Contents

List of Figures

Chapter 1

Introduction

The sea is really a particular environment. Its properties are very typical and it's why a lot of scientists work on it in order to determine its characteristics. The underwater acoustics is one of the domain which are particularly studied, because of its importance in a lot of applications, civilians and militaries. This is very useful also because the ocean is totally opaque to the electro-magnetic waves. So, the sound results to be the better mean to communicate under the surface of the sea. But, even if it is at the moment the better way for underwater communications, a lot of problems have been encountered as soon as we tried to transmit data. One of the problems encountered is about the distorsion of our data. Indeed, when you send a data under the sea surface, a lot of parameters make that the data received is not the same as the data sent. It is a big problem and we can't do differently. So, the only way is to process this data after receiving it, trying to do the inverse of what has been done during the communication, and so getting exactly what was sent. When we speak about this, it seems very simple. But the knowledge it requires, about the environment, the communications, for example, is so huge that it rapidly becomes really difficult.

Our project was to study the equalizers for the underwater acoustics. But, in order to study this, we needed a certain background in digital communications. This background could only be obtained by studying digital communications but not applied to underwater acoustics. The equalizers are not typical from underwater acoustics, they are also used for all the digital communications. So, we also had to study them in a general case not applied to the sea. All these constraints, added to the duration of our project which is very short, has done that in the most part of our report we don't speak about the underwater acoustics.

So, in a first part, we dealt with theoretical problems. This theory we have exposed is the necessary background we need if we want to work and understand well the equalizers. Then, we started speaking about the equalizers. It is obvious that we couldn't get directly a complicated equalizer we can use everywhere. It's why we have chosen to study first two simple ones. We have studied their characteristics to make them functioning well and to see really how an equalizer runs. Finally, we studied a more complicated equalizer, the adaptive equalizer, which is more adapted to the reality. For this equalizer, we finally have taken an environment for a communication which is more realistic, closer to what happens in the sea. This is the real application we did about equalizers for underwater acoustics.

Chapter 2

Theory on communication systems

In this chapter we are going to study the bases of digital communication systems. We present this theory which is very important in order to understand the equalizers. All this chapter is based on various references that we will name progressively. This in fact is a summary of what we have to know about communication systems to treat the subject. First, we will study how we construct a signal to transmit, so the characteristics of an input signal in a channel. Then, we will see some ways to transport the information to the channel after having created it, talking about carrier transmission. Finally, we will apply this to the transmission in a channel, but a band limited channel, which is closer to the reality.

2.1 Baseband Digital Transmission

Here, we are going to develop the mean to create signal,beginning with the simplest which is the binary signal, and complicating this more and more in order to get more and more information transmitted. But it is easy to imagine that having more information results in having more difficulties for the treatment of the signal.

2.1.1 Binary Signal Transmission (Reference [1], [2, [3] and [4])

Binary data consists of a sequence of 0's and 1's which are transmitted using two signal waveforms $s_0(t)$ and $s_1(t)$(Figure 2.1) with $0 < t < Tb$.

We consider that 0's and 1's are equally probable and that the signal is transmitted through an Additive White Gaussian Noise(AWGN) channel. Then the noise will be n(t) with power spectrum of $N_0/2$ watts/hertz.The received signal is:

$$r(t) = s_i(t) + n(t), \tag{2.1}$$

with i=0 or 1.

As an example, we consider that a 0 is transmitted.The receiver is going to try to minimize the transmission error. For an AWGN channel, the receiver consists of two buildings blocks: a signal correlator or a matched filter

Figure 2.1: Signal waveforms $s_0(t)$ and $s_1(t)$ for a binary communication system

and a detector. The signal correlator cross-correlates r(t) with the two possible transmitted signals which gives:

$$r_i(t) = \int_0^t r(\tau)s_i(\tau)d\tau, \tag{2.2}$$

Here, with 0 transmitted, we get:

$$\begin{aligned} r_0 &= \int_0^{T_b} r(t)s_0(t)dt \\ &= \int_0^{T_b} s_0^2(t)dt + \int_0^{T_b} n(t)s_0(t)dt \\ &= E + n_0, \end{aligned} \tag{2.3}$$

and

$$r_1 = n_1 \tag{2.4}$$

with

$$n_i = \int_0^{T_b} n(t)s_i(t)dt. \tag{2.5}$$

As n(t) is a sample function of a white Gaussian process with power spectrum of $N_0/2$, we have:

$$E(n_i) = \int_0^{T_b} s_0(t)E[n(t)]dt = 0 \tag{2.6}$$

and variances

$$\begin{aligned} \sigma_i^2 &= E(n_i^2) \\ &= \frac{EN_0}{2}. \end{aligned} \tag{2.7}$$

Then, the probability density functions of r_0 and r_1 are:

$$p(r_0|s_0(t)transmitted) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(r_0 - E)^2}{2\sigma^2} \tag{2.8}$$

$$p(r_1|s_0(t)transmitted) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-r_1^2}{2\sigma^2} \tag{2.9}$$

The two probability density functions are illustrated on Figure 2.2.

Using a matched filter is an other way to do but the result is exactly the same than with the signal correlator.

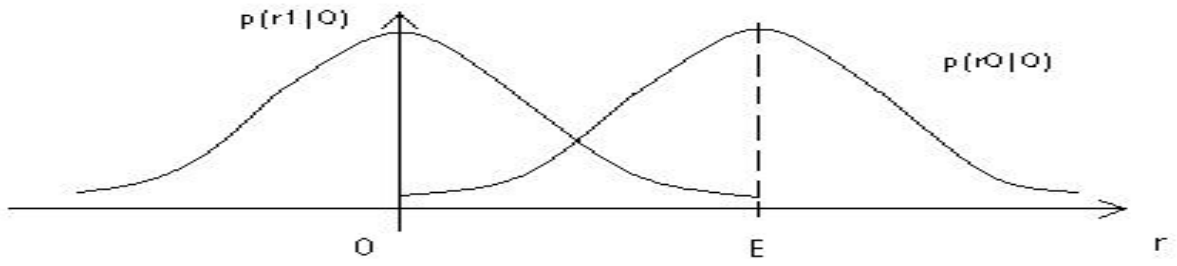


Figure 2.2: Probability density functions $p(r_0|0)$ and $p(r_1|0)$ when $s_0(t)$ is transmitted

The second block after one the two precedents is the detector. This one observes the outputs r_0 and r_1 and decides whether the signal transmitted was s_0 or s_1. The objective is to minimize the probability of error which is:

$$P_e = P(r_1 > r_0) = P(n_1 > E + n_0) = P(n_1 - n_0 > E) \tag{2.10}$$

If we note $x=n_1-n_0$, we know that x is zero-mean gaussian because n_1 and n_0 are too. So,

$$E(x^2) = 2(\frac{EN_0}{2}) = EN_0 = \sigma_x^2 \tag{2.11}$$

Finally, the probability of error is:

$$\begin{aligned} P_e &= \frac{1}{\sqrt{2\pi}\sigma_x}\int_E^\infty \exp\frac{-x^2}{2\sigma_x^2}dx \\ &= \frac{1}{\sqrt{2\pi}}\int_{\sqrt{E/N_0}}^\infty \exp\frac{-x^2}{2}dx \\ &= Q(\sqrt{\frac{E}{N_0}}), \end{aligned} \tag{2.12}$$

where,

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty \exp\frac{-t^2}{2}dt \tag{2.13}$$

We call the ratio E/N_0 the Signal to Noise Ratio(SNR) and SNRindB the signal to noise ratio in dB which is $10\log_{10}(E/N_0)$.Figure 2.3 shows an example obtained with Matlab of a graphic of a probability of error for orthogonal signals.

There are also other ways for transmitting binary information. The use of signal correlator, matched filter and detector is quite the same. So we just give the characteristics of other signal waveforms.

There are the on-off signals whose received signal waveforms are n(t)if a 0 is transmitted and n(t) + s(t) if a 1 is transmitted. Then there are also the antipodal signals whose received signal waveform is:

$$r(t) = \pm s(t) + n(t) \tag{2.14}$$

It is important also to deal with the signal constellation diagrams because they are very useful to obtain a concrete representation of the binary signals. Such a diagram is called signal constellation and uses a characterization based on energy. On Figure 2.4, you can see the signal constellations for orthogonal binary signals, for different values of

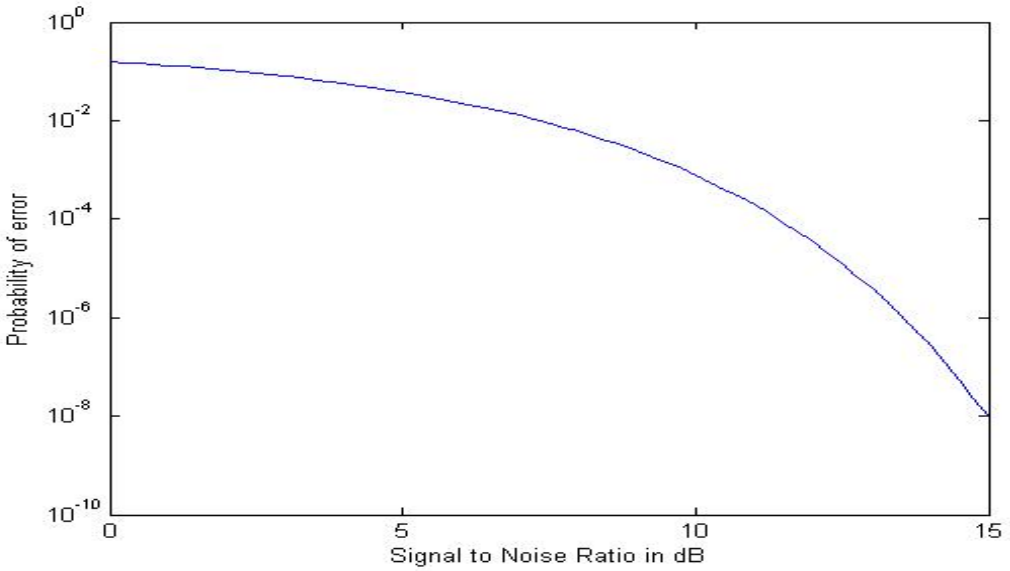
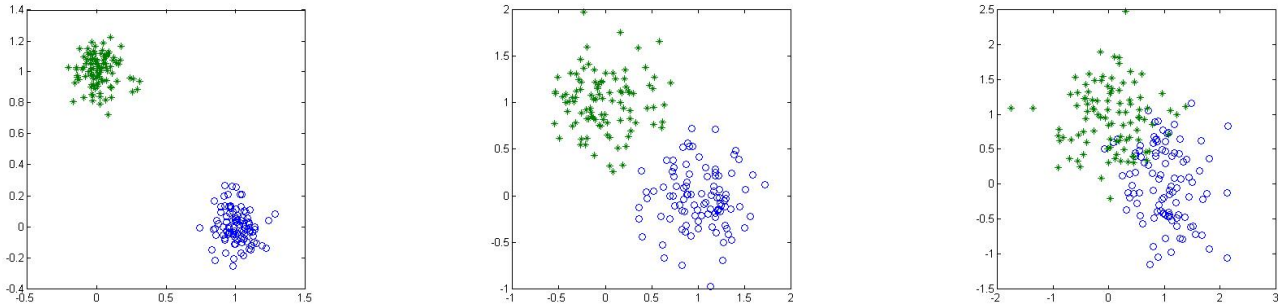


Figure 2.3: Probability of error for orthogonal signals

σ and using a Monte Carlo simulation. The principle of the Monte-Carlo simulation is to compare the emission of 10000 bits for example with the choice of the detector.



σ=0.1 σ=0.3 σ=0.5

Figure 2.4: Received signals points to the selector for orthogonal signals for three values of σ (Monte-Carlo simulation)

2.1.2 Multiamplitude Signal Transmission

In the precedent subsection, we have detailed the processus used for the transmission of digital information by use of binary signal waveforms. We are now going to see that this process can be extended to transmit multiple bits per signal waveform, using signal waveforms that take multiple amplitude levels. Then we can construct M=2^k multiamplitude signal waveforms represented as:

$$s_m(t) = A_m g(t), \tag{2.15}$$

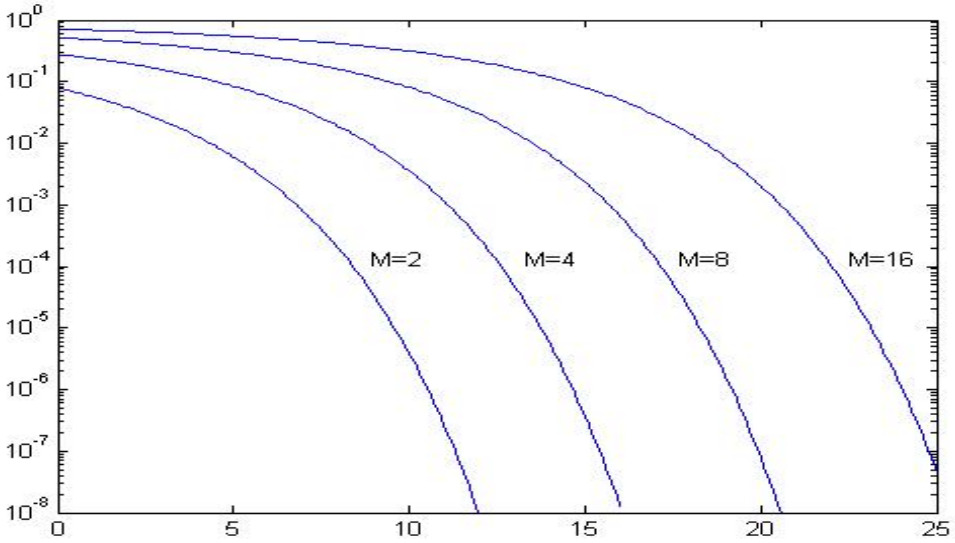


Figure 2.5: Symbol-error probability for M-level PAM for M=2,4,8,16

with $0 \leq t \leq T$, m=0,1,...,M-1, g(t) a rectangular pulse and where the amplitude values A_m are:

$$A_m = (2m - M + 1)d, \tag{2.16}$$

where 2d is the Euclidean distance between two adjacent points.

These signals are called Pulse Amplitude Modulated (PAM) signals. Then each signal waveform conveys $k=\log_2 M$ bits of information. The bit rate here is $R=1/T=1/kT_b$. Here the optimum receiver consists again of a signal correlator (or matched filter) followed by a detector but a particular one which is an amplitude detector that computes the Euclidean distances $D_i=|r-A_i|$ where r is the correlator output. So, the decision is made in favor of the amplitude level that corresponds to the smallest distance.The probability of error for the optimum detector is:

$$P_M = \frac{2(M-1)}{M} Q\left(\sqrt{\frac{6(log_2 M)E_{avb}}{(M^2-1)N_0}}\right), \tag{2.17}$$

where E_{avb} is the average energy for an information bit.We can see on Figure 2.5 a Matlab graphic which illustrates the probability of a symbol error for M=2,4,8,16.

