausal filter, just like the rect and sinc filters and so it is impossible to implement it exactly; however, good FIR approximation can be obtained using most of standard FIR design methods. In particular, since the impulse response is antisymmetric ($h[-n] = -h[n]$) and $h[0] = 0$, the resulting causal approximation $\tilde{h}[n]$ will be a type III FIR of length $2L+1$, where L is usually on the order of 20 taps. The real and imaginary parts of $s[n]$ can thus be expressed as

$$s_r[n] = c[n-L] \tag{2.11}$$

$$s_i[n] = \sum_{m=0}^{2L} \tilde{h}[m]c[n-m] \tag{2.12}$$

where the delay of L samples in the real part compensates for the processing delay inroduced by the causal FIR approximation. Carrier stripping is therefore accomplished by multiplying $s_r[n] + js_i[n]$ obtained above by a locally generated complex sinusoid $e^{-j\omega_c n}$ to obtain a local sequence $\hat{\boldsymbol{b}}[n]$:

$$\hat{\boldsymbol{b}}[n] = (s_r[n] + js_i[n])e^{-j\omega_c n} \tag{2.13}$$

As a final demodulation step, we need to retrieve and decode the original sequence of complex symbols $\boldsymbol{a}_i$. Remember that, in the modulator, $\boldsymbol{b}[n]$ is the convolution of the sequence $\boldsymbol{a}[n]$ at the output of the K-time upsampler with the pulse shape $g[n]$ and that we have designed the pulse shape so that we can recover the values $\boldsymbol{a}[n]$ at times multiples of K simply by downsampling. The decoded symbol sequence can therefore be obtained as

$$\hat{\boldsymbol{a}}_i = \hat{\boldsymbol{b}}[iK] \tag{2.14}$$

2.1.2 Noise and Slicing

Although we haven't worked out the details in full, intuitively any noise added to the signal before demodulation will leak through the demodulation process and will affect the values of the decoded symbol sequence. Even in the "back to back" configuration introduced above, numerical errors in the digital demodulation process together with the imprecision errors due to our FIR approximations of ideal filters cause a degradation in the decoded signal which looks just as if noise had been added. In a real transmission case, this numerical noise just adds up to the channel noise level. In the end, the received symbol sequence can be written as

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i + \boldsymbol{n}_i \tag{2.15}$$

where $\boldsymbol{n}_i$ is a sequence of complex noise samples moving the value of the received symbol away from its starting value. Graphically, for the constellation of Figure 1.7, the received symbols for different amounts of additive noise (different SNRs) will look as in Figure 2.2. A simple way to obtain an estimate of the transmitted sequence $\boldsymbol{a}_i$ is via *hard slicing*:

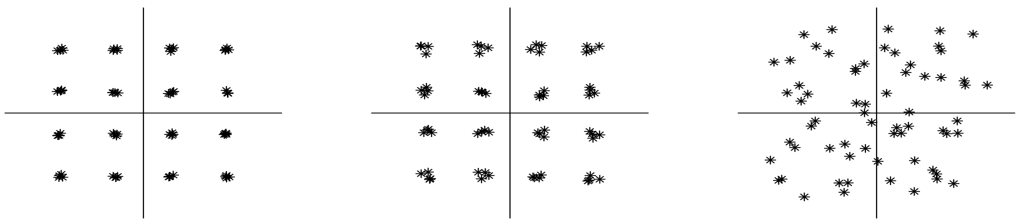

Figure 2.2: Received symbols for increasing noise levels.

simply associate to each $\hat{\boldsymbol{a}}_i$ the alphabet element $\boldsymbol{\alpha} \in A$ which is closest to $\hat{\boldsymbol{a}}_i$. In other words, the output symbols $\boldsymbol{a}_{i,\text{decoded}}$ are defined as

$$\boldsymbol{a}_{i,\text{decoded}} = \arg\min_{\boldsymbol{\alpha}\in A} |\hat{\boldsymbol{a}}_i - \boldsymbol{\alpha}|^2; \tag{2.16}$$

for the square constellation this becomes simply

$$\boldsymbol{a}_{i,\text{decoded}} = 2\lfloor \hat{\boldsymbol{a}}_i/2 \rfloor + 1. \tag{2.17}$$

As we stated previously, errors will occur if the instantaneous noise value is larger (in magnitude) than half the minimum distance between constellation points.

