Technical University of Denmark

DTU Fotonik

Bachelor Thesis

Channel Equalization in Radio-over-Fiber Transmission Links

Author:
Miguel Iglesias Olmedo

Supervisors:
Idelfonso Tafur Monroy
Neil Guerrero González

February 21, 2010

Contents

1 Summary

This project is about the channel equalization in radio over fiber transmission links. In particular, we focus on minimizing the effect of additive white Gaussian noise and the dispersion caused by transmission.

To solve the porblem of channel equalization, filters such as Finite Impulse Response (FIR) or Infinite Impulse Response (IIR), have already been investigated deep enough, thus, our project will be taken from another point of view. The goal of this project is to achieve signal equalzation in the receiver decision module, increasing the performance of the Bit Error Rate curves in digital formats. In order to do that, we will use artificial intelligence algorithms.

This document is divided into four chapters. First of all, an introduction to the **theoretical** background used in this project will be described. Then, we will define the **problem**, and possible solutions. Once a solution is chosen, we will start discussing the **implementation**, in order to justify the results obtained in the computer **simulations**.

Our systems under study uses digital modulations theory [2]. In concrete, we will be focusing on Phase-shift keying modulations, since it is going to be the base for our project.

The channel is one of the most important bulding blocks on our scheme, because it generates the signal distorsions that we want to mitigate. Thus, fiber channels model will be reviewed in order to extract the features that will produce the noise and how it influences the transmitted signal [1]. This noise can be classified as additive white Gaussian noise (AWGN) and the dispersion.

Our proposed and studied solution is based on artificial intelligence (AI) algorithms in the decider module of the receiver: Expectation Maximization (EM) and Hidden Markov Models (HMM) [3]. These two algortihms learn from the channel behaviour in the system and adapt the decision boundaries to make more correct decisons.

We performed computer simulaitons of our proposed solution, and the results proved an important improvement over the current systems. Our proposed approach is not only capable of significantly reducing the BER of the system, but it is also able of correcting other phenomena's such constellation phase rotation or changes in the distribution of the received symbols in the constellation.

2 Digital Modulation

Firstly, what do we mean by digital modulation? Typically the objective of a digital communication system is to transport digital data between two or more nodes. In radio communications this is usually achieved by adjusting a physical characteristic of a sinusoidal carrier, either the frequency, phase, amplitude or a combination thereof. This is performed in real systems with a modulator at the transmitting end to impose the physical change to the carrier and a demodulator at the receiving end to detect the resultant modulation on reception.

2.1 Phase-shift keying modulation (PSK)

An alternative to imposing the modulation onto the carrier by varying the instantaneous frequency is to modulate the phase. This can be achieved simply by defining a relative phase shift from the carrier, usually equi-distant for each required state. Therefore a two level phase modulated system, such as Binary Phase Shift Keying, has two relative phase shifts from the carrier, + or - 90°. Typically this technique will lead to an improved BER performance compared to MSK. The resulting signal will, however, probably not be constant amplitude and not be very spectrally efficient due to the rapid phase discontinuities. Some additional filtering will be required to limit the spectral occupancy. Phase modulation requires coherent generation and as such if an IQ modulation technique is employed this filtering can be performed at baseband.

2.2 Quadrature Phase Shift Keyed (QPSK)

Higher order modulation schemes, such as QPSK, are often used in preference to BPSK when improved spectral efficiency is required. QPSK utilizes four constellation points, as shown in figure 1 below, each representing two bits of data. Again as with BPSK the use of trajectory shaping (raised cosine, root raised cosine etc) will yield an improved spectral efficiency, although one of the principle disadvantages of QPSK, as with BPSK, is the potential to cross the origin, hence generating 100% AM.

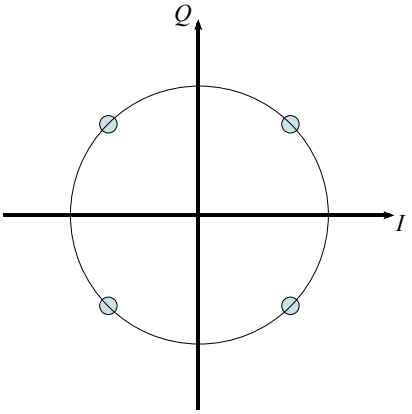

Figure 1: QPSK Constellation

As with BPSK, there are phase ambiguity problems at the receiver and differentially encoded QPSK is used more often in practice.

2.3 Bit Error Rate (BER)

Although QPSK can be viewed as a quaternary modulation, it is easier to see it as two independently modulated quadrature carriers. With this interpretation, the even (or odd) bits are used to modulate the in-phase component of the carrier, while the odd (or even) bits are used to modulate the quadrature-phase component of the carrier. BPSK is used on both carriers and they can be independently demodulated. As a result, the probability of bit-error for QPSK is the same as for BPSK:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \tag{1}$$

For the general M-PSK there is no simple expression for the symbol error probability if $M>4$. Unfortunately, it can only be obtained theoretically. This may be approximated for high M and high E_b/N_0 by:

$$P_b = \frac{1}{K} \cdot Q\left(\sqrt{\frac{2E_b}{N_0}} \sin\left(\frac{\pi}{M}\right)\right) \tag{2}$$

However, in order to achieve the same bit-error probability as BPSK, QPSK uses twice the power (since two bits are transmitted simultaneously). The symbol error rate is given by:

$$P_s = 1 - (1 - P_b)^2 \tag{3}$$

3 Fiber channels

Optical fiber channels are rapidly gaining an important foothold in the global telecom network, partly due to cost-effective, fully domestic production of a variety of fiber optic cables and systems.

The fiber is a glass or plastic fiber that carries light along its length. It works by sending light pulses through the fiber core.

There are two kinds of fiber: Multi-mode fiber, which allow you to carry the light in more than one way, what means that they are not going to arrive at the same time, and mono-mode fiber, more expensive but better because you can carry the data at a greater distance. In our case we will use the second one.

3.1 OSNR

Optical Signal to Noise Ratio (OSNR) [dB] is the measure of the ratio of signal power to noise power in an optical channel. OSNR is important because it suggests a degree of impairment when the optical signal is carried by an optical transmission system that includes optical amplifiers.