2.1.3 Multidimensional Signals

In the preceding subsection, the multiamplitude signal waveforms were one-dimensional signals.We now consider the construction of a class of $M=2^k$ signal waveforms that have a multidimensional representation. Then we can represent the set of signal waveforms in N-dimensional space. We consider the set of $M=2^k$ waveforms $s_i(t)$ for i=0,1,...,M-1 which have the properties of mutual orthogonality and equal energy, which is traduced by:

$$\int_0^T s_i(t)s_k(t)dt = E\delta_{ik}, \tag{2.18}$$

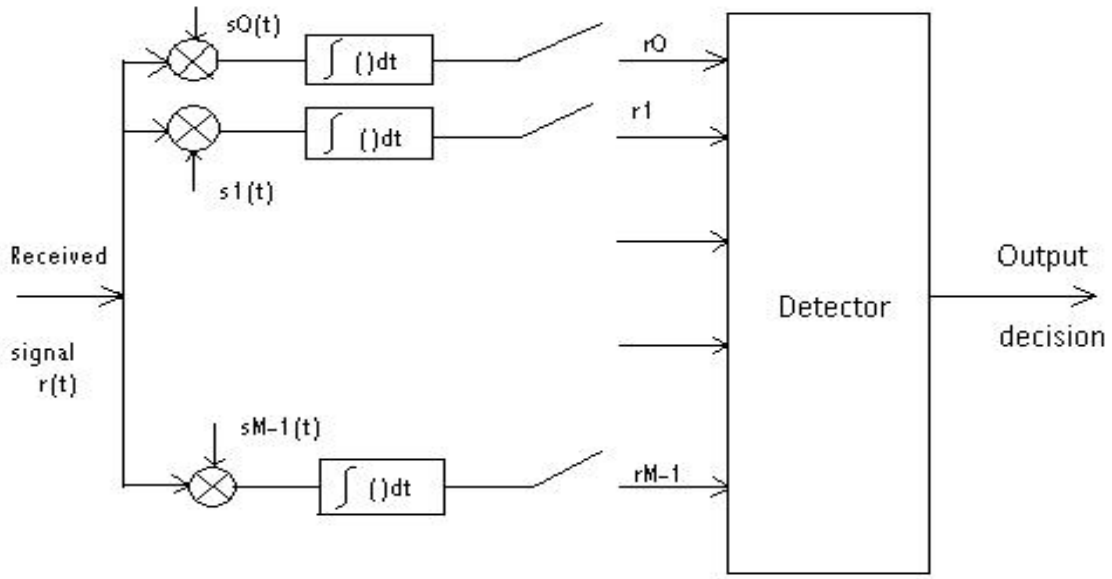


Figure 2.6: Optimum receiver for multidimensional signals

where i,k=0,1,...,M-1 and $\delta_{ik}$is the kronecker symbol. We consider that the channel used is the same and that the definitions for R, T_b, M are the same that the precedent subsections.The simplest way to construct a set of M=2^k equal-energy orthogonal waveforms in the interval (0,T) is to subdivide the interval into M equal subintervals of T/M duration. Then the signal waveforms have an energy E=$\frac{A^2T}{M}$. We can represent these orthogonal waveforms by orthogonal vectors:

$$s_0 = (\sqrt{E}, 0, 0, ..., 0)$$

$$s_1 = (0, \sqrt{E}, 0, ..., 0)$$

$$.$$

$$.$$

$$s_{M-1} = (0, 0, ..., 0, \sqrt{E})$$

If the transmitted waveform is s_0(t), the received waveform is:

$$r(t) = s_0(t) + n(t) \tag{2.19}$$

In order to minimize the error, we pass the signal r(t) through a parallel bank of M signal correlators (or matched filters) as shown on Figure 2.6

All the r_i can be represented with the vector r=[r_0,r_1,...,$r_M - 1$]^t where r_0=E+n_0 and r_i=n_i for i$\neq$0.Using the same approach than in the first subsection and because the noise is zero mean gaussian, we find that:

$$\sigma^2 = \frac{N_0 E}{2}, \tag{2.20}$$

and then, considering that $s_0(t)$ has been transmitted, the probability density functions for the correlator outputs are:

$$p(r_0|s_0(t) was transmitted) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(r_0 - E)^2}{2\sigma^2} \tag{2.21}$$

$$p(r_i|s_0(t) was transmitted) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-r_i^2}{2\sigma^2} \tag{2.22}$$

Then the optimum detector observes the M correlator outputs r_i and selects the signal corresponding to the largest correlator output.The probability of a correct decision is simply:

$$P_c = P(r_0 > r_1, r_0 > r_2, ..., r_0 > r_{M-1}), \tag{2.23}$$

and the probability of a symbol error is P_M=1-P_c. It is possible to show that P_M can be expressed as:

$$P_M = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (1 - [1 - Q(y)]^{M-1}) \exp \frac{-(y - \sqrt{\frac{2E}{N_0}})^2}{2} dy \tag{2.24}$$

This probability is the same for any $s_i(t)$ transmitted. It can also be interesting to convert the probability of a symbol error into an equivalent probability of a binary digit error. So, for equiprobable orthogonal signals, all symbol errors are the same and occur with probability:

$$\frac{P_M}{M-1} = P_M 2^k - 1 \tag{2.25}$$

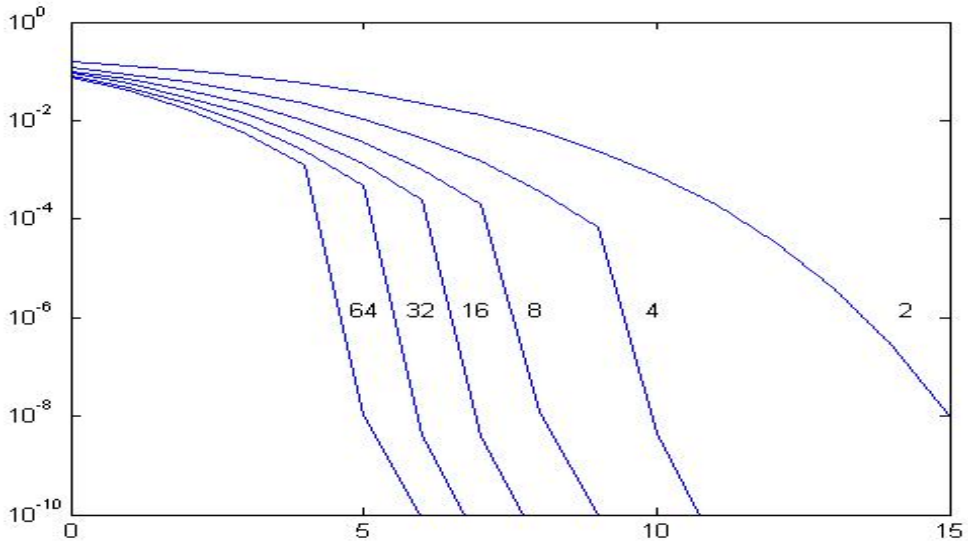


Figure 2.7: Bit-error probability for orthogonal signals for M=2,4,8,16,32,64.

But this error is for one symbol and if you want to obtain the average number of bit errors per k-bit symbol, you take the sum and you get $k\frac{2^{k-1}}{2^k-1}P_M$. Finally the average bit-error probability is the 1.25 result, divided by k, the number of bits per symbol:

$$P_b = \frac{2^{k-1}}{2^k - 1} P_M \tag{2.26}$$

Figure 2.7 shows a Matlab graphic of the probability of a binary digit as a function of the SNR per bit E_B/N_0 for M=2,4,8,16,32,64 where E_b=E/k is the energy per bit.

An other type of set can be done. One-half the waveforms are orthogonal and the other half are the opposite of these orthogonal waveforms. Such a set is called a biorthogonal set of signals. The way to obtain the probability of error is the same that with the orthogonal signals.

2.2 Digital Transmission via Carrier Modulation (Reference [5]and [6])

In the preceding section, we have considered the transmission of digital information through baseband channels. In such a case, the information-bearing signal is transmitted directly through the channel without the use of a sinusoidal carrier. But most communication channels are bandpass channels, so we have to shift the frequency of the information-bearing to the frequency band of the channel. In this section we study two types of carrier-modulated signals that are suitable for bandpass channels: amplitude-modulated signals and phase-shift keying. In this section we consider that g_T(t) is rectangular that is $g_T(t) = \sqrt{\frac{2}{T}}$, for 0$\leq t \leq$T.

2.2.1 Carrier-Amplitude Modulation

This is the first well known way to modulate a signal: using a carrier-amplitude modulation.

Modulation step

We use the same signal waveform as in the 1.1.2, so:

$$s_m(t) = A_m g_T(t) \tag{2.27}$$

with the same characteristics for A_m and g_T(t). The spectrum of the baseband signals is assumed to be contained in the frequency band $|f| \leq W$, where W is the bandwidth of $|G_T(f)|^2$. To transmit the digital signal waveform through a bandpass channel, the baseband signal waveforms s_m(t) are multiplied by a sinusoidal carrier of the form $cos2\pi f_c t$, where f_c is the carrier frequency and corresponds to the center frequency in the passband of the channel. So, we obtain a transmitted signal waveform as:

$$u_m(t) = A_m g_T(t) cos2\pi f_c t \tag{2.28}$$

When g_T(t) is rectangular, we call this Amplitude-Shift Keying (ASK). We know that the fourier transform of the carrier is $[\delta(f - f_c) + \delta(f + f_c)]/2$ so the multiplication of the two signals in the time domain corresponds to the convolution of their spectra in the frequency domain. The spectrum of the amplitude-modulated signal is:

$$U_m(f) = \frac{A_m}{2}[G_T(f - f_c) + G_T(f + f_c)] \tag{2.29}$$

The spectrum of the baseband signal s_m(t) is so shifted in frequency by the carrier frequency f_c. The bandpass signal is a Double-Sideband Suppressed-Carrier (DSBSC). Figure 2.8 shows the effect of this modulator.

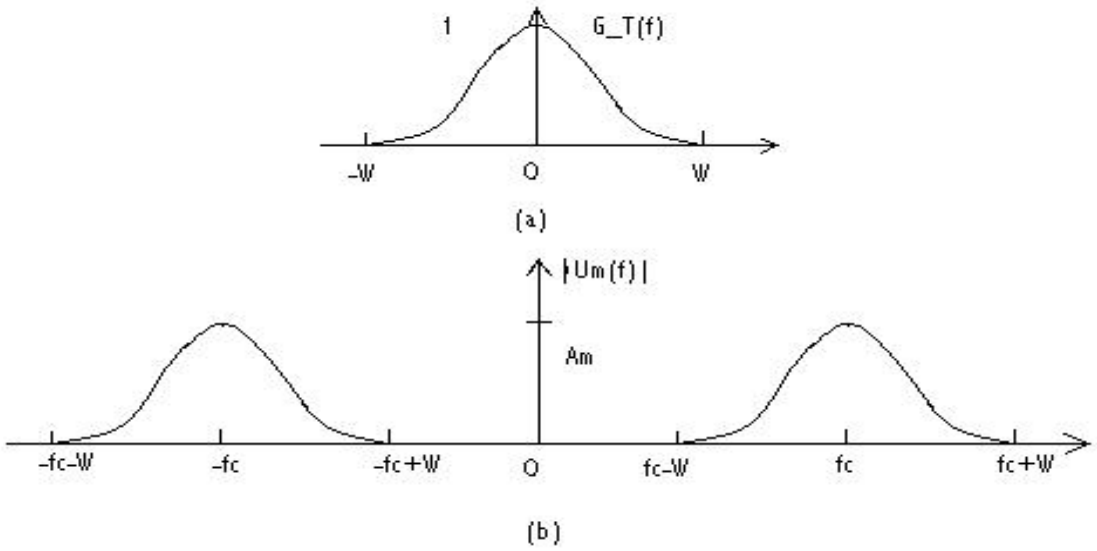


Figure 2.8: Spectra of (a) baseband and (b) amplitude-modulated signals

Demodulation step

For this part, we consider that:

$$u_m(t) = s_m\psi(t), \tag{2.30}$$

with

$$\psi(t) = g_T(t)cos2\pi f_c t \tag{2.31}$$

and s_m=A_m.

The demodulation can be done using a way as correlation or matched filtering. As an example, we consider a correlation-type demodulator.The received signal can be expressed as:

$$r(t) = u_m(t) + n(t) = s_m\psi(t) + n(t) = A_m g_T(t)cos2\pi f_c t + n(t) \tag{2.32}$$

where n(t) is a bandpass noise process:

$$n(t) = n_c(t)cos2\pi f_c t - n_s(t)sin2\pi f_c t, \tag{2.33}$$

where n_c and n_s are the quadrature components of the noise. So ,we cross-correlate the received signal r(t) with $\psi(t)$ and we get:

$$\int_{-infty}^{\infty} r(t)\psi(t)dt = A_m + n = s_m + n \tag{2.34}$$

where n is the noise component at the output of the correlator whose variance, thanks to its zero mean property, can be expressed as:

$$\sigma_n^2 = \int_{-\infty}^{\infty} |\Psi(f)|^2 S_n(f)df, \tag{2.35}$$

where $\Psi(f)$ is the fourier transform of $\psi(t)$ and is:

$$\Psi(f) = \frac{1}{2}[G_T(f - f_c) + G_T(f + f_c)], \tag{2.36}$$

and $S_n(f)$ is the power-spectral density of the additive noise whose process is: $S_n(f) = \frac{N_0}{2}$ for $|f - f_c| \leq W$ and 0 otherwise. Using 2.35 and 2.36, we obtain that $\sigma_n^2 = N_0/2$. Then the probability of error is the same as the one of a baseband PAM, given by 2.17. Figure 2.9 gives the spectra of base-band signal and amplitude-modulated signal, obtained with Matlab simulation.

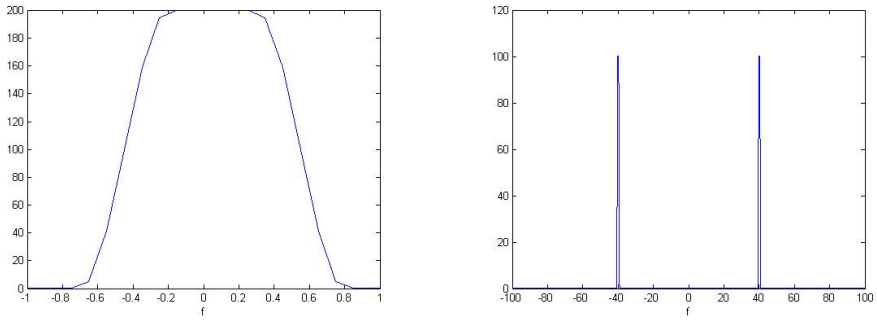


Figure 2.9: Spectra of baseband signal and amplitude-modulated (bandpass) signal

2.2.2 Carrier-Phase Modulation

This is the second well known way for the modulation of a signal in order to transport the information through a form which suits the best to the channel.