2.2 Advanced Topics: Synchronization

Let's now abandon the unrealistic case of back to back transmission and examine one of the major problems of modem design, synchronization. Correct demodulation requires the perfect knowledge of two fundamental values, the baud rate and the carrier frequency. The correct carrier frequency f_c (in Hertz) is necessary to convert the passband signal to a baseband signal via carrier stripping as we just saw before. The baud rate $1/T$ (or, equivalently, the symbol period T) is necessary to know where in the baseband signal to look for the symbol values; as we saw, in digital modems the internal sampling frequency is an integer multiple of the symbol frequency, $f_sT = K$, and so we need to know the right baud rate in order to correctly downsample the demodulated singnal as in (2.14). The nominal values of these frequencies are obviously known exactly, since they are part of the technical specification of a modem; so, in theory, we can just choose a sampling frequency f_s multiple of $1/T$ and use $\omega_c = 2\pi(f_c/f_s)$, $K = f_sT$. In practice, however, there are two main factors which complicate things: the imprecision of the sampler's clock and the propagation delay introduced by the channel. These two problems are interconnected; remember that the the signal to be demodulated is

$$v[n] = c((1/f_s)n - t_p) + \nu[n], \tag{2.18}$$

where t_p is the propagation delay. We can always express this delay as an integer number of symbol intervals plus a fractional delay ranging from $-T/2$ to $+T/2$, where $1/T$ is the baud rate:

$$t_p = DT + \tau; \tag{2.19}$$

the integer D is called the *bulk delay.* The reason we can do this is because modems send appropriate timing signals back and forth before transmitting real data to estimate the bulk delay. Once we know D, the problem reduces to estimating τ, which is much smaller. The delay affects both the baseband signal and the carrier: denote by $\boldsymbol{b}_\tau[n]$ the baseband signal $\boldsymbol{b}[n]$ delayed by the noninteger delay τ; the received, sampled, and equalized signal is therefore:

$$v[n] = \mathrm{Re}\{\boldsymbol{b}_\tau[n]e^{j(\omega_c n+\theta)}\}, \tag{2.20}$$

where θ is the phase offset in the carrier introduced by the propagation delay. In theory $\theta = \omega_c\tau$, so by estimating τ we could get an estimate for θ as well; in practice this is not done since the precision required on an estimate for θ is much greater than that required for τ. Different circuits with different strategies are used in the two cases, and therefore we will consider θ and τ independently.

Carrier Recovery

Carrier recovery is the modem functionality by which any phase offset in the received carrier is estimated and compensated for. A phase offset in the carrier frequency causes a rotation of the constellation points in the complex plane. This is easily seen if we apply the carrier stripping phases described above to the signal

$$c[n] = \mathrm{Re}\{\boldsymbol{b}[n]e^{j(\omega_c n+\theta)}\}; \tag{2.21}$$

after downsampling we obtain the sequence (neglecting noise)

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i e^{j\theta} \tag{2.22}$$

where the last term is clearly a rotation factor. Visually, the decoded constellation looks like in Figure 2.3, where $\theta = \pi/20 = 9'$. In the same figure, the slicing grid is also plotted: it is clear that in the rotated constellation, some points are much closer to the decision boundaries than they should. A smaller amount of noise is therefore sufficient to cause

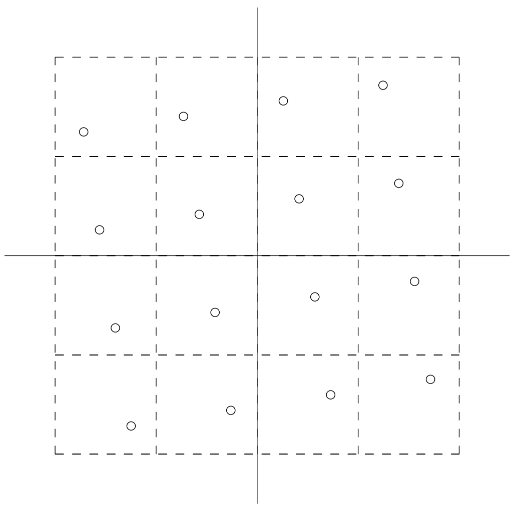

Figure 2.3: Effect of a phase offset on the received symbols.

slicing errors. An even worse situation happens when the local carrier frequency is slightly different from the transmitter's carrier frequency; in this case the phase offset changes over time and the points in the constellation start to rotate with an angular speed equal to the difference between transmitter's and receiver's carrier frequencies. In both cases, data transmission becomes highly unreliable; carrier recovery is then a very important part of a modem design.