Optical signal suffers more than only attenuation. In amplitude, spectrally, temporally signal interaction with light- matter, light- light, light-matter-light leads to other signal disturbances such as:

- Power reduction
- Dispersion
- Polarization
- Unbalanced amplification

To link BER measurements (section 2.3) to system, in which the dominant random signal impairment is noise from in-line optical amplifiers (ASE), one typically uses the optical signal-to-noise ratio (OSNR). The OSNR is defined as

$$OSNR = \frac{P}{2B_{rf}N_{ASE}} \tag{4}$$

Where P is the average signal power (in both polarizations for polarization-multiplexed systems), B_{rf} is an optical reference bandwidth (typically chosen as 0.1 nm, or 12.5 GHz at 1550 nm), and N_{ASE} is the power spectral density of the ASE in each polarization. Another quantity commonly used in digital communication is the SNR per bit. It is defined as the ratio Eb/N_0, where Eb is the energy per bit and N_0 is the noise power spectral density. The relation between the SNR and OSNR is given by

$$OSNR = \frac{R_S}{2B_{rf}}SNR \tag{5}$$

$$SNR = \frac{S}{N} \tag{6}$$

Where R_S is the symbol rate.

3.2 Dispersion

In mono-mode fibers, a significant fraction of the energy in the bound mode travels in the cladding as an evanescent wave, and this introduce the first of our problems: chromatic dispersion.

- **Chromatic Dispersion**

  This phenomenon is about the delay (or deformation) of one optical pulse while is traveling along the fiber. Chromatic dispersion arises because of two reasons:

  Material Dispersion: It is the principal reason, it is because the refract index of the silicon depends of the frequency. Therefore, the components of different frequencies travel at different speeds along the silicon.

  Dispersion in waveguides: This happen because the effective rate of one mode is between the refraction index of the core and the cladding, and depending on how the power is distributed in each one; the effective rate will be near one or the other. Since the power distribution depends on the wavelength, if it changes, the power distribution changes, causing a change in the effective rate or mode propagation constant.

- **Polarization Mode Dispersion**

  To better understand PMD, we will enlarge one of the pulses and study it closely. We then recognize that a standard pulse's energy is the sum of two pulses. Both components of the pulse are in a polarization state.

  This causes a narrowing in the bandwidth, because we have to decrease the bit rate to avoid the inter-symbol modulation (ISI)

Figure 2: Polarization states of a single pulse.

3.3 Shot noise

The noise caused by random fluctuations in the motion of charge carriers in a conductor, again therefore a consequence of discretization (of the energy in the electromagnetic field in this case). Shot noise is a main part of **quantum noise**.

Shot noise is measurable not only in measurements at the few-photons level using photomultipliers, but also at stronger light intensities measured by photodiodes when using high temporal resolution oscilloscopes. As the photocurrent is proportional to the light intensity (number of photons), the fluctuations of the electromagnetic field are usually contained in the electric current measured.

In the case of a coherent light source such as a laser, the shot noise scales as the square-root of the average intensity:

$$\Delta I^2 \stackrel{def}{=} \langle (I - \langle I \rangle)^2 \rangle \propto I$$

A similar lower bound of quantum noise occurs in linear quantum amplifiers. The only exception being if a squeezed coherent state can be formed through correlated photon generation. The reduction of uncertainty of the number of photons per mode (and therefore the photocurrent) may take place just due to the saturation of gain; this is intermediate case between a laser with locked phase and amplitude-stabilized laser.

4 Expectation Maximization

EM is an iterative optimization method to estimate some unknown parameters Θ, given measurement data U. However, we are not given some "hidden" nuisance variables J, which need to be integrated out. In particular, we want to maximize the posterior probability of the parameters Θ given the data U, marginalizing over J:

$$\Theta^* = \overset{argmax}{\Theta} \sum_{J \in J} P(\Theta, J|U)$$

The intuition behind EM is an old one: alternate between estimating the unknowns Θ and the hidden variables J. This idea has been around for a long time. However, instead of finding the best $J \in J$ given an estimate Θ at each iteration, EM computes a distribution over the space J.

4.0.1 K-means Clustering

We begin by considering the problem of identifying groups, or clusters, of data points in a multidimensional space. Suppose we have a data set $x_1, ..., x_N$ consisting of N observations of a random D-dimensional Euclidean variable x. Our goal is to partition the data set into some number K of clusters, where we shall suppose for the moment that the value of K is given. Intuitively, we might think of a cluster as comprising a group of data points whose inter-point distances are small compared with the distances to points outside of the cluster. We can formalize this notion by first introducing a set of D-dimensional vectors μ_k , where $k = 1, ..., K$, in which μ_k is a prototype associated with the k^th cluster. We can think of the μ_k as representing the centres of the clusters. Our goal is then to find an assignment of data points to clusters, as well as a set of vectors μ_k, such that the sum of the squares of the distances of each data point to its closest vector μ_k , is a minimum.

It is convenient at this point to define some notation to describe the assignment of data points to clusters. For each data point x_n , we introduce a corresponding set of binary indicator variables $r_{nk} \in 0, 1$, where $k = 1, ..., K$ describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then $r_{nk} = 1$, and $r_{nj} = 0$ for $j \neq k$. This is known as the $1 - of - K$ coding scheme. We can then define an objective function, sometimes called a *distortionmeasure*, given by

$$J = \sum_{n=1}^{N} \sum_{k=1}^{N} r_{nk} ||x_n - \mu_k||^2 \tag{7}$$

which represents the sum of the squares of the distances of each data point to its assigned vector μ_k.Our goal is to find values for the r_{nk} and the μ_k so as to minimize J. We can do this through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the rnk and the μ_k . First we choose some initial values for the μ_k . Then in the first phase we minimize J with respect to the r_{nj} , keeping the μ_k fixed. In the second phase we minimize J with respect to the μ_k , keeping r_{nk} fixed. This two-stage optimization is then repeated until convergence. We shall see that these two stages of updating rnk and updating μ_k correspond respectively to the E (expectation) and M (maximization) steps of the EM algorithm, and to emphasize this we shall use the terms E step and M step in the

context of the K-means algorithm. Consider first the determination of the rnk . Because J in (7) is a linear function of r_{nk} , this optimization can be performed easily to give a closed form solution. The terms involving different n are independent and so we can optimize for each n separately by choosing r_{nk} to be 1 for whichever value of k gives the minimum value of $||x_n - \mu_k||^2$. In other words, we simply assign the n^th data point to the closest cluster center. More formally, this can be expressed as

$$r_{nk} = \begin{cases} 1 \text{ if } k = \arg\min_j ||x_n - \mu_k||^2 \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

Now consider the optimization of the μ_k with the rnk held fixed. The objective function J is a quadratic function of μ_k , and it can be minimized by setting its derivative with respect to μ_k to zero giving

$$2\sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \tag{9}$$

which we can easily solve for μ_k to give

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \tag{10}$$

The denominator in this expression is equal to the number of points assigned to cluster k, and so this result has a simple interpretation, namely set μ_k equal to the mean of all of the data points x_n assigned to cluster k. For this reason, the procedure is known as the *K-means* algorithm.