Modulation step

In carrier-phase modulation, the information that is transmitted over a communication channel is impressed on the phase of the carrier. Since the range of the carrier phase is $0 \leq \theta \leq 2\pi$, the carrier phases used to transmit digital information via digital-phase modulation are $\theta_m = 2\pi m/M$, for m=0,1,...,M-1. The general representation of a set of M carrier-phase-modulated signal waveforms is:

$$u_m(t) = Ag_T(t)cos(2\pi f_c t + \frac{2\pi m}{M}), \tag{2.37}$$

where $g_T(t)$ is the transmitting filter pulse shape and A the signal amplitude. This type of modulation is called Phase-Shift Keying (PSK). PSK signals have equal energy that is:

$$\begin{aligned} E_m &= \int_{-\infty}^{\infty} u_m^2(t)dt \\ &= \int_{-\infty}^{\infty} A^2 g_T^2(t)cos^2(2\pi f_c t + \frac{2\pi m}{M})dt \\ &= \frac{1}{2}\int_{-\infty}^{\infty} A^2 g_T^2(t)dt + \frac{1}{2}\int_{-\infty}^{\infty} A^2 g_T^2(t)\cos(4\pi f_c t + \frac{4\pi m}{M})dt \end{aligned}$$

$$
\begin{aligned}
&= \frac{A^2}{2}\int_{-\infty}^{\infty} g_T^2(t)dt \\
&= E_S, \qquad (2.38)
\end{aligned}
$$

where E_s is the energy transmitted per symbol. Then the transmitted signal waveforms in the symbol interval $0 \leq t \leq T$ may be expressed as, considering that $A=\sqrt{E_S}$:

$$u_m(t) = \sqrt{\frac{2E_S}{T}} cos(2\pi f_c t + \frac{2\pi m}{M}) \qquad (2.39)$$

The result is that the transmitted signals have a constant envelope and the carrier phase changes abruptly at the beginning of each signal interval.Using a trigonometric function, we can split 2.39 in:

$$
\begin{aligned}
u_m(t) &= \sqrt{E_S} g_T(t) \cos(\frac{2\pi m}{M}) cos 2\pi f_c t - \sqrt{E_S} g_T(t) sin(\frac{2\pi m}{M}) sin 2\pi f_c t \\
&= s_{mc}\Psi_1(t) + s_{ms}\Psi_2(t), \qquad (2.40)
\end{aligned}
$$

where:

$$s_{mc} = \sqrt{E_S}\cos(\frac{2\pi m}{M})$$

$$s_{ms} = \sqrt{E_S}\sin(\frac{2\pi m}{M})$$

$$\Psi_1(t) = g_T(t) cos 2\pi f_c t$$

$$\Psi_2(t) = -g_T(t) sin 2\pi f_c t$$

Figure 2.10 illustrates the eight waveforms for the case in which $f_c=6/T$ by a Matlab simulation.

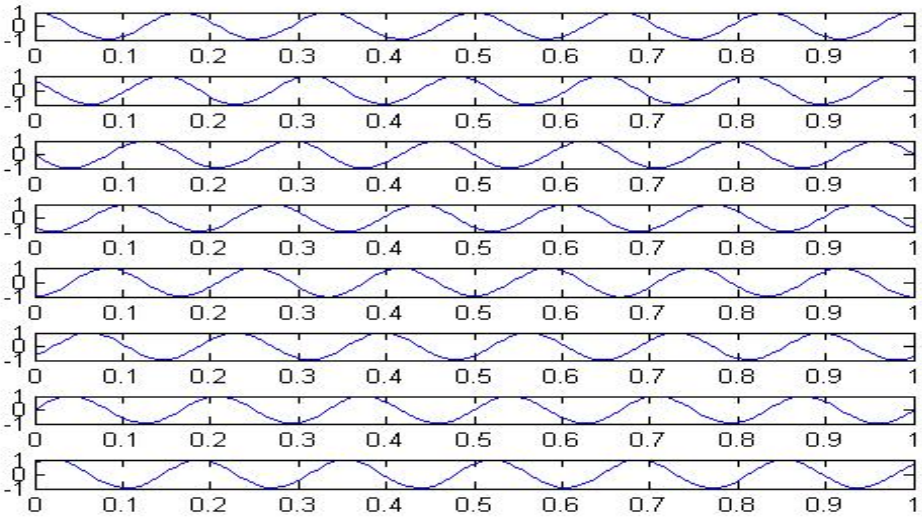


Figure 2.10: M=8 constant amplitude PSK waveforms

Demodulation and detection step

We use the same noise as in the precedent subsubsection. Then, the received bandpass signal is:

$$\begin{aligned} r(t) &= u_m(t) + n(t) \\ &= u_m(t) + n_c(t)cos2\pi f_c t - n_s(t)sin2\pi f_c t \end{aligned} \tag{2.41}$$

The received signal may be correlated with Ψ_1(t) and Ψ_2(t) which is expressed as:

$$\begin{aligned} r &= s_m + n \\ &= (\sqrt{E_S}cos_f rac2\pi mM + n_c; \sqrt{E_S}sin_f rac2\pi mM + n_s, \end{aligned} \tag{2.42}$$

with:

$$n_c = \frac{1}{2}\int_{-\infty}^{\infty} g_T(t)n_c(t)dt$$

$$n_s = \frac{1}{2}\int_{-\infty}^{\infty} g_T(t)n_s(t)dt$$

The optimum detector projects the received signal r onto each of the M possible transmitted signal vectorss$_m$ and selects the vector corresponding to the largest projection.

2.3 Digital Transmission through Bandlimited Channels

In this chapter, we treat several aspects of digital transmission trough band-limited channels. We begin by describing the spectral characteristics of PAM signals. Then, we consider the characterization of band-limited channels and the problem of designing signal waveforms for such channels. Finally, we treat the problem of distortion caused by band-limited channels, showing that channel distortion results in intersymbol interference, which causes errors in signal demodulation.

2.3.1 The Power Spectrum of a Digital PAM Signal

A digital PAM signal at the input of a communication channel can be represented as

$$v(t) = \sum_{n=-\infty}^{\infty} a_n * g(t - nt) \tag{2.43}$$

where a_n is a sequence of amplitudes, g(t) is a pulse waveform, T is the symbol interval. Since the information sequence is a random sequence the sequence a_n is also random.Consequently the PAM signal v(t) is a sample function of a random process V(t), and to determine its special characteristics, we must evaluate its power spectrum. The power spectrum is the Fourier transform of the average autocorrelation function for the PAM signal. Thus:

$$R_v(t + \tau; t) = E[V(t)V(t + \tau)] \tag{2.44}$$

Random processes that have periodic mean value and a periodic autocorrelation function are called periodically stationary or cyclostationary, and variable t can be eliminated by averaging R_v over a single period. Hence,$\bar{R_v}(\tau)$ can be expressed as

$$\bar{R_v}(\tau) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} R_v(t+\tau;t)dt \tag{2.45}$$

$$\bar{R_v}(\tau) = \frac{1}{T}\sum_{m=-\infty}^{\infty} R_a(m)R_g(\tau - mT) \tag{2.46}$$

where $R_a(m) = E[a_n a_{n+m}]$ is the autocorrelation of the sequence a_n and $R_g(\tau) = \int_{-\infty}^{\infty} g(t)g(t+\tau)dt$ Hence,the power spectrum of V(t) is

$$\gamma_v(f) = \int_{-\infty}^{\infty} \bar{R_v}(\tau)e^{-j2\pi f\tau}dt = \frac{1}{T}\gamma_a(f)|G(f)|^2 \tag{2.47}$$

where γ_a is the power spectrum of a_n.Thus γ_a is defined as:

$$\gamma_a(f) = \sum_{m=-\infty}^{\infty} R_a(m)e^{-j2\pi fmT} \tag{2.48}$$

Notice that when the sequence $a_n$is uncorrelated, the power spectrum of V(t) is dependant only on the spectral characteristics of the pulse g(t).

2.3.2 Characterization of the Band Limited Channels and Channels Distortion

Most of communication systems may be characterized as bandlimited linear filters.That is why we describe such channels by their frequency response:

$$C(f) = A(f)e^{j\theta(f)} \tag{2.49}$$

where A(f) is called the amplitude response and $\theta(f)$ is called the phase response.Instead of phase response it can be used the envelope delay or group delay defined as:

$$\tau(f) = -\frac{1}{2\pi}\frac{d\theta(f)}{df} \tag{2.50}$$

A channel is said nondistorting or ideal if, within the bandwith W occupied by the transmitted signal, A(f) is constant and θ is a linear function. If A(f) is not constant, the distortion is called amplitude distortion. If $\theta(f)$ is not constant, it is called delay distortion.As a result, a succession of pulses are smeared to the point that we cannot distinguish them at the receiving terminal. This overlapping is called intersymbol interference (ISI) . Figure 2.11 illustrates a bandlimited pulse having zeros periodically spaced in time at points ± T ,± 2T,... and the received pulse after a transmission through a channel modeled as having a linear envelope delay $\tau(f)$.

We notice the zero crossings are no longer periodically spaced. As we will see in the following chapter it is possible to compensate for the nonideal frequency response characteristic of the channel by using a filter or an equalizer at the demodulator.

Radio channels, such as short-wave ionospheric propagation(HF),mobile cellular radio are examples of time-dispersive wireless channels. In these channels, the number of paths and the relative time delays among the paths

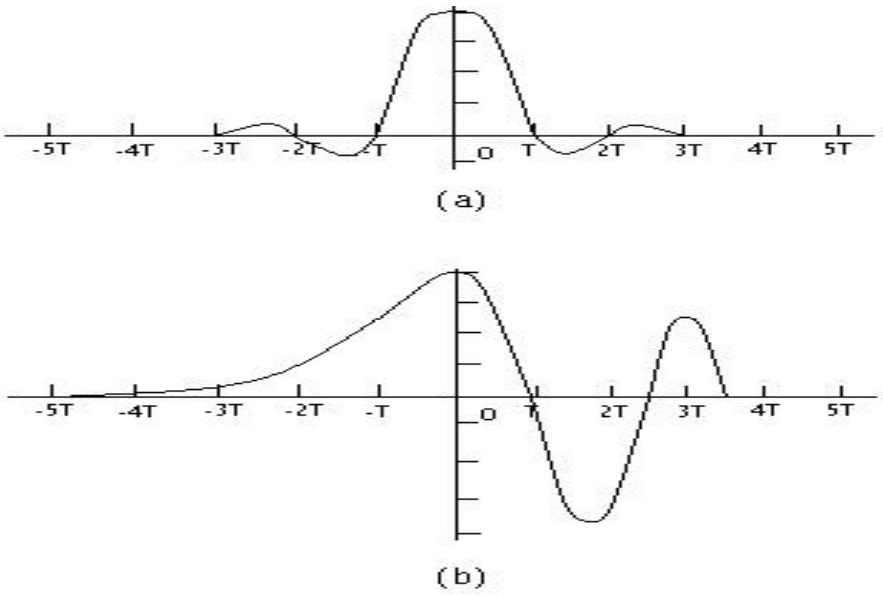


Figure 2.11: Effect of channel distorsion: (a) channel input; (b) channel output.

vary with time.We call these radio channels time-variant multipath channels. These time-variant conditions rise many variety of frequency response characteristics.As a result, these radio channels are characterized statistically in terms of the scattering function, which is a bidimensional representation of the average received signal power as a function of relative time delay and doppler frequency spread.

Notice that a transmission rate of 10^7 sympbols/second it will result a interference that smears about 7 symbols.

A bandlimited channel can be modelled as a linear filter whose frequency response characteristics match the frequency response characteristics of the channel.

Chapter 3

Simulation of Linear Equalizers on Matlab

As we know the theory about communication systems and signals, we know the problems (ISI, channel distortion, etc...) (Reference [7] and [8]) which can be encountered in the transmission of a signal. The aim now is to solve these problems by using equalizers which purpose vulgarly is to get the inverse of the channel in order to obtain the input signal. But it's easy to imagine that it's not so simple. A transmission channel can be very complicated and it can be very difficult to build a good equalizers. So we are first going to study the theory of building an equalizer, but just for linear equalizers in this chapter. Then we will use again the Matlab software to simulate signals and channels in order to build the good equalizers. We will detail all the parameters of an equalizer which can have an importance in the treatment of a signal. The signals and the channels we are going to use will be more and more complicated but will stay simple because of the complexity of the phenomenon.

3.1 Theory About Linear Equalizers (Reference [9])

The most common type of channel equalizer used in practice to reduce ISI is a linear FIR filter with adjustable coefficients c_i as shown in Figure 3.1.

On channels whose frequency response characteristics are time-invariant, the parameters of the equalizer are fixed during the data transmission.it is called preset equalizers.When parameters can be changed, it is called adaptive equalizer.

First, let us consider the design characteristic for a linear equalizer from a frequency domain viewpoint. Figure 3.2 shows a block diagram of a system that employs a channel equalizer.

The demodulator consists of a receiver filter with frequency response $G_R(f)$ in cascade with a channel equalizing filter that has a frequency response $G_E(f)$. The receiver filter that has a frequency response $G_R(f)$ is matched to the transmitter response,and the product $G_R(f)G_T(f)$ is designed so that there is either zero ISI at the sampling

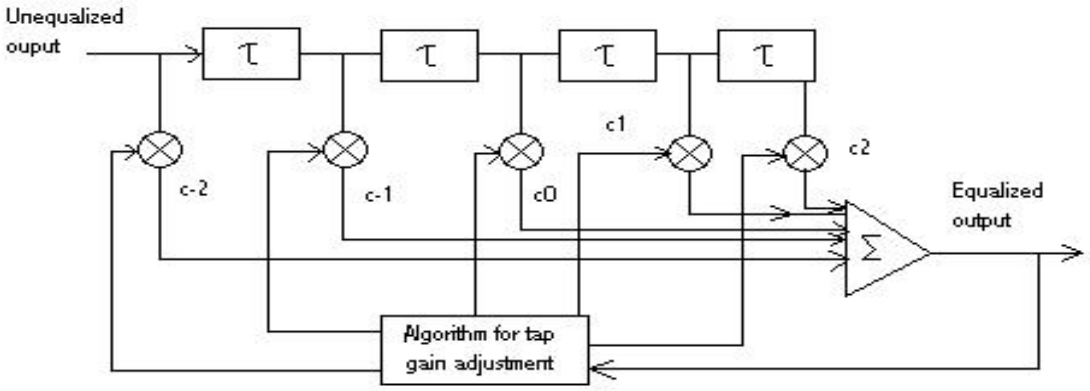


Figure 3.1: Linear transversal filter

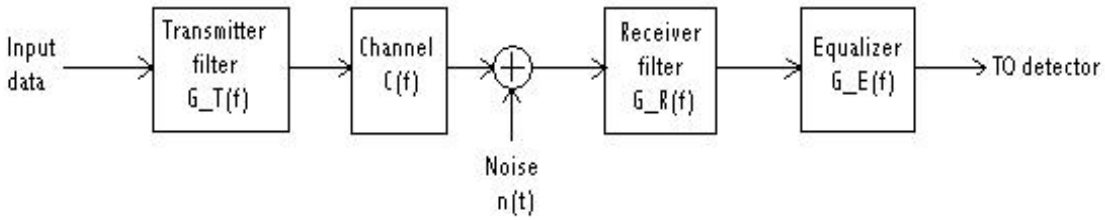


Figure 3.2: Block diagram of a system with an equalizer

instants or controlled ISI for partial response signal. For the system described in the figure 2.14 the desire condition for zero ISI is

$$G_T(f)C(f)G_R(f)G_E(f) = X_{RC}(f) \tag{3.1}$$

where $X_{RC}(f)$ is the desired raised-cosine spectral characteristic.

We design $G_R(t)$ as $G_R(f)G_T(f) = X_{RC}(f)$. As a result, the frequency response of the equalizer that compensates the channel distortion is

$$G_E(f) = \frac{1}{C(f)} = \frac{1}{|C(f)|}e^{-j\theta_c(f)} \tag{3.2}$$

In this case the amplitude equalizer is said to be the inverse channel filter to the channel response. This filter completely eliminates the ISI by forcing them to be zero at the sampling instant t=kT for k=0,1,... . Thus it is called zero-forcing equalizer.Thanks to it, the input to the detector is

$$z_k = a_k + \eta_k, k = 0, 1, ... \tag{3.3}$$

In practice, the ISI caused by channel distortion is usually limited to a finite number of symbols on each side of the desired symbol.That is why we can implement the channel equalizer as a finite -duration impulse response

(FIR) filter, with adjustable coefficients c_n as illustrated before in Figure 3.1. The time delay τ between adjacent coefficients may be selected as large as T, and in this case the filter is called symbol-spaced equalizer. The input to the equalizer is

$$y_k = a_k + \sum_{n=0, n\neq k}^{\infty} a_n x_{k-n} + v_k \tag{3.4}$$

where x_k is the signal pulse response of the receiving filter x(t) sampled at times $t = kT, k = 0, 1, ...$

where a_k represents the desired information symbol at the kth sampling instant

where v_k is the additive noise at the kth sampling instant.