The most common technique for QAM carrier recovery over well-behaved channels is a *decision directed loop*; this works when the overall SNR is sufficiently high, and this is the case for telephone channels. Consider the picture in Figure 2.4; the received symbol is the rotated $\hat{\boldsymbol{a}}$ which is sufficiently close to the correct value $\boldsymbol{a}$ to be decoded correctly. In the z plane, consider the two vectors $\vec{a}_1$ and $\vec{a}_2$, from the origin to $\hat{\boldsymbol{a}}$ and $\boldsymbol{a}$ respectively. Basic linear algebra tells us that the magnitude of their vector product can be expressed in terms of their components:

$$|\vec{a}_1 \times \vec{a}_2| = \mathrm{Re}\{\hat{\boldsymbol{a}}\}\,\mathrm{Im}\{\boldsymbol{a}\} - \mathrm{Im}\{\hat{\boldsymbol{a}}\}\,\mathrm{Re}\{\boldsymbol{a}\}; \tag{2.23}$$

or in terms of their magnitudes and their relative angle

$$|\vec{a}_1 \times \vec{a}_2| = |\vec{a}_1||\vec{a}_2|\sin\theta. \tag{2.24}$$

We can therefore obtain an estimate for the phase offset by calculating

$$\sin\theta = \frac{\mathrm{Re}\{\hat{\boldsymbol{a}}\}\mathrm{Im}\{\boldsymbol{a}\} - \mathrm{Im}\{\hat{\boldsymbol{a}}\}\mathrm{Re}\{\boldsymbol{a}\}}{|\vec{a}_1||\vec{a}_2|} \tag{2.25}$$

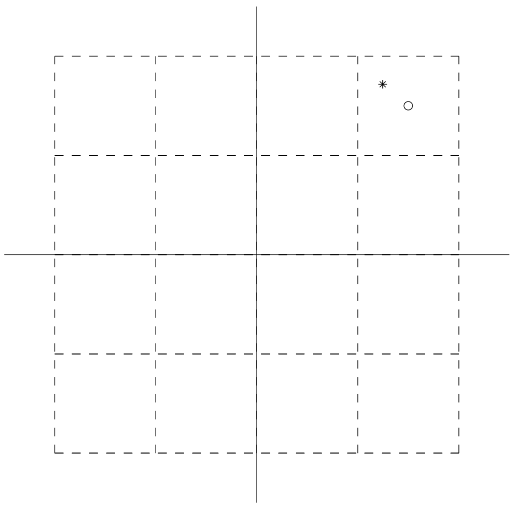

Figure 2.4: Estimation of the phase offset.

and by assuming that the phase offset is small enough to allow the use of the small angle approximation for the sine:

$$\sin\theta \approx \theta. \tag{2.26}$$

This is the correction phase we must apply to the internal carrier.

As we mentioned before, due to component tolerances and other effects the local carrier frequency might be slightly different from the nominal value ω_c. This is equivalent to a phase offset varing slowly over time, so that the offset estimation process must be performed continuously to *lock* the internal carrier frequency to the received signal. To see how this is done, consider how the sinusoidal carrier is generated digitally; the DSP software will possess a complex exponential function algorithm which computes $y = e^{jw}$; the carrier is generated by initializing the variable w to zero and, at each sample interval, by computing $y = e^{jw}$ and by incrementing w by the right amount for the next time. Ideally, at each step, w is incremented by ω_c, the digital frequency so that the algorithm looks like this:

- Initialization:

  $\varphi = \omega_c;$

  $w = 0;$

  where φ is the local estimate of the carrier digital frequency.