The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated in turn until there is no further change in the assignments (or until some maximum number of iterations is exceeded). Because each phase reduces the value of the objective function J, convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of J.

The K-means algorithm is illustrated using the Old Faithful data set in Figure 3. For the purposes of this example, we have made a linear re-scaling of the data, known as standardizing, such that each of the variables has zero mean and unit standard deviation. For this example, we have chosen $K = 2$, and so in this case, the assignment of each data point to the nearest cluster center is equivalent to a classification of the data points according to which side they lie of the perpendicular bisector of the two cluster centres. A plot of the cost function J given by 7 for the Old Faithful example is shown in Figure 4.

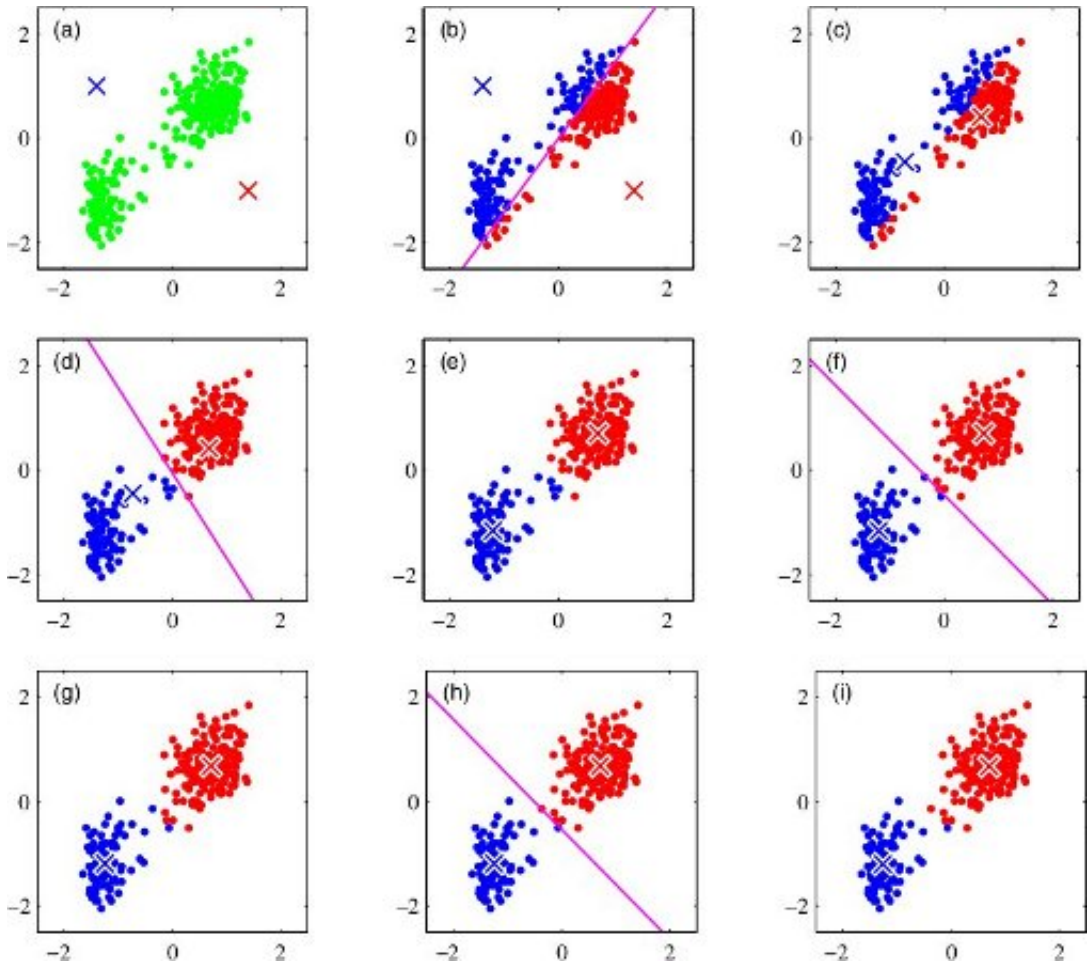

Figure 3: Illustration of the K-means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster center is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster center is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

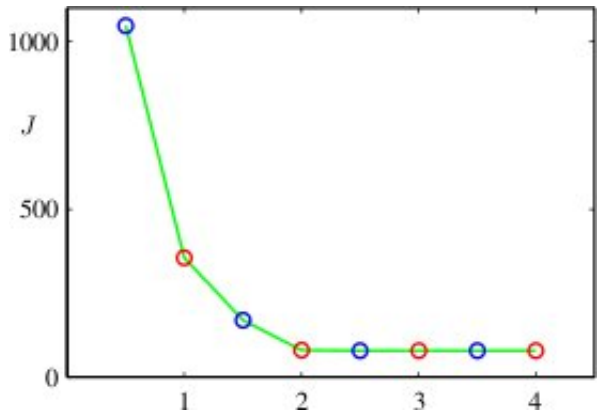

Figure 4: Plot of the cost function J given by 3 after each E step (blue points) 1000 and M step (red points) of the K-means algorithm for the example shown in Figure 3. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.

5 Hidden Markov models

Here we consider a particularly important class of such data sets, namely those that describe sequential data. These often arise through measurement of time series, for example the rainfall measurements on successive days at a particular location, or the daily values of a currency exchange rate, or the acoustic features at successive time frames used for speech recognition.

The easiest way to treat sequential data would be simply to ignore the sequential aspects and treat the observations as independent and identically distributed (i.i.d.). Such an approach, however, would fail to exploit the sequential patterns in the data, such as correlations between observations that are close in the sequence. Suppose, for instance, that we observe a binary variable denoting whether on a particular day it rained or not. Given a time series of recent observations of this variable, we wish to predict whether it will rain on the next day. If we treat the data as i.i.d., then the only information we can glean from the data is the relative frequency of rainy days. However, we know in practice that the weather often exhibits trends that may last for several days. Observing whether or not it rains today is therefore of significant help in predicting if it will rain tomorrow.