However, when the symbol rate 1/T ¡ 2W frequencies above 1/T cannot be distinguished from frequencies below 1/T. In this case the equalizer compensates for the aliased channel-distorted signal.

On the other hand, when the time delay τ is selected such that $1/\tau \geq> 1/T$, no aliasing occurs, thus an inverse channel equalizer compensates for the true channel distortion. The channel equalizer is called fractionally spaced equalizer. Usually, τ is selected at $\tau = T/2$, to have 2/T as a sampling rate.

The impulse response of a FIR equalizer is

$$g_E(t) = \sum_{n=-K}^{K} c_n \delta(t - n\tau) \tag{3.5}$$

and the corresponding frequency response is

$$G_E(t) = \sum_{n=-K}^{K} c_n e^{-j2\pi f n\tau} \tag{3.6}$$

where c_n are the $2K + 1$ equalizer coefficients and K is chosen so that the equalizer recover the length of the ISI: it involves $2K + 1 > L$, where L is the number of signal samples containing ISI. The equalized output signal pulse is

$$q_E(t) = \sum_{n=-K}^{K} c_n x(t - n\tau) \tag{3.7}$$

Hence,we can find the conditions on the $2K + 1$ values with

$$q_E(mT) = \sum_{n=-K}^{K} c_n x(mT - n\tau) = \begin{cases} 1 & : \quad m = 0 \\ 0 & : \quad m = \pm 1, \pm 2, ..., \pm K \end{cases} \tag{3.8}$$

which can be expressed in matrix form as $Xc = q$, where X is a $(2K+1)*(2K+1)$ matrix with elements $x(mT-n\tau)$, c is the $(2K + 1)$ coefficient vector and q is the $(2K + 1)$ column vector with one nonzero elements. We should underline that the FIR zero-forcing equalizer does not completely eliminate the ISI because it has a finite length.

One drawback to the zero-forcing equalizer its that it may enhance greatly the noise. Indeed, for a frequency range where C(f) is small, the channel equalizer $G_E(f) = 1/C(f)$ compensates by placing a large gain in that frequency range. An alternative is to relax the zero ISI condition and select the channel equalizer characteristic such that both of the combined power in the residual ISI and the additive noise at the output of the equalizer is minimized. A channel equalizer that is based on the minimum mean-square error (MMSE) criterium can accomplish the desired goal (Reference [10]).

To elaborate such a filter let us consider the sampled FIR equalizer output:

$$z(mT) = \sum_{n=-K}^{K} c_n y(mT - n\tau) \tag{3.9}$$

The desired response at the output of the equalizer is the transmitted symbol a_m. The error is defined as $z(mT) - a_m$, then the mean-square error is

$$MSE = E|z(mT) - a_m|^2 = E\left[\left|\sum_{n=-K}^{K} c_n y(mT - n\tau) - a_m\right|^2\right] \tag{3.10}$$

$$MSE = \sum_{n=-K}^{K}\sum_{k=-K}^{K} c_n c_k R_y(n-k) - 2\sum_{k=-K}^{K} c_k R_{ay}(k) + E(|a_m|^2) \tag{3.11}$$

where the correlations are defined as

$$R_y(n-k) = E[y^*(mT - n\tau)y(mT - k\tau)] \tag{3.12}$$

$$R_{ay}(k) = E[y(mT - k\tau)a_m^*] \tag{3.13}$$

Of course, the minimum MSE solution is obtained by differentiating the expression behind. We get:

$$\sum_{n=-K}^{K} c_n R_y(n-k) = R_{ay}(k), \qquad k = 0, \pm 1, \pm 2, ..., \pm K \tag{3.14}$$

We get 2K+1 equations for the equalizer coefficients. In practice, the autocorrelation matrix $R_y(n)$ and the cross-correlation vector $R_a y(n)$ are unknown, but they can be estimated by transmitting a test signal over the channel and using the time-average estimates

$$\hat{R}_y(n) = \frac{1}{K}\sum_{k=1}^{K} y^*(kT - n\tau)y(kT) \tag{3.15}$$

$$\hat{R}_a y(n) = \frac{1}{K}\sum_{k=1}^{K} y(kT - n\tau)a_k^* \tag{3.16}$$

instead of the ensemble averages to solve these equations.

3.2 Simulating Linear Equalizers with Matlab

We are going to study in this section two equalizers which are the zero-forcing equalizer and the MSE equalizer and their properties, advantages and defaults.

3.2.1 Zero-Forcing Equalizers

We have elaborated the Matlab program which permits us to simulate the impulse response of a channel. So the input signal is like a dirac. Then, the Matlab program does what we have described in the precedent section, in

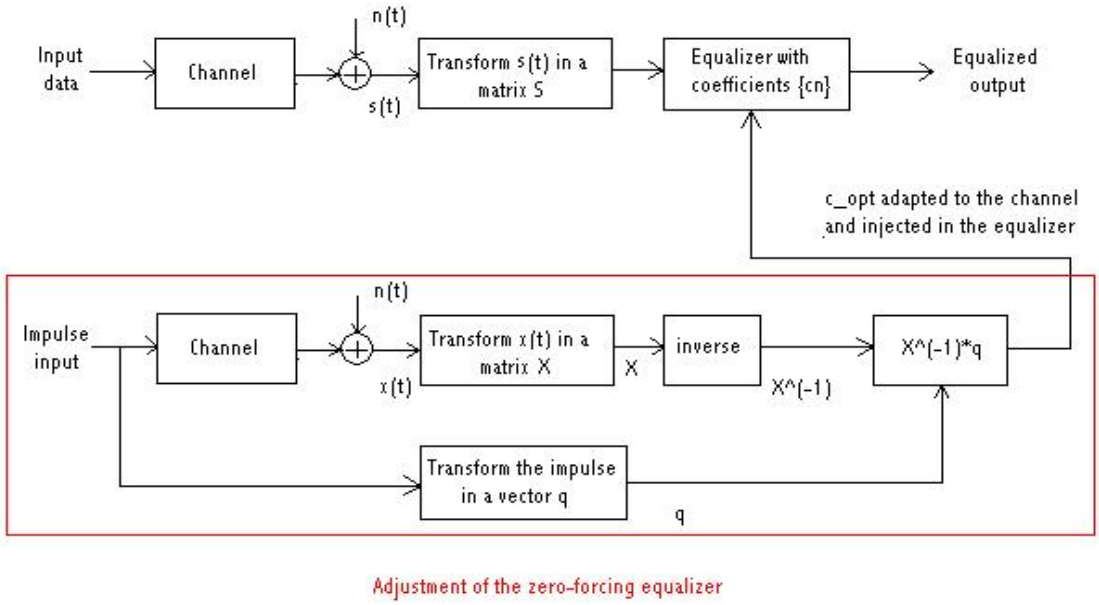


Figure 3.3: Block diagram of a zero-forcing equalizer

theory. It is the same approach but transformed for an optimum Matlab use. The script of the program is given on Annexe A with the meaning of each step and Figure 3.3 gives the block diagram of the zero-forcing equalizer.

The output of the channel is the following function:

$$x(t) = 1/(1 + (2t/T)^2) \tag{3.17}$$

The result of the equalizing is represented on Figure 3.4 which permits the comparison between the input and the output of the equalizer.

We can already see some important characteristics. First, we can see in the program that we take a 5-dimension vector for c_{opt} and for the matrix ,representing the channel, that we take the inverse. It represents in fact the length of the equalizing window. On Figure 3.4, we constat that the signal is equalized on five points: the central point and the two adjacent points close to this one. On the other points, the signal is the original one. So the result is very good because it is obvious that the dirac is detected. We are going to study the influence of some parameters to see how the output of the equalizer evolves.

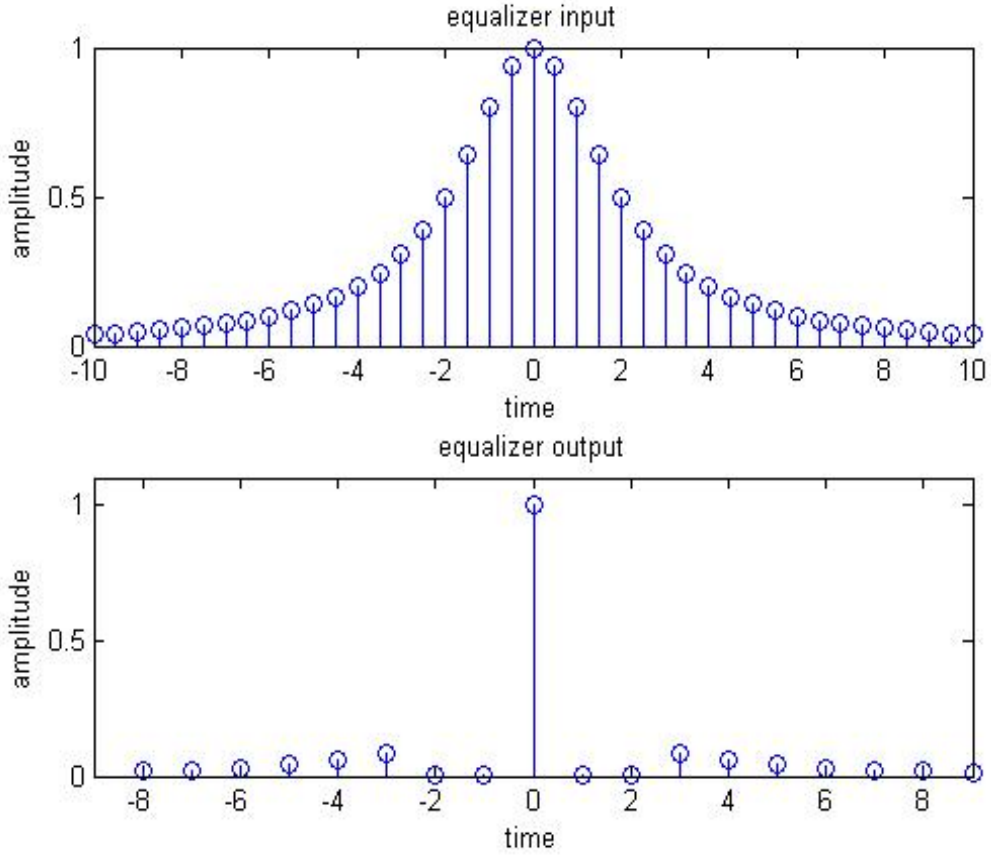


Figure 3.4: Comparison between the input and the output of the zero-forcing equalizer

Influence of the channel

We take an other output for the channel which is:

$$x_{bis}(t) = sinc(t) \tag{3.18}$$

We obtain Figure 3.5 which shows the equalized signal. We can see that the result is totally acceptable and as good as with the other signal. So the zero-forcing equalizer can be adapted to other signals. The limit is that in the program given on Annexe A, the matrix D has to be reversible. But sometimes, the matrix is reversible but is close to singular or badly scaled. In this case, the result is not inaccurate. This happens for example with a function using an exponential, like a gaussian function, gaussian noise for example.

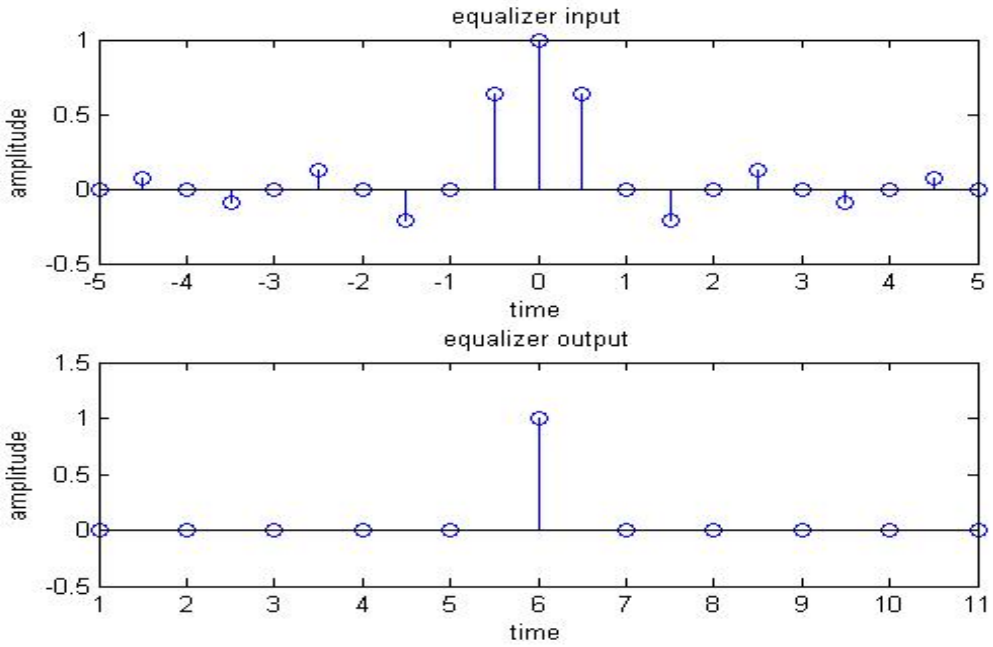


Figure 3.5: Change of the channel for the zero-forcing equalizer

Influence of the length of the equalizing window

In this subsubsection, we take a signal which is going to be more appropriated to see the influence of the window's length. The channel used is:

$$x_w(t) = 1/(1 + (0.4t/T)^2) \tag{3.19}$$

Indeed we have seen that the signals we study are not considered as finite ones. So, the equalizing window is going to be finite. But how can we choose judiciously this length. One of the reason why we can't use infinite signals and windows is due to the complexity it generates because the larger the window is, the higher the number of equations to solve to get a good equalizer is going to be. This equations are given in the theoretical subsubsection. So, when the length of the window is increasing, the difficulty for solving these equations will increase too. Anybody who has done some mathematics know that. So, for the choose of the length of the window, compromises have to be done in order to obtain a good result but without complicate too much the computation. We have again the use of the "cond" function in Matlab. This function evaluates the possible reversibility of a matrix, the matrix D here. This function returns a numeral. The higher this numeral is, the more difficult it will be to inverse the matrix. This numeral gives the ratio of the largest singular value to the smallest. Then when this numeral is increasing, the equalizing becomes difficult. We have noted that it increases with the increasing of the window's length. This is for us the right limit for the length of the window. We can see on Figure 3.6 two equalizing results but with different lengths for the window.