- At each step n:

```
    Compute $e^{jw}$
    $w \leftarrow w + \varphi$
```

If we detect a phase offset, however, we must do two things: fisrt we must compensate for it in order to rotate back the constellation in place; secondly, we must take into account the fact that the phase offset might be due to a carrier frequency mismatch, so we must correct the estimate of the sampling frequency in the correct way. A positive instantaneous phase offset means our carrier is too fast, a negative offset that our carrier is too slow. To compensate for possible drifts, however, the carrier generation algorithm is modified as follows

- Initialization;

```
    $\varphi = \omega_c$;
    $\psi = 0$;
    $w = 0$;
```

  where ψ is a correction factor.

- At each step n:

```
    Compute $e^{jw}$
    With $e^{jw}$, demodulate and compute phase offset $\theta$
    $\varphi \leftarrow \varphi - \theta$
    $w \leftarrow w + \varphi - \theta$
```

where the last two lines mean that we assume the local carrier is too fast (or too slow) and we change its frequency accordingly; and that we compensate for the instantaneous phase drift.

2.2.1 A Digital PLL

Let's now consider what happens if the initial A/D converter of the digital demodulator exhibits drifts or component tolerances. Consider the simple system shown in Figure 2.5: an A/D converter at a nominal frequency f_s with a continuous time sinusoidal input $x(t)$ at frequency $2\pi f_0$:

$$x(t) = \sin(2\pi f_0 t); \tag{2.27}$$

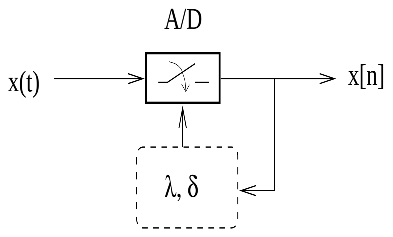

Figure 2.5: Simple timing recovery system.

this signal is depicted in Figure 2.6 with a solid line. The output of the A/D converter is the discrete time signal

$$x[n] = \sin(\omega_0 n) \tag{2.28}$$

with

$$\omega_0 = 2\pi \frac{f_0}{f_s}. \tag{2.29}$$

Assume we design the A/D so that its sampling frequency is $f_s = f_0$; if the A/D works as planned, the digital frequency of $x[n]$ will be exactly $\omega_0 = 2\pi$ and therefore

$$x[n] = \sin(2\pi n) = 0 \quad \forall n; \tag{2.30}$$

the sampled signal is plotted in Figure 2.6 with circles. Assume however that due to component tolerances or temperature drifts the frequency of the A/D moves away from its nominal value f_0. If $f_s < f_0$ it means we are sampling slower than we should and the the values for $x[n]$ will be the ones marked with a cross in Figure 2.7-(a); conversely, if $f_s > f_0$ then we're sampling too fast and $x[n]$ will take on the values marked with a square in Figure 2.7-(b). Note that, if the sampling frequency is almost right, the A/D's output is positive when we're sampling slightly too fast and negative if we're sampling slightly too slow. We would like to find a system to *lock* the sampler at the right speed; since we know the value of the nominal frequency f_0 this should be possible and the idea is to use the value of the samples to estimate the deviation from the right sampling frequency. The following derivation follows closely the mechanism implemented for carrier recovery.

Assume we have a mechanism to change the sampler's speed; it is convenient to express this in terms of the sampling *time* $t[n]$ which is the sequence of time instant at which the samples are taken. With this notation, the sampled signal can be expressed as

$$x[n] = x(2\pi f_0 t[n]). \tag{2.31}$$

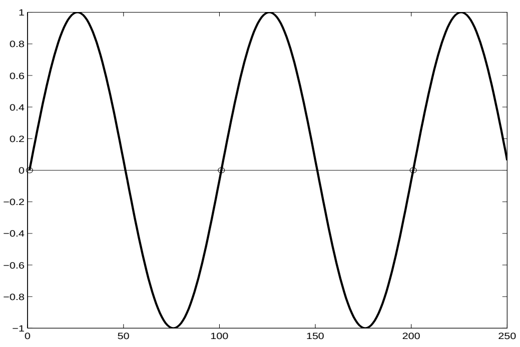

Figure 2.6: Continuous time sine wave and samples at every period.