To express such effects in a probabilistic model, we need to relax the i.i.d. assumption, and one of the simplest ways to do this is to consider a Markov model. First of all we note that, without loss of generality, we can use the product rule to express the joint distribution for a sequence of observations in the form:

$$p(x_1, ..., x_N) = \prod_{n=1}^{\infty} p(x_n | x_1, ..., _{n-1}) \tag{11}$$

If we now assume that each of the conditional distributions on the right-hand side is independent of all previous observations except the most recent, we obtain the first-order Markov chain, which is depicted as a graphical model in Figure 5 joint distribution for

a sequence of N observations under this model is given by:

$$p(x_1, ..., x_N) = p(x_1) \prod_{n=2}^{\infty} p(x_n|x_{n-1}) \tag{12}$$

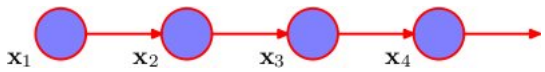

Figure 5: A first-order Markov chain of observations x_n.

The hidden Markov model can be viewed as a specific instance of the state space model of Figure 6 in which the latent variables are discrete. However, if we examine a single time slice of the model, we see that it corresponds to a mixture distribution, with component densities given by $p(x|z)$. It can therefore also be interpreted as an extension of a mixture model in which the choice of mixture component for each observation is not selected independently but depends on the choice of component for the previous observation.

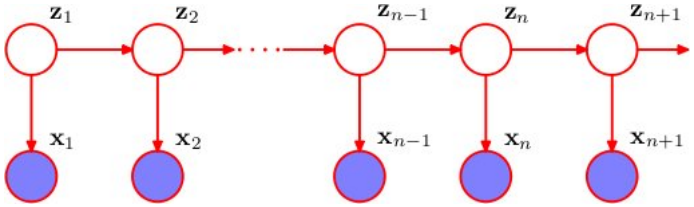

Figure 6: This important graphical structure forms the foundation both for the hidden Markov

We now allow the probability distribution of z_n to depend on the state of the previous latent variable $z_{n?1}$ through a conditional distribution $p(z_n|z_n?1)$. Because the latent variables are K-dimensional binary variables, this conditional distribution corresponds to a table of numbers that we denote by $\mathbf{A}$, the elements of which are known as transition probabilities. They are given by $A_{jk} \equiv p(z_{nk} = 1|z_{n?1,j} = 1)$, and because they are probabilities, they satisfy $0 \leq A_{jk} \leq 1$ with $\sum_k A_{jk} = 1$, so that the matrix $\mathbf{A}$ has $K(K-1)$ independent parameters. We can then write the conditional distribution explicitly in the form

$$p(z_n|z_{n-1}, \mathbf{A}) = \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j} z_{nk}} \tag{13}$$

The initial latent node z1 is special in that it does not have a parent node, and so it has a marginal distribution $p(z_1)$ represented by a vector of probabilities π with elements $\pi_k \equiv p(z_{1k} = 1)$, so that

$$p(z_1, \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}} \tag{14}$$

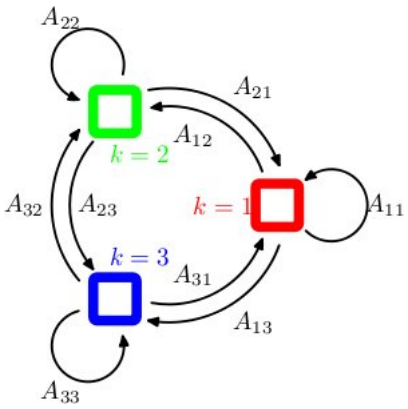

Figure 7: Transition diagram showing a model whose la- tent variables have three possible states corresponding to the three boxes. The black lines denote the elements of the transition matrix A_{jk}

The transition matrix is sometimes illustrated diagrammatically by drawing the states as nodes in a state transition diagram as shown in Figure 7 for the case of K = 3. Note that this does not represent a probabilistic graphical model, because the nodes are not separate variables but rather states of a single variable, and so we have shown the states as boxes rather than circles.

The specification of the probabilistic model is completed by defining the conditional distributions of the observed variables $p(x_n|z_n, \phi)$, where ϕ is a set of parameters governing the distribution. These are known as *emission probabilities*, and might for example be given by Gaussian if the elements of x are continuous variables, or by conditional probability tables if x is discrete. Because x_n is observed, the distribution $p(x_n|z_n, \phi)$ consists, for a given value of ϕ, of a vector of K numbers corresponding to the K possible states of the binary vector z_n . We can represent the emission probabilities in the form

$$p(x_n|z_n, \phi) = \prod_{k=2}^{K} p(x_n|\phi_k)^{z_{nk}} \tag{15}$$

6 Describing the problem

Before describing the problem, we have to define the environment. In this experiment, we are going to transmit a M-PSK signal through a fiber channel. Figure 8 shows the block diagram:

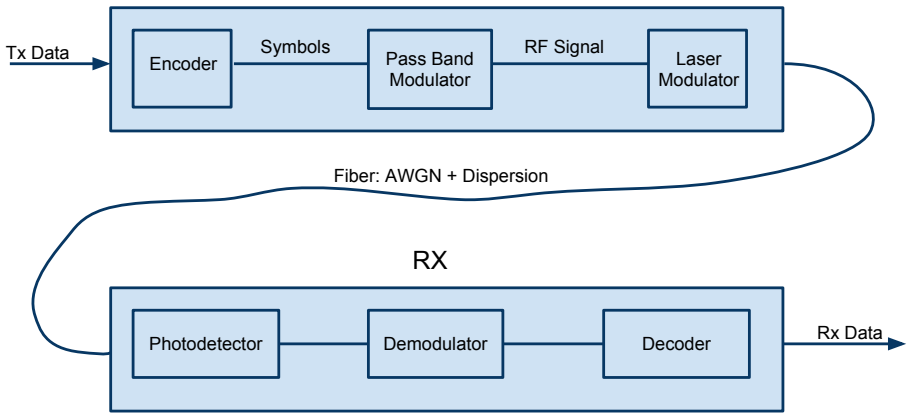

Figure 8: Block Diagram

In the **transmission module**, we define three blocks: encoder, pass band modulator and laser modulator.