On each diagram are superposed the input and the output signals of the equalizer in order to see better the effect of the equalizer. It is obvious that when we do not take an enough large window, we don't get an equalized

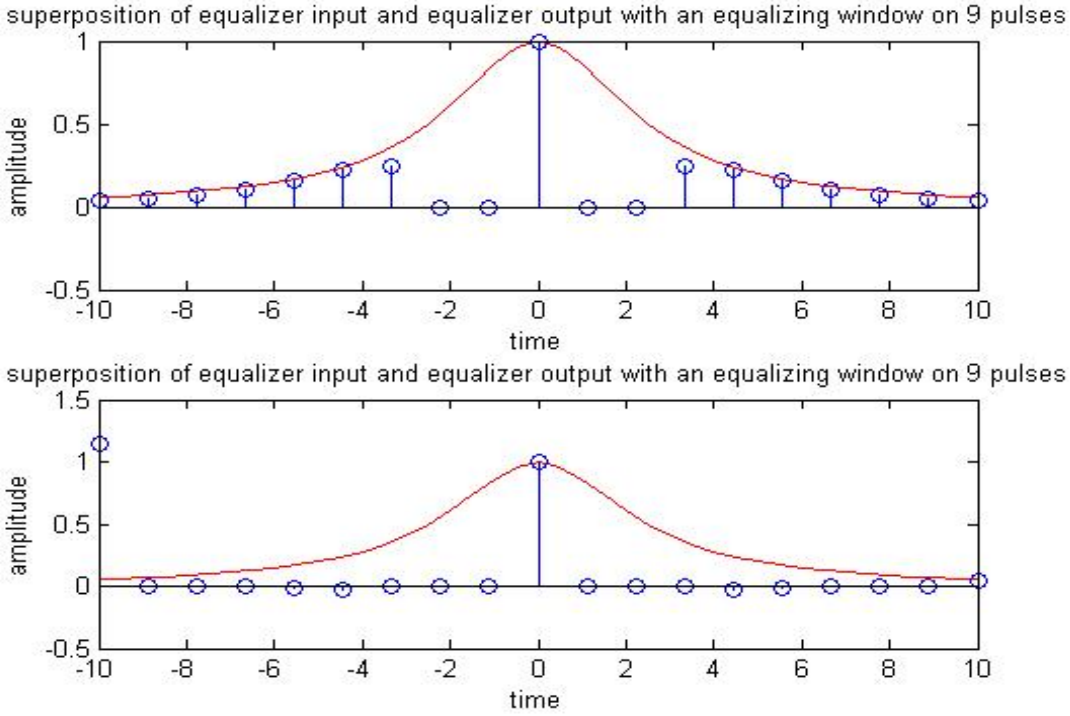


Figure 3.6: Change of the equalizing window for the zero-forcing equalizer

signal which can be considered enough satisfactory. We can see that on the first diagram of Figure 3.6, the first pulses outside of the window are too much important compared to the principal pulse. So we can see here the necessity of increasing the length of the window. When it is possible to do this, we get something like on the second diagram where we obtain quite perfectly the principal pulse, which is the image of the dirac at the input of the system. Finally, the choice of the length is a compromise problem between an acceptable result and a computation not too complex in order to avoid that it would get too much time.

Influence of an additive noise

Now we are going to study the influence of an additive noise on the result given by the equalizer. The channel is x(t), and we take the same Matlab script as on Annexe A. We just add the noise. The intensity of the noise is controlled with the power spectrum of the noise $N_0/2$, so N_0. Higher N_0 is, higher the noise is. So, we are going to increase the noise and to look what are the effects on the output of the equalizer. The thing that have to be known is that the noise is generated with random. So this figure shows one example but an other one with the same Matlab program would show an other noise. But the most importance is the variation of its intensity. So, we can see on Figure 3.7 the evolution of the equalizer's input signal with the evolution of noise intensity. We first take a low noise and we increase it, by increasing N_0.

Noise has really a bad influence on the channel and we can already imagine the problems caused by the noise in order to get back the system's input signal. When the noise is very low, the aspect of the signal at the output

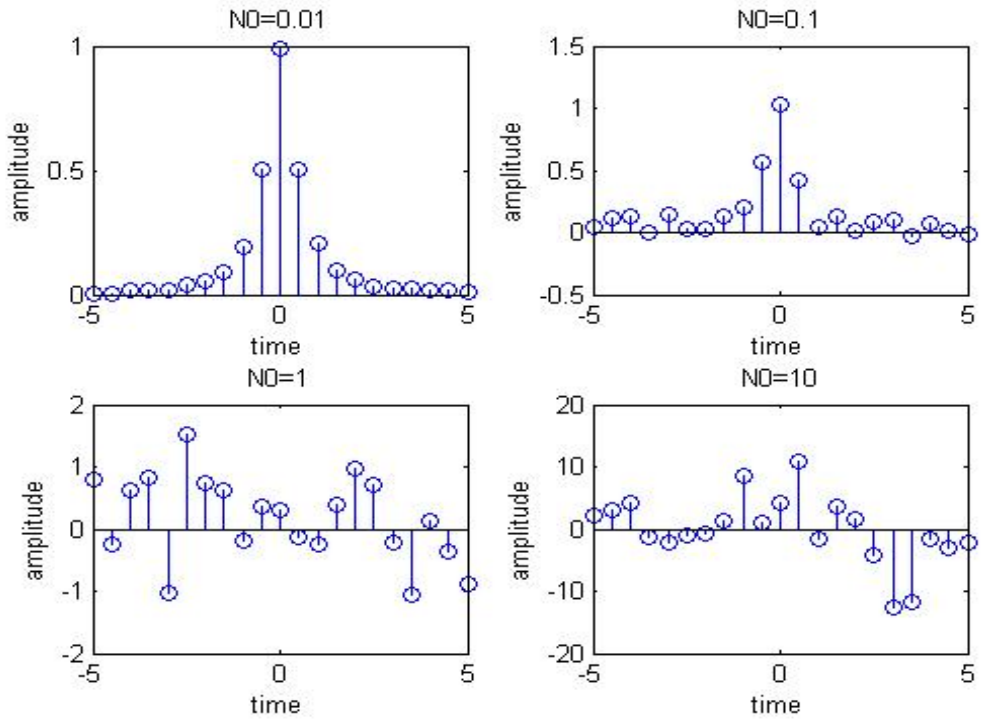


Figure 3.7: Evolution of the equalizer input signal by increasing the noise

of the channel is regular and foreseeable. The noise is going to trouble this. Does the equalizer can correct this? Figure 3.8 gives the result after the equalizing sequence.

We can see on the first diagram, when N_0=0.01, which is a very low noise, that we get an equalized result similar to the no-noise result. So, if the equalizing runs well, we almost must get only our dirac. On the second diagram, we choose N_0=0.1 and the result is still acceptable, even if there is already some small imperfections. The result begins to be really unacceptable and unusable for a value of N_0 of 1. Here it's impossible to determine which signal has been sent. We can simply wonder why we don't just increase the length of the window. But when we do this we encounter the same problem as we dealt with before, that is that we can't equalize because of the channel's matrix representation. We have to get a compromise between the length of the window and the interference pulses created by the noise. But, when the noise is too important, we can't do anything. The fourth diagram, with N_0=10, illustrates the fact that when you increase the intensity of the noise, it is really impossible to get a good result with the zero-forcing equalizer. This joins the theory saying that in zero-forcing equalizer, we increase the gain where we have the pulse we want to obtain. But we increase also the noise in this window. Then, we have to find an other solution, which is the MSE equalizer.

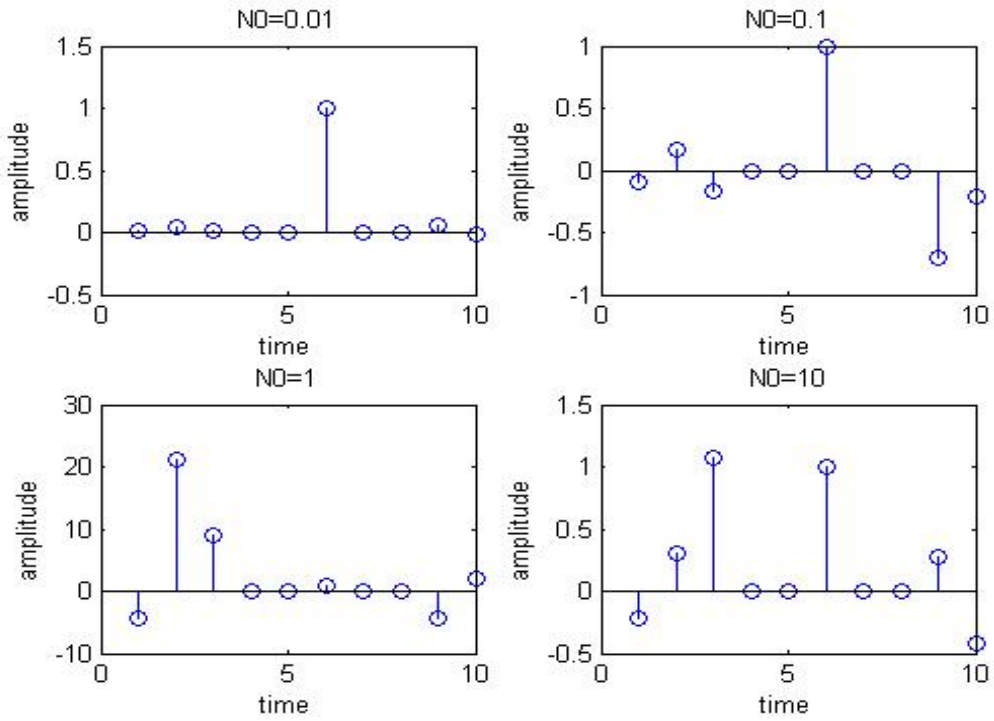


Figure 3.8: Evolution of the equalizer output signal by increasing the noise

3.2.2 MSE Equalizers

Here, we also use the same way as in the theory but there are some mathematic computations to get a form which can be used in the Matlab programs. The big problem is to get a form for R_y which is exploitable. This needs a difficult computation that we are going to explain to get the matrix R_y with the R_y(n-k) given by the theory.

We are going to use the form given by the theory to obtain this form we will use in our Matlab programs which is:

$$R_y(n-k) = E[y*(mT-n\tau)y(mT-k\tau)] \tag{3.20}$$

We consider that y(k)=x(k)+v(k) where x is the channel response and v the additive noise. Then, the computation can be developed like that:

$$\begin{aligned} R_y(n-k) &= E[x*(mT-n\tau)x(mT-k\tau)] + E[v*(mT-n\tau)v(mT-k\tau)] \\ &= R_x(n-k) + R_v(n-k) \end{aligned} \tag{3.21}$$

$$\tag{3.22}$$

with:

$$R_x(n-k) = E[x*(mT-n\tau)x(mT-k\tau)] \tag{3.23}$$

and,

$$R_v(n-k) = E[v*(mT-n\tau)v(mT-k\tau)] \tag{3.24}$$

So, we develop the expression of R_x

$$R_x(n-k) = \sum_{m=-K}^{m=K} x(mT - n\tau)x(mT - k\tau) \tag{3.25}$$

R_y(n-k) is just one element of the matrix R_y, so R_x(n-k) is also one element of the matrix R_x. Then, when you generalize 3.25 to all the matrix R_x, we obtain that:

$$R_x = X^t * X \tag{3.26}$$

where X is the matrix composed by the elements x(mT-nτ) which is the matrixal representation of the impulse response of the channel. Now, we are going to develop the expression of R_v(n-k):

$$\begin{aligned} R_v(n-k) &= E[v * (mT - n\tau)v(mT - k\tau)] \\ &= \Phi_{vv}((n-k)\tau) \qquad (3.27) \\ & \qquad (3.28) \end{aligned}$$

where Φ is the autocorrelation function. So it gives that R_v(n-k)=$N_0/2$ if n=k and 0 else. Thus, we obtain R_v which is a matrix composed by R_v(n-k) elements:

$$R_v = \frac{N_0}{2} I \tag{3.29}$$

Finally, the general form for R_y is:

$$R_y = X^t * X + \frac{N_0}{2} I \tag{3.30}$$

Then, we obtain the block diagram of the MSE-equalizer given on Figure 3.9

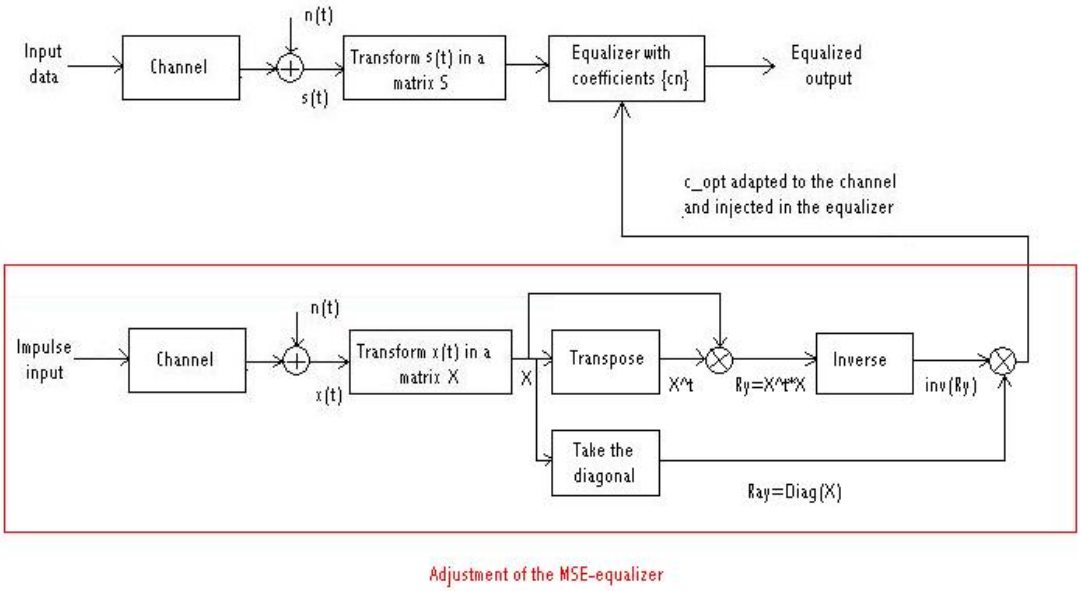


Figure 3.9: Block diagram of a MSE-equalizer

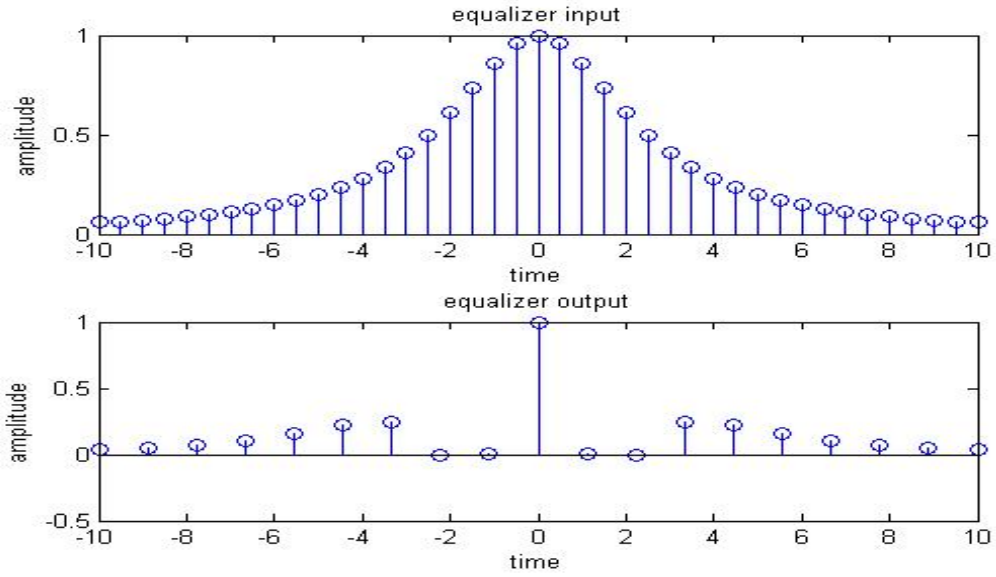


Figure 3.10: Comparison between the input and the output of the MSE-equalizer

This is a form we can now use in our Matlab programs because it is easy for us to get the matrix X when we knox what is the impulse response x(t) of the channel, by taking x(mT-nτ). We can now use it in the Matlab scripts, and the one used to perform MSE equalizers is given on Annexe B.So, we are going to use this script, and to transform it ti study the properties of the MSE equalizer. For each property, we will use the script (Annexe B) and we will notice the changes we do. Figure 3.10 shows the result of the equalization of the signal x(t).