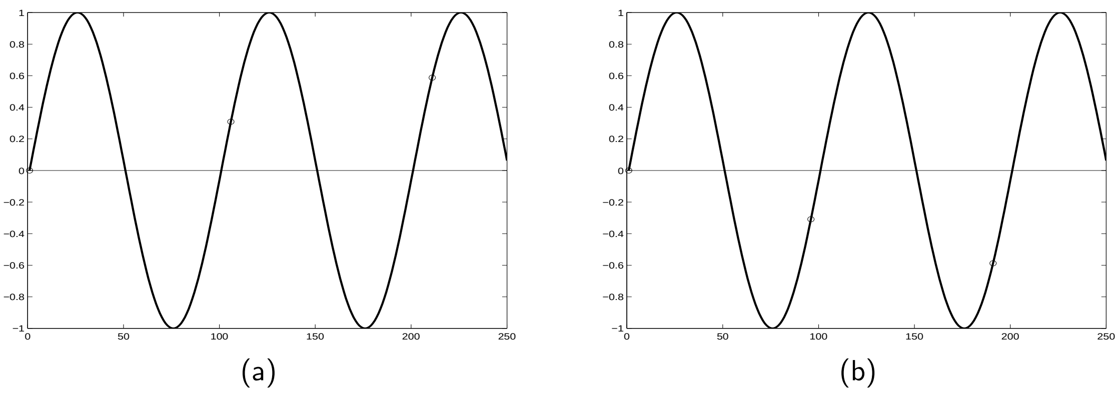

Figure 2.7: Since wave sampled (a) too slow; (b) too fast.

The sampler computes the each sampling period as

$$t[n] = t[n] + \lambda - \delta \tag{2.32}$$

where λ is the current estimate of the right period and δ is a correction term. Assume that, initially, $\lambda = 1/f_0$, the right value, and $\delta = 0$. While this goes on, it will be

$$x[n] = \sin(2\pi f_0 n\lambda) = \sin(2\pi n) = 0. \tag{2.33}$$

Now imagine that at time $n-1$ the sampler's speed starts to move slightly away from its right value, for example imagine that λ starts to increase. At time n we will therefore have

$$x[n] = \sin(2\pi f_0\,(1/f_0)(n-1) + 2\pi f_0\lambda) = \sin(2\pi f_0\lambda) \tag{2.34}$$

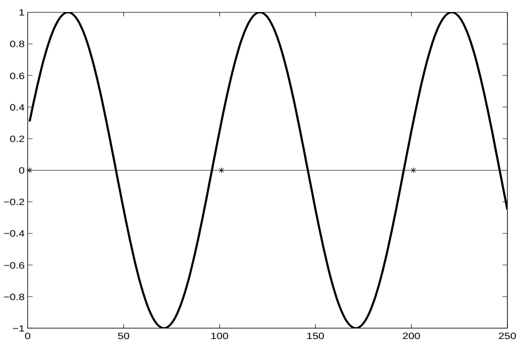
Figure 2.8: Delayed sine wave.

and, since we can safely assume that the drift is very small, we will notice this speed increase by the fact that $x[n] > 0$. Call γ the small extra time so that $\lambda = (1/f_0) + \gamma$. Since this time is very small (slow drift), we can use the small angle approximation $\sin(x) \approx x$ (with x in radians) so that

$$x[n] = \sin(2\pi f_0((1/f_0) + \gamma)) = \sin(2\pi f_0\gamma) \approx f_0\gamma. \tag{2.35}$$

To get back on track we now need to do two things: first we need to reduce the sampling period estimate to avoid repeating the mistake at the next sampling instant; second, we need to compensate the next sampling interval for the delay we already accumulated with respect to the correct sampling time. In formulas, the new sampling time will be computed as:

Estimate γ from $x[n]$, $\gamma \approx x[n]/f_0$;

$\lambda \leftarrow \lambda - \gamma$;

$$t[n+1] = (t[n] - \gamma) + \lambda; \tag{2.36}$$

This update obviously works also if the sampler's speed slows down, and is repeated at each sampling interval: look at x[n], update λ, compute $t[n+1]$. In practice, however, for the updates we use $\gamma = \alpha x[n]$ where $\alpha \ll 1$ is a damping factor which prevents instabilities and is a compromise between speed of response of the system and adjustment accuracy. If α is very small, we will be able to adjust $t[n]$ very precisely, but it will take a longer time since we move little by little; conversely, if α is large, we will move faster, but the sampling time will be oscillating more around its right value.