- **Encoder** This is the first module in our transmission scheme. It receives the digital data in bits and encode them in symbols. To do that, it takes bits in packets of K by K bits, where $K = \log_2(M)$, Assign to each symbol a combination of bits, and consequently generates the proper symbol. The most important parameter related to this module is the Bit Rate(R_B).

- **Modulator** This module (figure 9) is in charge of preparing the symbols to be transmitted in the channel depending on the features of the channel. In this case, it is the pass band channel which will transform the symbols in sinusoidal signals. Since the PSK modulation is used, the output will be a signal that preserves the amplitude and the frequency. Important factors are to be considered and they are the bit rate and the carrier frequency. Since the QPSK modulation is being used, 2 bits per symbol is assumed, and as it is known, in order to preserve the consistency of the system, the bit rate must be approximately equal to the carrier frequency. As a result an important relationship is obtained:

$$F_C = \frac{R_b}{\log_2 M}$$

  Where Fc is the carrier frequency, and M is the number of symbols we have.

- **Laser** This is the last step in the transmitter, it prepares the signal to be transmitted by a fiber channel. It transforms the signal from the electrical domain in to the light domain. As with the digital modulations, there are different variables to modulate the laser: Amplitude, Phase, or Frequency. The last two will lead to a discussion about coherent detection, but since this is not the topic of the report,

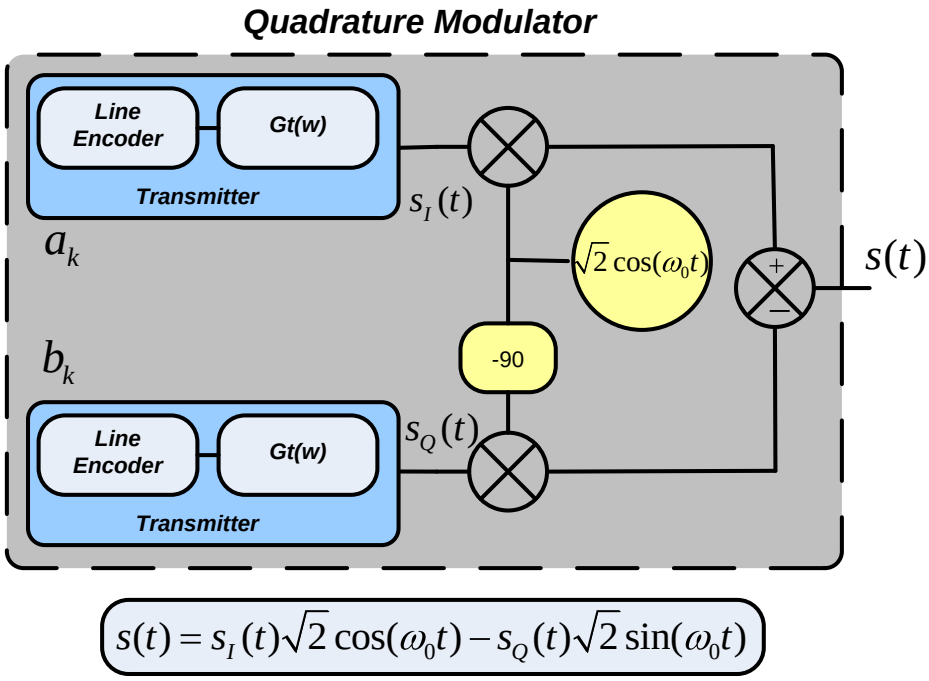

Figure 9: Bandpass Signal Modulator

only the amplitude modulation will be used in order to simplify the system. The laser can be modeled as follows:

$$E_S = \sqrt{P_S} \cdot e^{j(\omega_s t + \varphi_S)} \tag{16}$$

Now, the signal will be transmitted by the **fiber channel**. This means that the chromatic dispersion and the Gaussian noise will appear. The Gaussian noise is governed by the SNR parameter. It is a linear distortion of the signal that can be seen as a sum of random values to the signal. These random values are produced in a Gaussian distribution with $\mu = 0$ and variance $\sigma^2 = 0.5 \cdot 10^{\frac{E_B/N_0}{10}}$.

In the case of the chromatic dispersion, it is a little bit more complicate. This phenomenon is about the delay (or deformation) of one optical pulse while it is traveling along the fiber. The result of this phenomenon to the signal, is a pulse broadening.

The receiver module has the most difficult task, which is to reconstruct the signal and return the digital data without many errors.

- **Photo-detector** It receives the light carried in the fiber and returns the envelope signal in the electrical domain. Since a non-coherent detection is used to obtain the envelope, the photo-detector model is given by this operation:

  $$y = \sqrt{E_S \cdot E_S^*}$$

  Where y is the modulated signal, and E_S is the laser carrier modeled in the equation 16.

- **Demodulator** Here the symbol recovery is performed by extracting the SI and SQ channels. Since the transmitted signal becomes noisy, the symbols will not be in their expected order, creating a "bench" of points around each symbol's place.
- **Decoder** This is the most important part for this project. This module classifies each symbol and decides which *true* (transmitted) symbol belongs to which received symbol. This is the basis for the discussion (solution), since this module does't deal with the nature of the data that is being handled.

Figure 10 shows an example of how the signals look like when they pass through the system with 8 dB of **SNR**. In this figure, the top plot shows the output of the modulator, next to the output of the laser modulator, and at the bottom the effect of the noise. Figure 11 shows how the signal should look like in the decider module.

For the **dispersion** case, the effect of what has been explained before can be seen. The red line in the last plot of figure 12 reveals the pulse broadening effect. In the figure 13 two effects for the decider module can be seen: horizontal widening of the constellation, and an important change in the distribution of the symbols. This causes several problems for the decider module in zones near the horizontal axis due to the new distribution of the data.