This will be our basic result and we will compare after other signals to this one to discuss about the acceptability of these results. We can see at the moment that the equalizer is very good. For this basic simulation, we get a result which is as satisfying as the one obtained with a zero-forcing equalizer. We really get our dirac back.

Influence of the channel

We are going to use the result given by the same signal used for the zero-forcing equalizer, x_{bis}.

Figure 3.11 gives the equalized result. As we saw with the zero-forcing equalizer, there's no matter of changing the aspect of the channel. The equalizer runs for other channels without changing anything, just the function of the impulse response of the channel. We get our dirac as we wanted and as it was previewed.

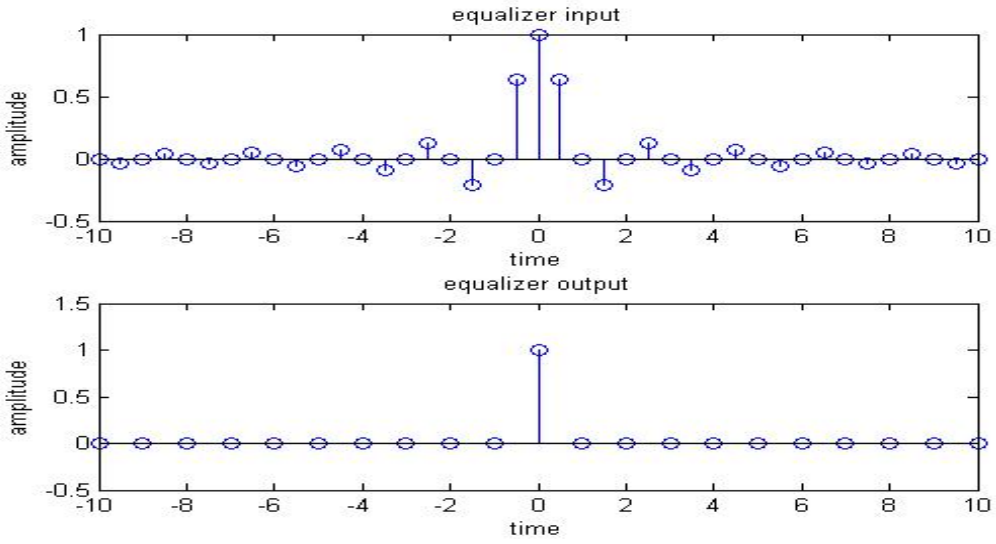


Figure 3.11: Change of the channel for the MSE-equalizer

Influence of the length of the equalizing window

Here, we do exactly the same thing that with the zero-forcing equalizer. We try to increase the window in order to get a better and better result. The problems we encounter are exactly the same. Indeed, the more we increase the window, the more the computations are complicated but the more the result is good.

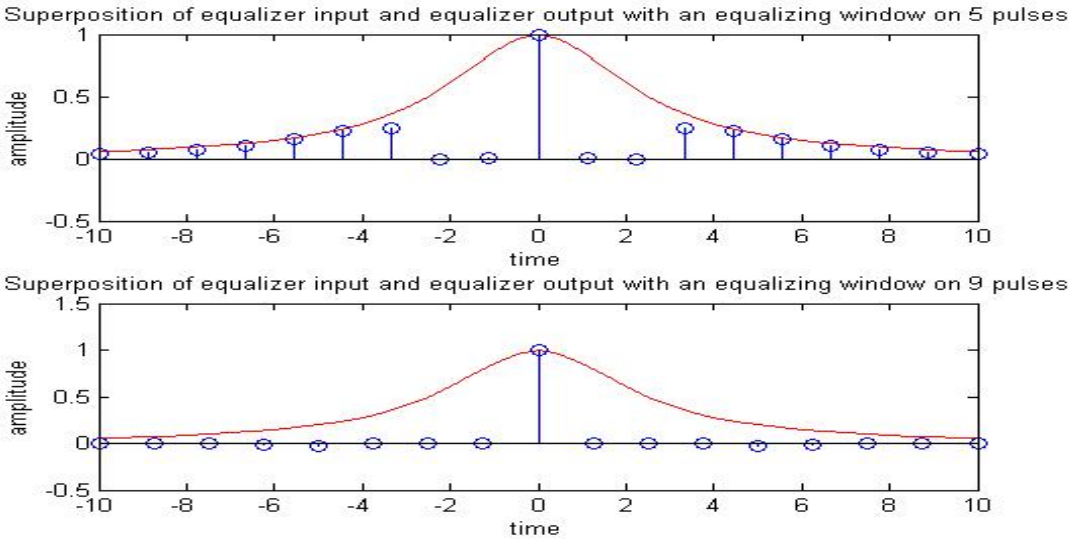


Figure 3.12: Change of the equalizing window for the MSE-equalizer

We can see on the Figure 3.12 the results for two different windows. We have superposed the equalizer input to the equalizer output to show really what is the effect of the equalizer in the window concerned. We have taken two cases where the equalizing works good. So we can see that the MSE equalizer works but does not improve anything

for the problem of the window.

Influence of the noise

Here, we are going to show the real advantages of the MSE equalizer. In fact, we said and showed at the end of the precedent subsubsection that the zero-forcing equalizer was not satisfying for an input signal containing noise. We said also that we had to find a solution and that this solution was the MSE. We are now going to prove it. The intensity of the noise is again controlled with its power spectrum which is N_0.

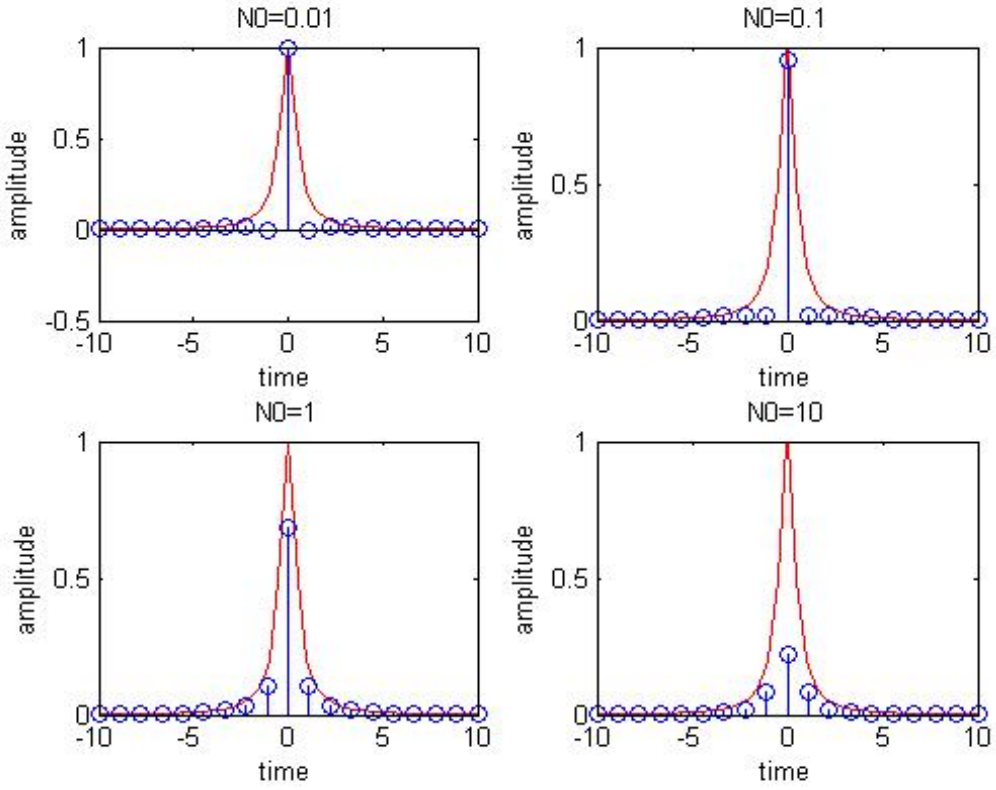


Figure 3.13: Evolution of the equalizer output by increasing the noise for the MSE-equalizer

Figure 3.13 compares the results given in the output of the equalizer with different power noises. In fact we take the same powers as with the zero-forcing equalizer in order to get comparable results. So, for the first diagram, with N_0=0.01, the results are quite the same. Indeed, in the both cases, the noise is too weak to constrict really the transmission of the signal. But with the following diagram, and N_0=0.1, we see that the MSE equalizer is obviously better than the zero-forcing one. This is confirmed with the two last diagrams, and N_0=1, N_0=10. The results are really better than with the zero-forcing equalizer. But we don't have perfect results as we had with N_0=0.1 for example. We can't get a perfect dirac at the output when the noise's power increases too much, it's normal.

Finally we can say that the MSE equalizer is really a good one. It is obviously not a perfect equalizer. But when we consider other channels than perfect ones which begins to be the case as soon as we add noise, the results

can't be perfect. This is true with everything which concerns telecommunications. The more we try to perform something closer to the reality, the more we go far from what we would want in theory. The problem is to make compromises. Here, we are just looking at few factors and we already have to make some compromises.

We have seen that the tap coefficients of a linear equalizer can be determined by solving a set of linear equations, in both cases zero-forcing equalizer and MSE equalizer. The general form of the set of equations we have to solve is:

$$Bc = d, \tag{3.31}$$

where B is the matrix representing the channel, c the vector of equalizer coefficients is a vector with one non-zero element. In order to get the best equalizer, we have tried to obtain:

$$c_{opt} = B^{-1}d \tag{3.32}$$

But, in the reality, this is not what happens. Indeed, we avoid to compute the inverse matrix of B, B^{-1}.We saw in this section that this compute was really a problem and that it has a big influence on equalizer's performances. We are now going to study the adaptive equalizer which will permit us to be closer to the reality.

Chapter 4

Simulation of Adaptive Linear Equalizers on Matlab

4.1 Theory on Adaptive Linear Equalizers (Reference [11], [12] and [13])

We have seen in the precedent subsection that the set of linear equations to solve in order to determine the optimum equalizer may be expressed with matrix:

$$Bc = d, \tag{4.1}$$

where B is a (2K+1)×(2K+1), c is a column vector representing the 2K+1 equalizer coefficients, and d is a (2K+1)-dimensional column vector. So,the matrix solution is:

$$c_{opt} = B^{-1}d \tag{4.2}$$

In practical implementations of equalizers, we use adaptive equalizers, which are really required for example when we have channels whose characteristics change with time. It permits to avoid the explicit computation of the inverse of the matrix B.The procedure is iterative with a steepest descent method. We start with an arbitrary coefficient vector c_0 for c that we are going to optimize. If we take the precedent example of MSE criterion, c_0 corresponds to a point on the quadratic MSE surface in the (2K+1)-dimensional space of coefficients. The derivative of the MSE, the gradient vector g_0 is computed at this point and each tap coefficient is changed in the direction opposite to its corresponding gradient component. Then the gradient vector g_k is:

$$g_k = Bc_k - d, \tag{4.3}$$

where:

$$c_{k+1} = c_k - \Delta g_k \tag{4.4}$$

Δ is chosen to be a small positive number and $g_k \longrightarrow 0$ when k $\longrightarrow \infty$ and naturally $c_k \longrightarrow c_{opt}$. We can understand that c_{opt} will be never reached but approached more and more if you raise the number of iterations. When we use an adaptative channel equalization for a time varying channel, c_opt varies with time because the coefficients of B vary also. So the iterative method described previously uses estimates of the gradient components which gives the same equation as 4.4 but with estimated variables:

$$\hat{c}_{k+1} = \hat{c}_k - \Delta \hat{g}_k \tag{4.5}$$

In the case of MSE criterion, we can define g_k as:

$$g_k = -E(e_k y_k*) \tag{4.6}$$

which gives with approximations:

$$\hat{g}_k = -e_k y_k*, \tag{4.7}$$

where e_k is the difference between the desired output (a_k) at the kth instant and the actual output z(kT)(z_k), and y_k denotes the column vector of 2K+1 received signal values contained in the equalizer at time instant k.We have:

$$e_k = a_k - z_k \tag{4.8}$$

Finally the adaptative algorithm for optimizing the tap coefficients for MSE criterion is:

$$\hat{c}_{k+1} = \hat{c}_k + \Delta e_k y_k* \tag{4.9}$$

We can see on Figure 4.1 the schema of the linear adaptive equalizer based on the MSE criterion.

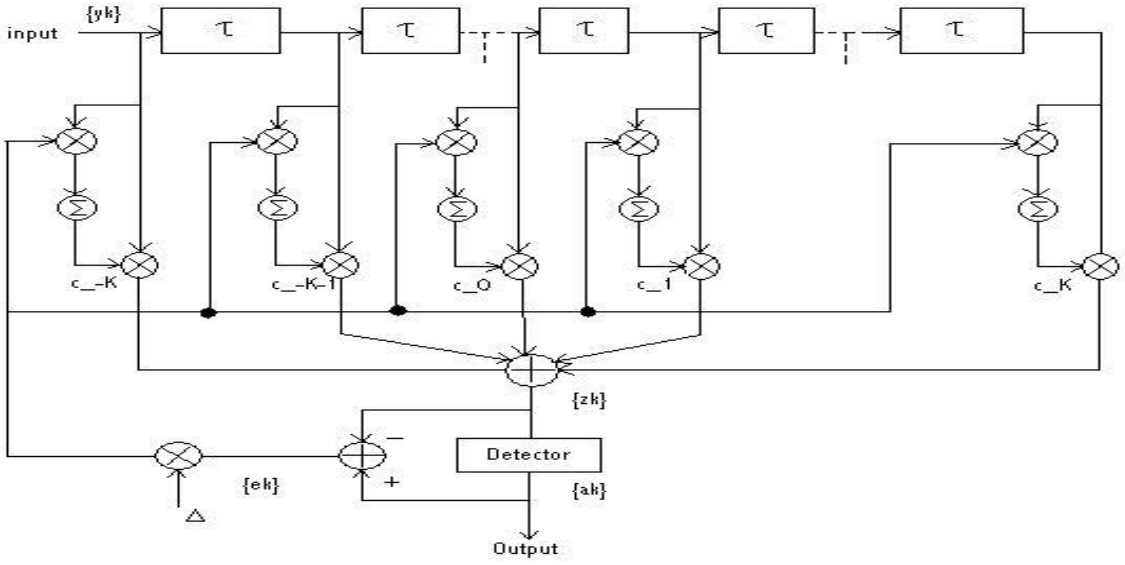


Figure 4.1: Linear adaptive equalizer based on the MSE criterion

We just have to choose Δ now. We can admit that a good value of Δ in order to ensure convergence and good tracking capabilities in slowly varying channels is:

$$\Delta = \frac{1}{5(2K+1)P_R}, \tag{4.10}$$

where P_R denotes the received signal-pulse-noise power, which can be estimated from the received signal.

4.2 Simulating Adaptive Linear Equalizers with Matlab

The basic script of the adaptive equalizer we are going to use in this section is given on Annexe C.