This kind of system is a particular realization of a Phase-Locked Loop (PLL), in the sense that it locks the sampling instants of the A/D to the instants in which the

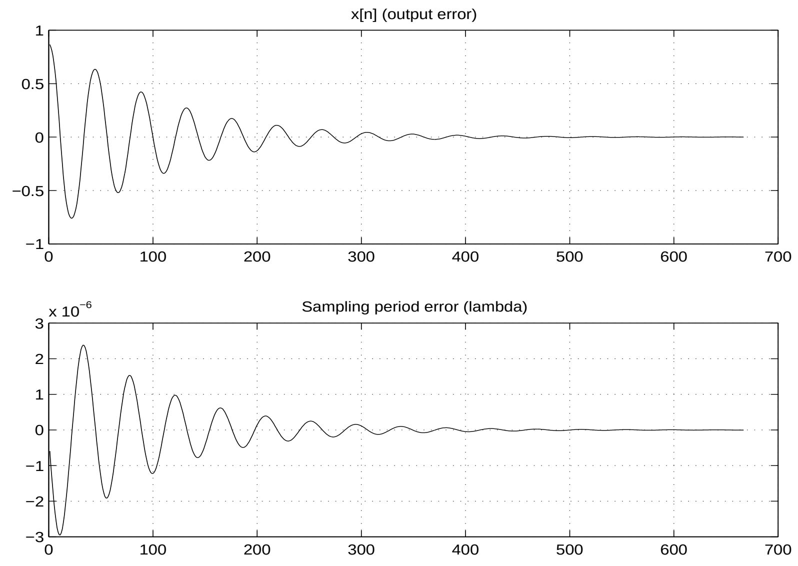

Figure 2.9: Timing recovery in action: output error (top panel); sampling period errror (bottom panel).

instantaneous phase of the input sinusoid is zero. The system works in a feedback fashion, since it brings back an error signal (the downsampler output) to the system's input (the A/D).

The system works also when it comes to estimating a delay τ in the input signal, and we will just see the intuition behind this. A typical case is shown in Figure 2.8 for $\tau < 0$; this delay will cause the downsampler's output to be positive, so the system will start reducing the sampling estimate according to (2.36) even if the original sampling period is correct. By sampling slower than we should, we are actually reducing the delay; at one point, however, we will also find ourselves lagging behind the true sampling instants because the sampling period has become smaller. So the system will start increasing λ again, and its value will oscillate around $1/f_0$ by means of several adjustments until the delay is compensated for and the sampling period is the correct one. Figure 2.9 shows a typical plot of $x[n]$ and $\lambda - 1/f_0$ for an initial delay $\tau = 1/4f_0$ and an initial estimate 1% faster $\lambda = 1.1/f_0$.

Timing recovery

By timing recovery we designate the operation by which the modem locks its input sampler to $f_s = 1/KT$, K times the baud rate of the incoming signal. This operation is necessay since, in order to extract the transmitted symbol values, we need to take baseband signal values at exactly the right instants, i.e. at exact multiples of T. If the input sampler works at $f_s = 1/KT$, this is accomplished simply by a K-times downsampler, as we have seen. If the incoming signal contained a sinusoid at a frequency $2\pi/T$, where T is the baud rate, we could downsample this sinusoid by K, which would move the digital frequency of the sinusoid to $2\pi/KT$ and then we could use the PLL described above to perform this operation. The fact is, the transmitter does not send an explicit sinusoidal signal at that frequency in order not to waste power and bandwidth; so we have to see if, somehow, we can find such a reference signal hidden in the transmitted data. Consider the received signal after the carrier has been stripped off:

$$\boldsymbol{b}[n] = \sum_i \boldsymbol{a}_i g[n - iK]; \tag{2.37}$$

now, if always the same symbol was transmitted, i.e. $\boldsymbol{a}[i] = \boldsymbol{\alpha} \quad \forall i$, then $\boldsymbol{b}[n]$ would be periodic with period K samples: $\boldsymbol{b}[n + hK] = \boldsymbol{b}[n]$. In this case the Fourier transform of $\boldsymbol{b}[n]$ would contain only spectral lines at multiples of $2\pi/K$; we could then isolate the fundamental at $2\pi/K$ and feed it to the PLL to achieve perfect lock with the baud rate. The transmitted symbols, however, are not constant. We might be tempted to think that, at least "on average", $\boldsymbol{b}[n]$ might look periodic but this is not the case. If we take the expectation of the signal, we have

$$E[\boldsymbol{b}[n]] = E[\boldsymbol{a}_i] \sum_i g[n - iK] = 0 \tag{2.38}$$

since, as explained in the notes about modulation, the set of symbols is designed to have zero mean. No luck here, then.