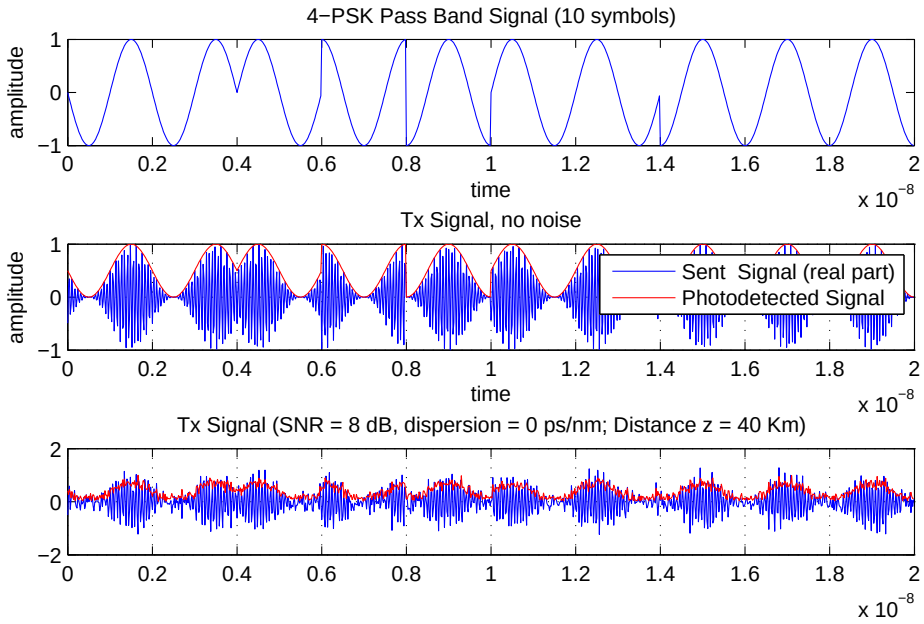

Figure 10: SNR effect in the received signal

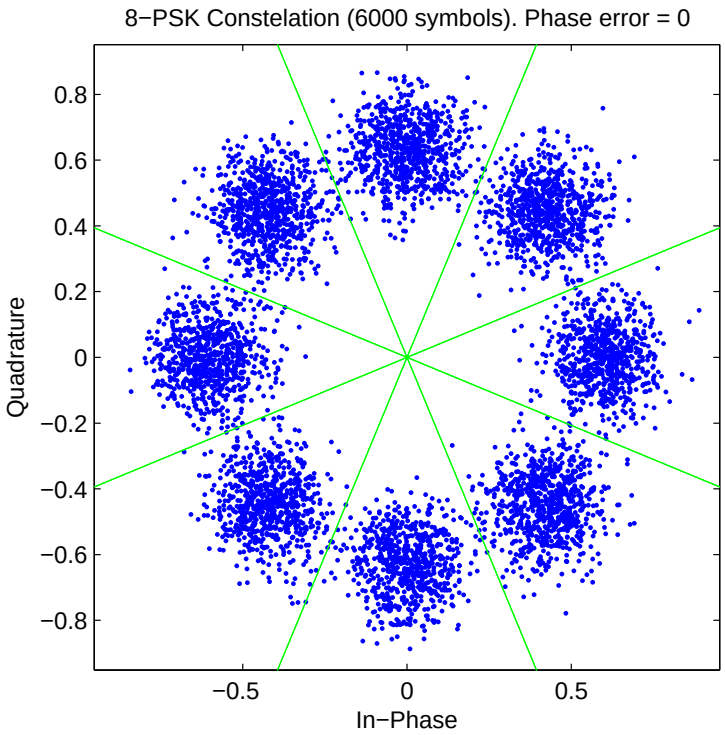

Figure 11: SNR effect in the decider module

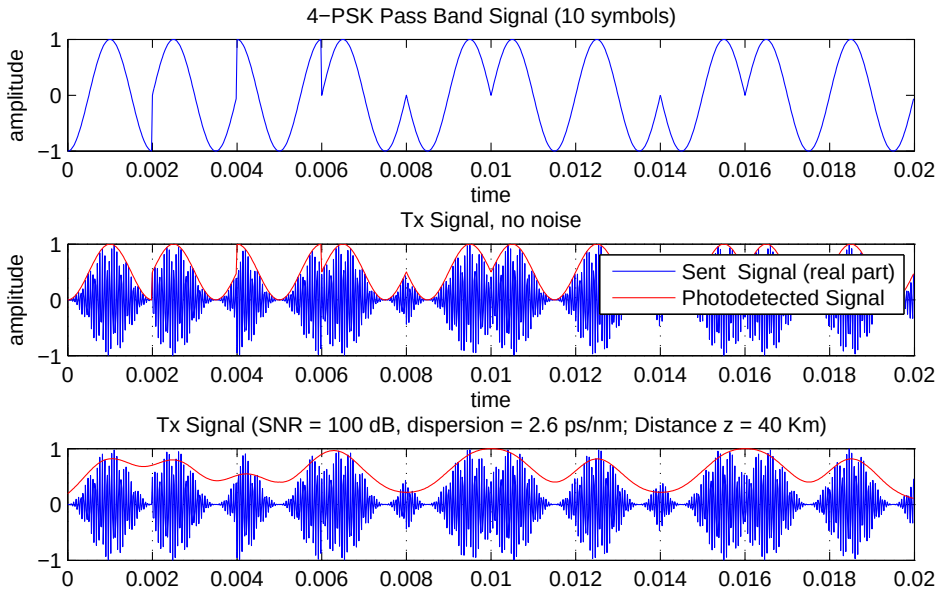

Figure 12: Dispersion effect in the received signal

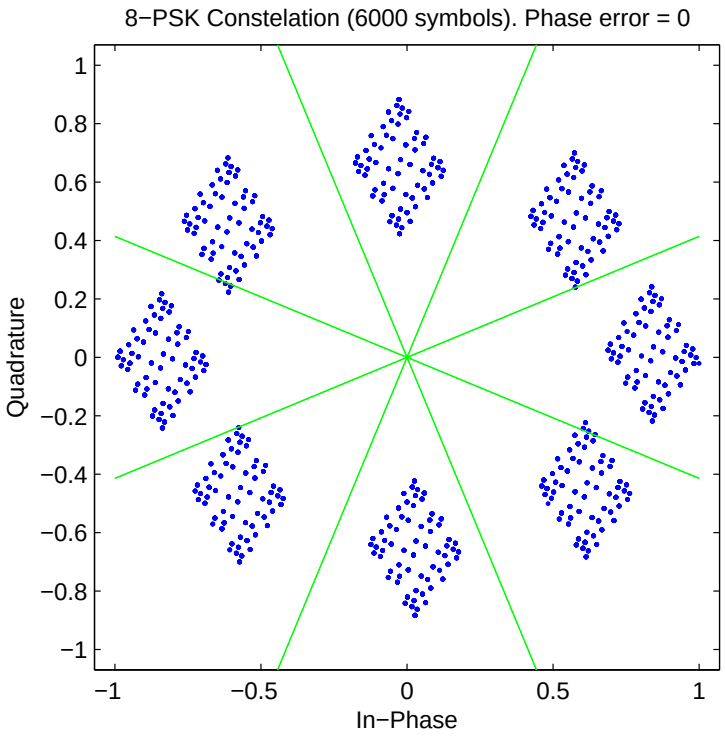

Figure 13: Dispersion effect in the decider module

Another problem, when performing the carrier recovery, comes from the phase offset error. It is caused by the demodulator when it begins working, since it does't know when exactly to start; thus the effect of rotation of the whole constellation. That means that the decider will misinterpret the position of the symbols, and it will start classifying based on a wrong model. Figure 14 shows a QPSK constellation rotated 0.65 radians, it is clearly visible that half of the symbols are incorrect due to wrong decision boundaries. For the rest of the study we assume that frequency estimation and synchronization are perfect.