4.2.1 Adjustment of some Parameters and Properties of Adaptive Equalizer

The first thing we have to say is that the adaptive equalizer we have built is a derivative of the MSE equalizer we introduced in the precedent section. It is also possible to build one derived from the precedent zero-forcing equalizer. So here, being derived from a MSE equalizer, we are going to have the same properties as this one. These properties are principally about the influence of the channel and of the noise. For the channel, it is normal because we have said in the theoretical part that this equalizer is used when the channel evolutes during the time. So, it is obvious that you can change your channel as many times as you want, it will adapt at each time.

We are now going to detail a bit the way of thinking of such an equalizer. Imagine we have a channel whose characteristics have just changed. So, we have a new channel and the equalizer at the output of the channel don't have the good coefficients to treat the signal. Then, we are going to send a lot of information (composed of diracs in general) and the equalizer is going to follow an iterative process to determine its c_{opt} which are the coefficients of the equalizer, adapted for the treatment of the new channel. We have to know that we don't really get the c_{opt} but that we approach them. Finally, when we have theses coefficients, we can send the signal we want to send which is going to be well treated by the equalizer, if the channel has not changed again. If it happens, the iterative process has to be done again. So, in the concrete case of a communication, it is easy to understand that the research of the good coefficients for the equalizer must not be too long because the channel can vary more or less rapidly.

It is there that we can see the importance of some factors given in the script of Annexe C, which are N, the length of the information sequence we send to regulate the equalizer, so the number of iterations; Δ, which is the step of our iterative process and Num-of-realizations. All these factors have to be chosen on two criterions: the whole sequence must not be too long for the channel not to change during the sequence, but not too short in order to get a satisfying approximation for the optimum coefficients of the equalizer. If we don't have a too precise approximation, we won't be able to treat our signal after. So, we are going to see how to choose these three parameters in one case of channel, the one given as x(t) in the precedent chapter.

Choice of the parameter Δ

With our Matlab program, the way to know if the parameters are well chosen is to look at the diagram of the Mean square error. Indeed, it gives the value of the error for the values of the number of iterations. We are going to see the influence of Δ on this curve.

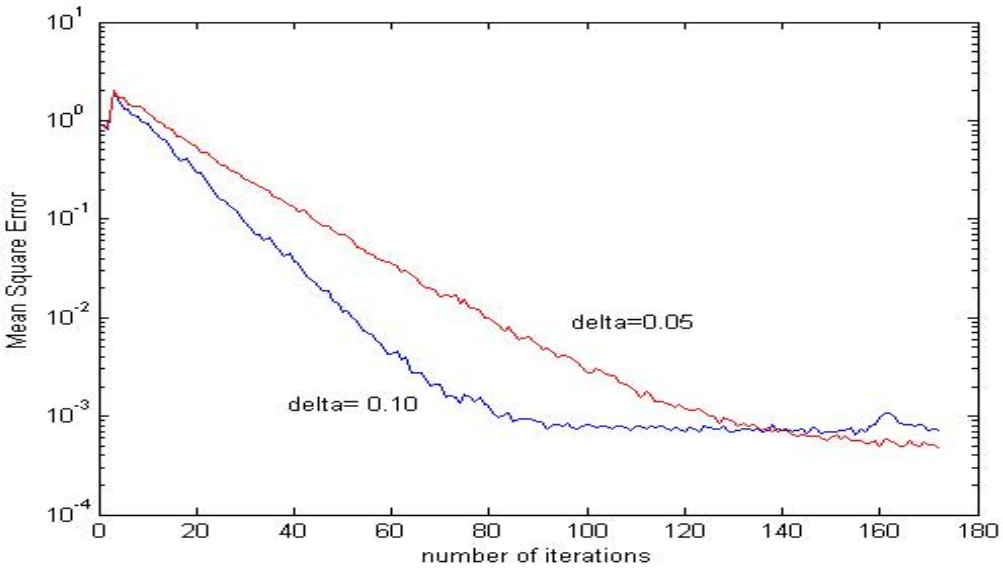


Figure 4.2: Influence of the parameter Δ on the performance of the adaptive equalizer

Figure 4.2 shows the superposition of three curves for two values of Δ, N and Num-of-realizations are stated with respective values 180 and 1000. We can see that when we increase Δ, the Mean square error decreases, which is good because we have a more precise result. But, the time to get a result with the Mean square error minimum is longer. So, we have again a compromise to do between precision and computation time. We have seen in the theoretical section that we can empirically set Δ with the value given by the equation 4.10. This is the value we are going to keep for the other simulations, as a good compromise.

Choice of the parameter N, number of iterations

We are going to work with the same diagram (fig. 4.2) as before but with only one curve, which is the one obtained by taking the value of Δ we have posed.

We can see that the curve has a Mean square error limit when we increase the number of iterations. The aim is to get a value of N which permits to have this limit as the value of the Mean square error, but taking the smaller one, in order again to reduce the time of computation. So, we are going to choose N graphically. Thus, we can say that N=120 seems to be a good value. In fact, we have to take a bigger value to have results we can see easier. Then, we pose N=180 in this case.

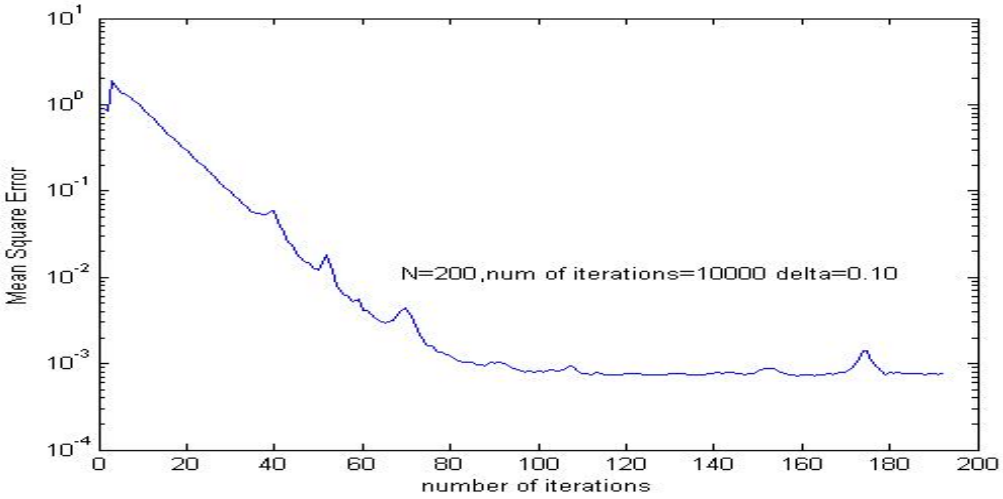


Figure 4.3: Influence of the parameter N on the performance of the adaptive equalizer

Choice of the number of realizations (Num-of-realizations)

The number of realizations is a parameter which is added in order to affine the precision of our equalizer. Indeed, the following figure (Figure 4.4) illustrates very well the difference we can have if we just do our iterations once.

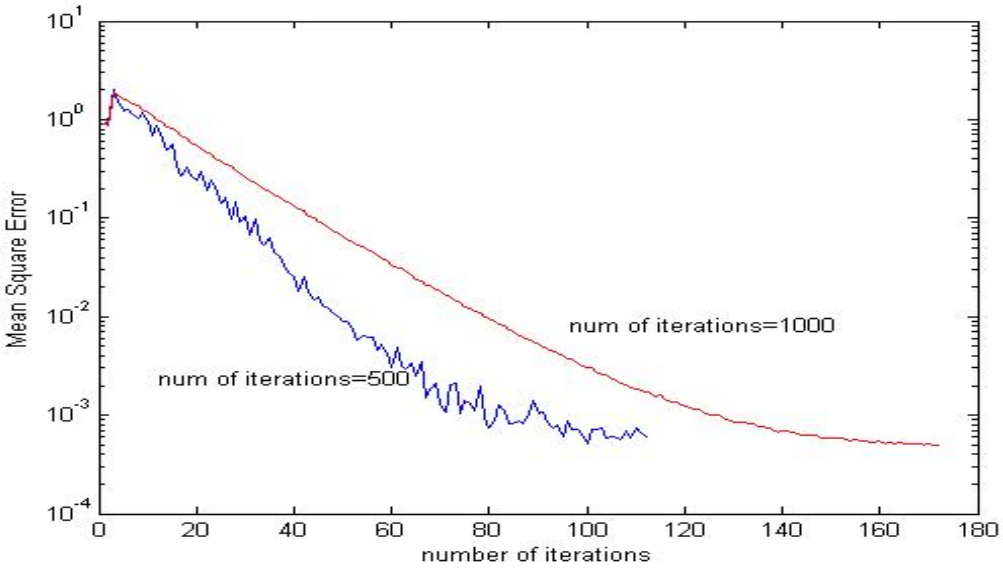


Figure 4.4: Influence of the number of realizations on the performance of the adaptive equalizer

The curve of the Mean square error is not enough smooth. So, we can sometimes have a value of the Mean square error which is not acceptable. The figure 3.4 shows this very well, the first diagram is obtained with five hundred realizations and the second with a thousand. On the contrary of the first, we can see on the second one that there are no important pick. So, the approximation of the coefficients of the equalizer will be better. But,

even if the result is good with a value of 1000, we can't forget that equalizing is a compromise matter. So, we have to test with other values if, for example, we can reduce this value in order to get computation times smaller. After a lot of tests, we can say that some hundreds for this number is good. So, 1000 is still a good compromise. Its the value we have chosen for the following experiments with this channel and this equalizer.

Finally, to resume, we remember the values chosen for this channel:

$$\Delta = \frac{1}{5(2K+1)P_R}, \tag{4.11}$$

as given in the equation 3.10; N=180 and Num-of-realizations=1000.

4.2.2 Application of Adaptive Equalizer with a Time-Varying Channel

Indeed, a very useful characteristic of the adaptive equalizer is that it is really good for working with a channel which not a constant one. This equalizer adapts itself to the new channel. The principle is that it computes every time its own coefficient to adapt them to the channel. The first limit is that the successive channel can't be totally different for two reasons which are again about a compromise. First, the equalizer, when the channel changes, computes the new coefficients using the old one to begin the computation. So, if the new channel is totally different, the new coefficients will be also totally different from the precedents. So, it is not sure, even if the equalizer is powerful, that it can computes the new coefficients. This is linked with the second limit. Since the beginning or our study on equalizers, we have seen that it is important to consider a compromise with the last of the computation. So, if the two channels are very different, the time of computation will be very long and the channel can change an other time during the computation. Finally, we have to constat the simple fact that, in the reality, the variations of the channels are not total. A variation of the channel is for example due to a small variation of temperature, which is not going to modify completely the channel. So, it's no use developing equalizers we can use for big variations. Then, when we study time varying channel, we are going to give some limits for the variation of the channel. So, we quite know how the channel is going to be. Then, we can adjust the parameters Δ, N, Number of realizations for the more complex channel which means that we are going to use in our Matlab program the values for these parameters which are the bigger. After a lof of essays, we have determine that, for the example we are going to treat, these parameters have to be chosen as:

$$N = 200 \tag{4.12}$$

$$Numberofrealizations = 1000 \tag{4.13}$$

$$\Delta = 0.009 \tag{4.14}$$

The way of equalizing with a time-varying channel is done step by step.

Normal equalization

In this part, we have data which is sent in a channel and which is equalized with the equalizer regulated for the channel of the beginning of the experiment.

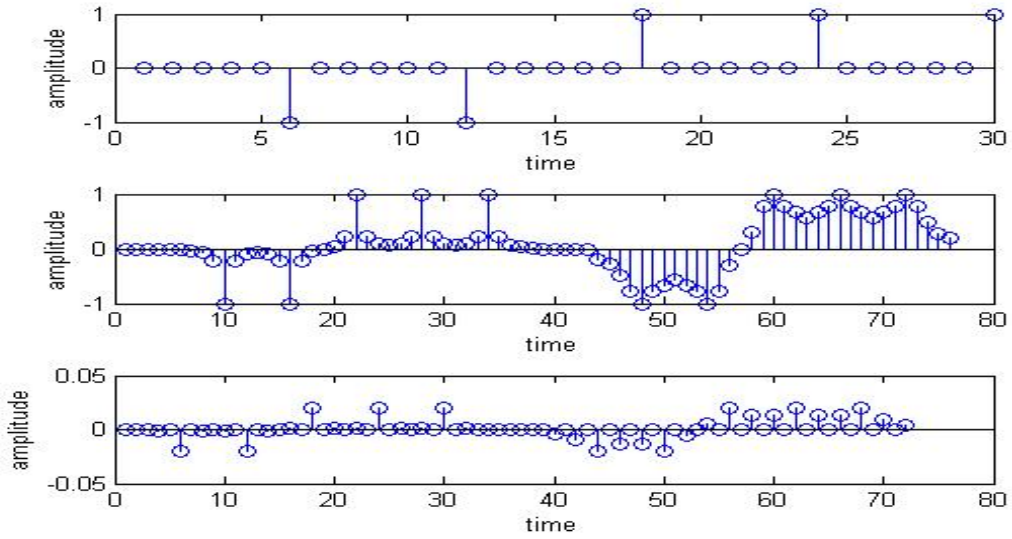


Figure 4.5: Diagrams of the input, the output of the channel an,d the output of the equalizer

Figure 4.5 gives the three diagrams of the input, the output of the channel and the result after equalizing. We can see first that the variation of the channel we have chosen is small, as we said before. But, we can see on the equalized diagram that the equalization seems to be good at the beginning but that it becomes bad. This correspond to the moment when the channel varies. So, we can see there that the equalizer is not adapted to the channel. The following steps are going to explain how does the equalizer do for adjusting its coefficients.

Thresholding and normalizing step

The aim is to detect when there is an error. So, we take the equalized signal and we introduce a threshold: everything which is inferior to a certain value is equal to zero and everything which is over is equal to one. So the diracs are normalized to one.

Figure 4.6 shows a diagram after this step. Thus we can see that we obtain a signal similar to the input, a series of diracs. But the problem is that if a dirac has been too much altered by the channel, its amplitude is going to be under the threshold and then after this step, we won't have it anymore. In fact, it's what interests us, to detect when there is an error.

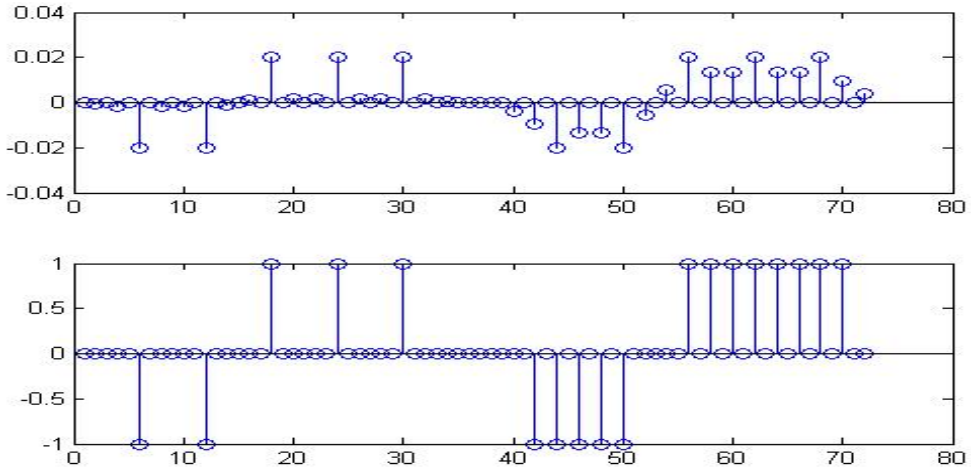


Figure 4.6: Diagrams obtained after the threshold and the normalization

Error step

Now, we want to say when there is an error in order to know when the equalizer has to be adjusted. So, we do the difference between the normalized and thresholded signal and the input. It is the normal way in the reality, when we want to adjust the equalizer, we sent a sequence that is known by the receiver. So, when we do the difference, the errors are going to be detected.