The way around this impasse is found by using a fantastic "trick which dates back to the old days of analog radio receivers: process the signal through a *nonlinearity.* One of the most common such nonlinearities is the square magnitude, $y(z) = |z|^2$; if we process $\boldsymbol{b}[n]$ with this nonlinearity, on average it will be (remember $g[n]$ is real):

$$E[|\boldsymbol{b}[n]|^2] = \sum_k \sum_i \{E[\boldsymbol{a}_k \boldsymbol{a}_i] g[n - iK] g[n - iK]\}. \tag{2.39}$$

Since we have assumed that $\boldsymbol{a}_i$ is an uncorrelated iid sequence, $E[\boldsymbol{a}_k \boldsymbol{a}_i] = \sigma_A^2 \delta[k - i]$ and therefore

$$E[|\boldsymbol{b}[n]|^2] = \sigma_A^2 \sum_i (g[n - iK])^2. \tag{2.40}$$

The last term in the above equation is periodic with period K and this means that, on average, the squared signal will contain a periodic component at the frequency we need. By filtering the squared singnal through a bandpass filter centered around $2\pi/K$ we obtain a sinusoidal component suitable for tracking by the PLL.

Exercises

(**Note:** For the Matlab execises you will find the necessary M-files and data files following the link in the main web page for the class.)

Exercise 1

Assume the specifications for a given telepone line are $f_{\text{min}} = 300$ Hz, $f_{\text{max}} = 3600$ Hz, and a SNR of at least 28 dB. Design a set of operational parameters for a modem transmitting on this line (baud rate, carrier frequency, constallation size). How many bits per second will you be transmitting?

Exercise 2

In these notes we have only considered square constellations, where the points belong to a regular grid and their number is a perfect even square ($M = 4, 16, 36, 64$, etc.). In fact, many other constellation shapes exist, where points are placed in arbitrary arrangements. One of the reasons for choosing constellations of different shapes is that the energy of the transmitted signal is proportional to the parameter σ_A^2 (see equation (1.19)). By arranging the same number of points in a different manner, we can sometimes reduce σ_A^2, and therefore use a larger amplification gain while keeping the total output power constant, which in turn lowers the probaility of error.

Consider the two 8-point constellation in the following figure. Compute their average energy σ_A^2. What do you notice?

(Matlab) Exercise 3

The M-file `modulator.m` (which is also printed in the last pages of this section) implements a digital modulator. The syntax is

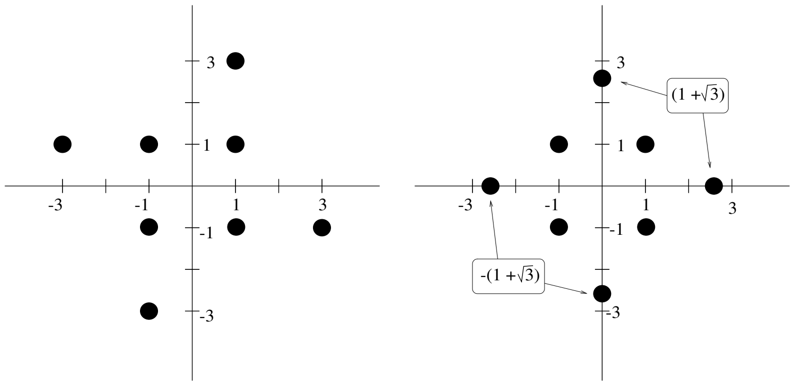

Figure 2.10: Two 8-point constellations.

```
[c b a] = modulator(N, M, fb, fc)
```

where `N` is the total number of symbols to transmit, `M` is the constellation size (which must be a perfect even square), `fb` is the baud rate $1/T$ and `fc` is the carrier frequency. The returned values are `c`, the transmitted signal, `b` the complex baseband signal, and `a`, the complex symbol sequence. Use the program to transmit with the parameters derived in Section 1.3. Plot the various signals (hint: use `plot(a, 'o')` and `axis equal` for the constellation points) and play around with the parameters.