When we mix the phase error phenomenon with the dispersion, the risk in the decider module increases because they interact each other. Figure 15 shows an 8PSK constellation with $\pi/2$ of phase offset and 2.6 ps/nm of dispersion. The orientation of the constellation and the distribution of the data have completely changed, causing worst effects in zones near the vertical axis.

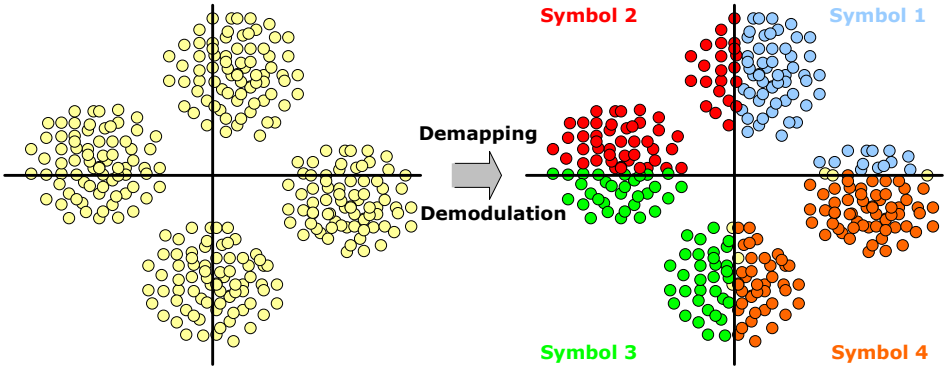

Figure 14: Phase error effect with AWGN noise

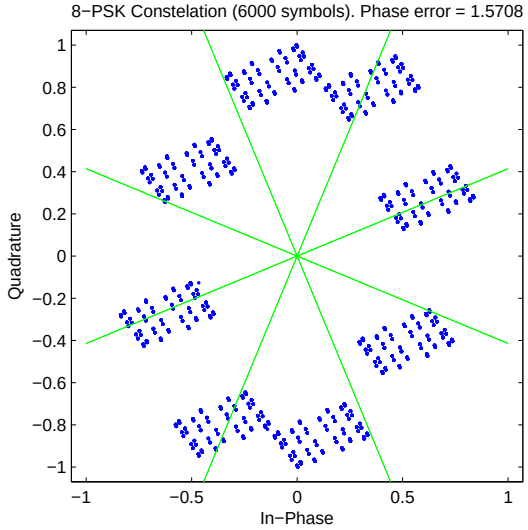

Figure 15: Phase error effect with dispersion

The problem arises when the dispersion and the noise are relatively high in comparison with the number of symbols of the constellation (M). In that case, an effect known as ISI (Inter-Symbol Interference) appears leaving the decider with a zone of erroneous symbols, as shown in figure 16.

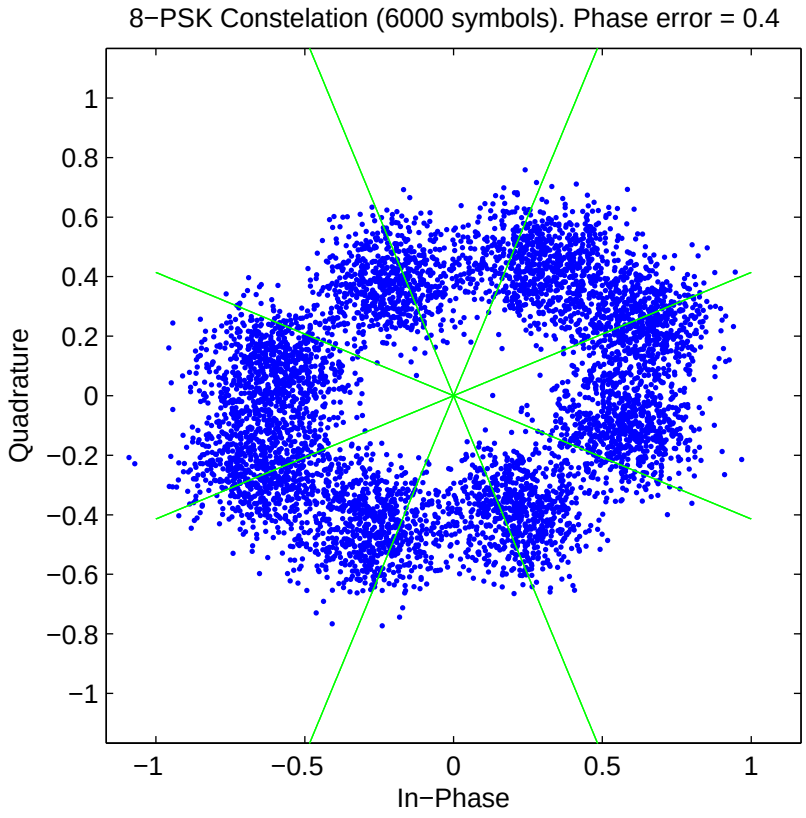

Figure 16: 8PSK affected by dispersion, AWGN, and phase error.

7 Proposed solution

Our solution will be performed in the decider module and it evolves also the encoder module. So we will create a special protocol of communications based on Markov theory. It consists of generating a time relationship between the sequence of symbols transmitted so that allow us to use hidden Markov theory. In the next sections we describe the process of how this relationship is achieved.