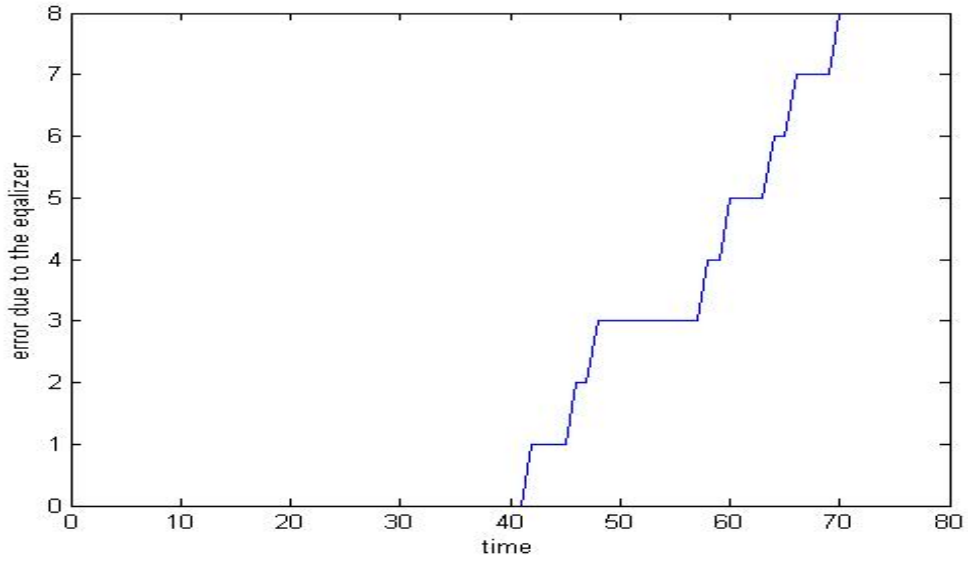


Figure 4.7: Diagram of the error

Figure 4.7 shows a diagram of the error. This one is an increasing function because we tot up the error. We can see that it is easy to choose a threshold from which we say that the error is not acceptable anymore. It's what we

do and from that, this message is sent to the equalizer which now knows that its coefficients are not good enough. So, it's going to readjust its coefficients with the same way that he did for the first channel.

Final step: re-equalizing

Now, we have the good coefficients for the channel. It is important to say that the equalizer is not going to analyze the whole sequence. The sequence (the beginning) which was successfully transmitted has been kept like this and the equalizing process has started again from the point where we detected the error.

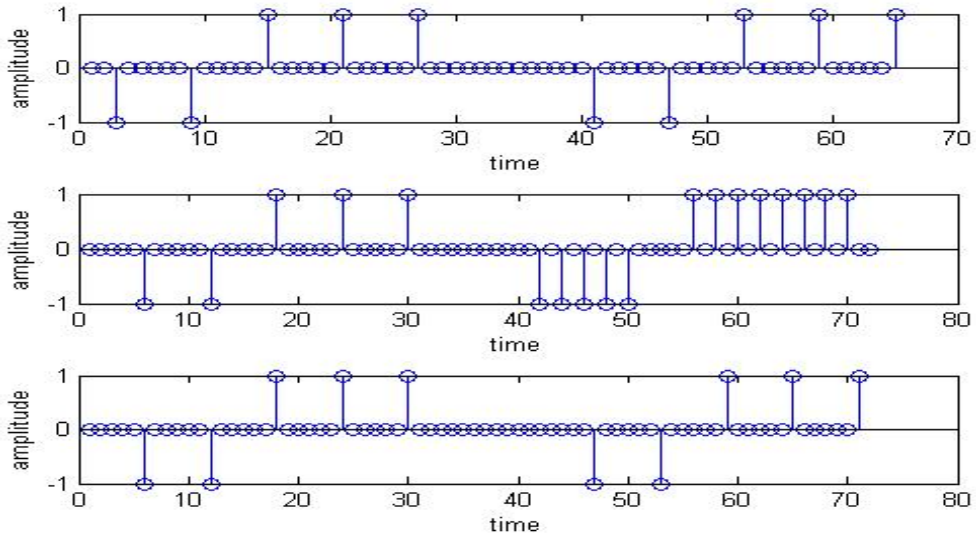


Figure 4.8: Diagrams to compare the input, the output after the first equalization and the final result

Figure 4.8 shows three diagrams with the input, the first output of the equalizer and the final equalized signal. We can see the evolution and the use of such a filter.

4.2.3 Study of a Realistic Case

The environment

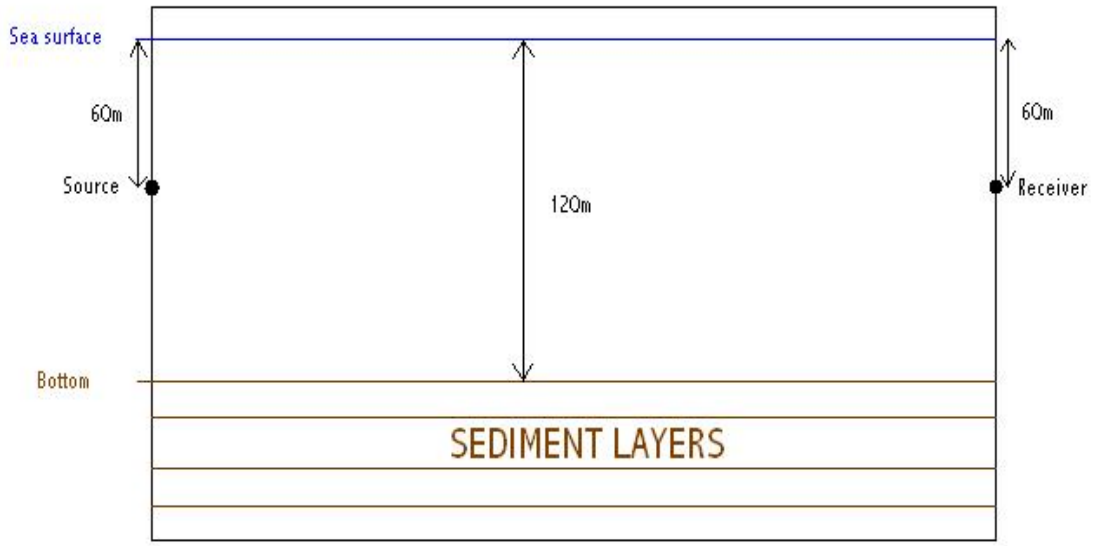


Figure 4.9: The Environment

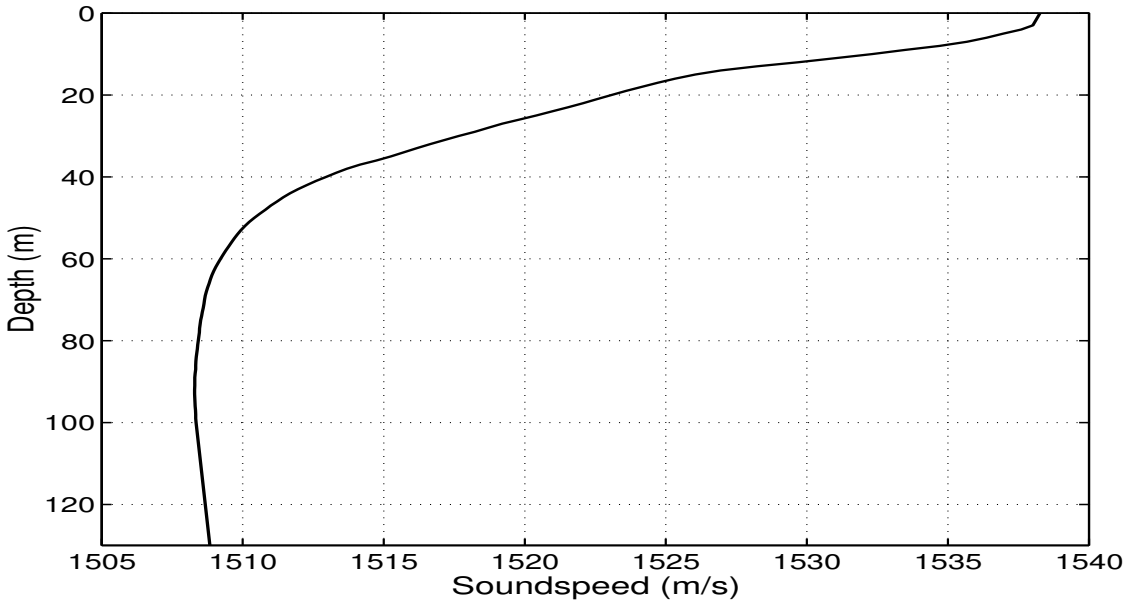


Figure 4.10: Environment: the profile of the sound speed

Chapter 5

Conclusion

Annexe A

Matlab basic script for the zero-forcing equalizer

```
clear all;
close all;
echo on
T=1;
N0=1;
Fs=2/T;                                          %sampling frequency
Ts=1/Fs;
t=-10*T:T/2:10*T;
x=1./(1+((0.5/T)*t).^2);                         %input signal
stem(t,x)
X=[]
for m=-2:1:2
    for n=-2:1:2
  X(m+3,n+3)=1./(1+((0.5/T)*(m*T-n*T/2)).^2);
 end;
end;                                             % creation of the matrix X
%D=D+N0/2*eye(5)
K=inv(X)                                         %inverse of the matrix X
c_opt=K*[0 0 1 0 0]'                             %computation of the
                                                  equalizer coefficients
equalized_x=filter(c_opt,1,[x 0 0])
figure
stem(equalized_x)
equalized_x=equalized_x(3:length(equalized_x));
figure
stem(t,equalized_x)
for i=1:2:length(equalized_x),
  downsampled_equalizer_output((i+1)/2)=equalized_x(i);
end;                                             %downsampling
figure
s=size(downsampled_equalizer_output)
t=linspace(-10*T,10*T,s(2))
stem(t, downsampled_equalizer_output)
```

Annexe B

Matlab basic script for the MSE-equalizer

```
clear all;
close all;
echo on
T=1;
X=[]
for m=-2:1:2
    for n=-2:1:2
  X(m+3,n+3)=1./(1+((0.4/T)*(m*T-n*T/2)).^2);
 end;
end;                                                    %creation of the matrix
X
Y=X'*X;
N0=0.01;                                                % assuming that N0=0.01
Ry=X+(N0/2)*eye(5);                                     %creation of the matrix
Ry
Riy=diag(D);                                            %creation of the matrix
Riy
c_opt=inv(Ry)*Riy;                                      % optimal tap
coefficients
% find the equalized pulse...
t=-10:1/2:10;
x=1./(1+(0.4*t/T).^2);
% sampled pulse
equalized_pulse=conv(x,c_opt);
% decimate the pulse to get the samples at the symbol rate
decimated_equalized_pulse=equalized_pulse(5:2:(length(equalized_pulse)-3));
s=size(decimated_equalized_pulse);
t2=linspace(-10*T,10*T,s(2))
subplot(211)
plot(t,x,'red')
hold on
stem(t2, decimated_equalized_pulse)
```

Annexe C

Matlab basic script for the adaptive equalizer

```
clear all;
echo on
N=180;                                  %number of iterations
T=1;
K=4;                                    % length of the information sequence
t=-6:1:6;
x=1./(1+(2*t/T).^2);                    %input signal
sigma=0.01;
delta=0.1;                              %step parameter
Num_of_realizations=1000;
mse_av=zeros(1,N-2*K);
for j=1:Num_of_realizations,
                                        %compute the average over a number of
                                        %realizations
                                        % the information sequence

  for i=1:N,

    if (rand<0.5),
      info(i)=-1;
    else
      info(i)=1;
    end;
    echo off;

  end;
  if (j==1);          echo on; end;


  y=filter(x,1,info);                    % the channel output
  for i=1:2:N,
    [noise(i) noise(i+1)]=gngauss(sigma);
  end;

  y=y+noise;
                                         % now the equalization part follows
  estimated_c=[0 0 0 0 1 0 0 0 0 ];      % initial estimate of ISI
  for k=1:N-2*K,
    y_k=y(k:k+2*K);
    z_k=estimated_c*y_k.';
    e_k=info(k)-z_k;
    estimated_c=estimated_c+delta*e_k*y_k;
    mse(k)=e_k^2;                        %Mean Square Error
    echo off;
 end;
 if(j==1);echo on; end;

    echo on;
    mse_av=mse_av+mse;
    echo off;
 end;
 echo on;
mse_av=mse_av/Num_of_realizations;
semilogy(mse_av)
gtext('delta= 0.10')
```

Bibliography

[1] North, D.O.. *An Analysis of the Factors Which Determines Signal/Noise Discrimination in Pulse-Carrier Systems*. RCA Tech.: Report No.6 PTR-6C, 1943.

[2] Cahn, C.R.. *Combined Digital Phase and Amplitude Modulation Communication Systems*. IRE Trans. Commun. Syst., September, 1960. Vol. CS-8,pp. 150-155.

[3] Hancock, J.C. and Lucky, R.W.. *Performance of Combined Amplitude and Phase-Modulated Communication Systems*. IRE Trans. Commun. Syst., December, 1960. Vol. CS-8, pp. 232-237.

[4] Gersho, A. and Lawerence, V.B.. *Multidimensional Signal Constellations for Voiceband Data Transmission*. IEEE J. Selected Areas Commun., September, 1984. Vol. SAC-2, pp. 687-702.

[5] Davenport, W.B.Jr and Root,W.L.. *Random Signals and Noise*. New York:Mc Graw-Hill, 1958.

[6] Van Trees, H.L.. *Detection, Estimation, and Modulation Theory,Part I*. New York: Wiley, 1968.

[7] Gerst, I. and Diamond, J..*The Elimination of Intersymbol Interference by Input Pulse Shaping,*. Proc.IRE, July, 1961. Vol. 53.

[8] Smith, J.W.. *The Joint Optimization of Transmitted Signal and Receiving Filter for Data Transmission Systems*. Bell Syst. Tech.J., December, 1965. Vol. 44, pp. 1921-1942

[9] Lucky, R.W.. *Automatic Equalization for Digital Communications,*. Bell Syst. Tech. J., April, 1965. Vol. 44, pp. 547-588.

[10] Widrow, B.. *Adaptive Filters: Aspects of Network and System Theory, R.E. Kalman and N.DeClaris (eds.)*. New York: Holt Rinehart and Winston, 1970.

[11] Lucky, R.W.. *Techniques for Adaptive Equalization of Digital Communication*. Bell Syst. Tech. J., 1966. Vol. 45, pp. 255-286.

[12] Proakis, J.G.. *Adaptive Digital Filters for Equalization of Telephone Channels*, IEEE Trans. Audio and Electroacoustics, June, 1970. Vol. AU-18, pp. 195-200.

[13] Adaptive equalizer exercise[en ligne]. Helsinki University of Technology, Laboratory of Signal Processing, April 27, 2000. Available on web:¡http://keskus.hut.fi/opetus/s38411/¿.

[14] Kopka, H. and Daly, P.W.. *A guide to latex*. Third edition. Harlow: Pearson Education, 1999. 600 p. ISBN 0-201-39825-7.