Ok, now that you're familiar with it, write an M-function which demodulates the signal. To help you in the task, the M-file `demodulator.m` sets up the necessary filters for you. Once you have that, you might consider playing around by adding noise to the received signal and see how that affects the decoded symbol. Also, what happens if you modulate with one carrier frequency and try to demodulate with a slightly larger frequency?

modulator.m

```
function [c, b, a] = modulator(N, M, fb, fc)
%
% N    : number of symbols to be transmitted
% M    : size of the constellation (must be a square)
% fb   : baud rate 1/T (Hertz)
% fc   : carrier frequency (Hertz)

%%%%%%%%%%%%%%%%%%%%  Internal parameters: %%%%%%%%%%%%%%%%%%%%%%%

% Upsampling factor
K = 3;

% Internal sampling frequency:
fs = fb * K;

% Raised cosine filter, beta = 0.1, 19 taps
% NOTE: the raised cosine is designed for K=3
g = [3.5801e-17   9.6667e-02   1.1224e-01  -3.7544e-17  -1.6114e-01 ...
    -2.0333e-01   3.8619e-17   4.1178e-01   8.2613e-01   1.0000e+00 ...
     8.2613e-01   4.1178e-01   3.8619e-17  -2.0333e-01  -1.6114e-01 ...
    -3.7544e-17   1.1224e-01   9.6667e-02   3.5801e-17];


%%%%%%%%%%%%%%%%%%%% Parameters check %%%%%%%%%%%%%%%%%%%%%%%%%%%

if (fc < fb/2)
  error('Carrier frequency must be at least twice the baud rate');
end
if (sqrt(M)/2 ~= round(sqrt(M)/2))
  error('M must be a perfect even square');
end

%%%%%%%%%%%%%%%%%%%%  Create random data:   %%%%%%%%%%%%%%%%%%%%%

% Symbol values along the real and complex axis
symbolSet = -sqrt(M)+1:2:sqrt(M)-1;
```

```
% Random values between 1 and sqrt(M)
i = floor(sqrt(M) * rand(N,1)) + 1;
q = floor(sqrt(M) * rand(N,1)) + 1;

% Sequence of complex symbols
a = symbolSet(i) + j * symbolSet(q);


%%%%%%%%%%%%%%%%%%%%%%%%  Modulation:   %%%%%%%%%%%%%%%%%%%%%%%%%%


% Upsample symbol sequence
a = kron(a, [1 zeros(1, K-1)]);

% Zero pad for filtering transients
a = [ zeros(1, 20) a zeros(1, 20)];

% Pulse shaping
b = filter(g, 1, a);

% Modulation
bMod = b .* exp(j * 2 * pi * (fc/fs) * (0:length(b)-1));

% Transmission
c = real(bMod);
```

demodulator.m

```
function a = demodulator(c, fb, fc, fs);
%
% c    : transmitted signal
% fb   : baud rate 1/T (Hertz)
% fc   : carrier frequency (Hertz)
% fs   : sampling frequency

%%%%%%%%%%%%%%%%%%  Internal parameters: %%%%%%%%%%%%%%%%%%%%%%%

% Downsampling factor
K = fs/fb;

% Carrier digital frequency
omegac = 2*pi * (fc/fs);

% Hilbert filter, 2L+1 taps
L =  19;
% the Hilbert filter is antisymmetric, so we need only first half
h = [-1.2787e-03   3.1722e-07  -3.2636e-03   9.4594e-07  -7.1031e-03 ...
      2.3216e-06  -1.3551e-02   2.7060e-06  -2.3770e-02   3.4028e-06 ...
     -3.9568e-02   3.6339e-06  -6.4415e-02   3.2654e-06  -1.0703e-01 ...
      2.6130e-06  -1.9945e-01   1.5714e-06  -6.3227e-01];
% Complete impulse response
h = [h 0 -fliplr(h)];



%%%%%%%%%%%%%%%%%%%%%%%%  Demodulation:   %%%%%%%%%%%%%%%%%%%%%%%%%%
```

. . . to be completed . . .