7.1 New transmission scheme

Let us take a 8-PSK signal as an example. When we send the symbols one by one, there is no relationship between them. But if we modify the behavior of how the symbols are created, that is, the behavior of the encoder module, we can easily generate the symbols in a different way in order to get what we want. Let us see our 8-PSK signal as a combination of two QPSK Signals as it is shown in figure 17

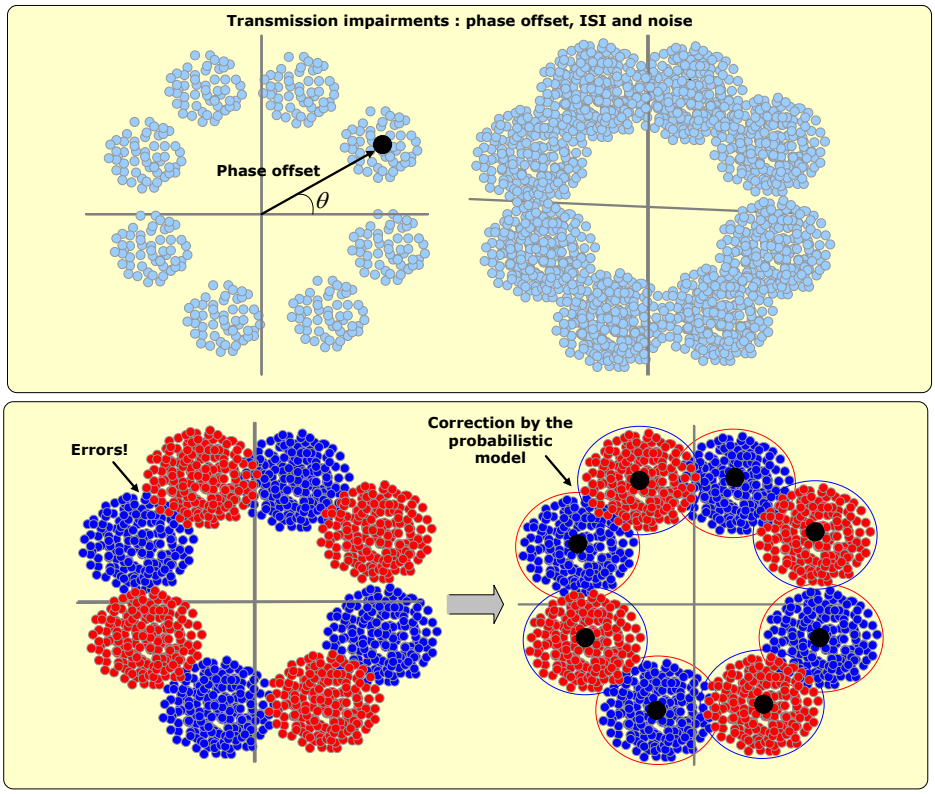

Figure 17: Spiting process

Here we have changed the scheme: instead of having one 8-PSK we have two sub-QPSK signals. In the time domain, this can be seen as a multiplexation in time of two QPSK signals, in which one of them (the red one) has been rotated $2\pi/M$ rads. So, when we are generating the symbols in the encoder module, it is certain that two consecutive symbols will never belong to the same sub-QPSK, and that is a relationship in the time domain. Figure 18 explain how this is performed in the encoder module.

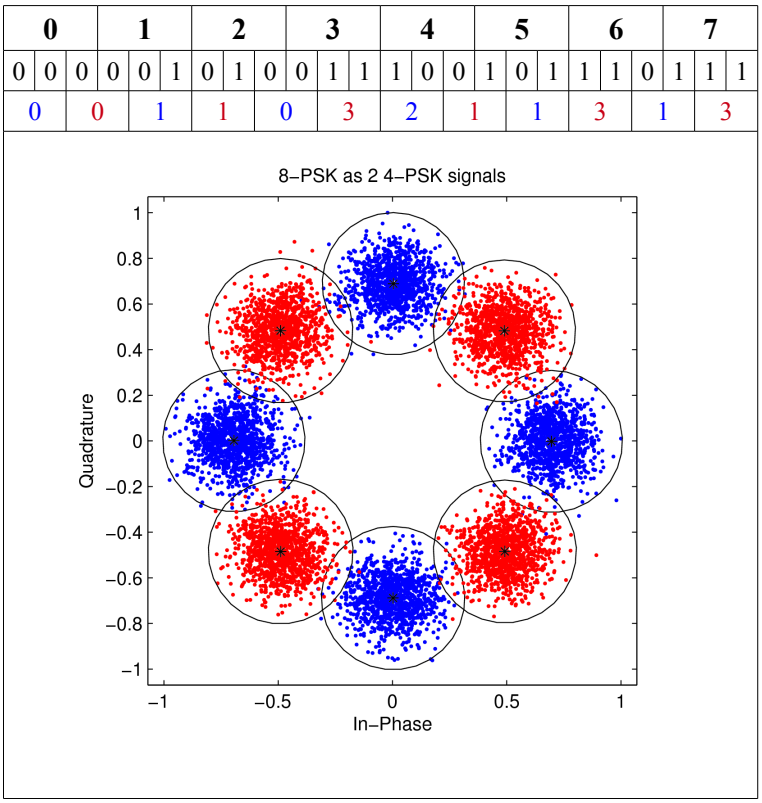

Figure 18: Multiplexing process

The multiplexation process is very simple. First, we take the symbols from the 8-PSK signal, and create a bit map therefrom. This bit map is very simple since the information of which QPSK is the symbol belonging to is attached to the time; so what have to do is only to group the bits in packets of two and construct the symbols as if it was a normal QPSK signal.

Afterwards, we start coding the symbols. To do that, we only need to code every two symbol with a $2\pi/M$ ads of phase offset. That will create the form of 8-PSK symbols, but the symbols are placed in the region following a Markovian rule (following some time domain dependency). Once we have created the symbols, the process continues as if it was a normal 8-PSK signal, since for the rest of the components, the signal looks like a normal 8PSK signal.

Now, let us generalize the problem. We have an M-PSK signal that we want to split in Q sub N-PSK signals, where M is the number of symbols of the global system, Q is the number of sub-psk signals, and N is the number of symbols of each sub-psk signal (M_{psk}). It is easy to see that $Q \cdot N = M$. The time dependency is then generalized to Q time slots, meaning that every Q consecutive symbols will never belong to the same sub-psk signal. In this case, the phase offset of each sub N-PSK is given by $Q_i \cdot 2\pi/M$. Figure 19 shows an example of that.

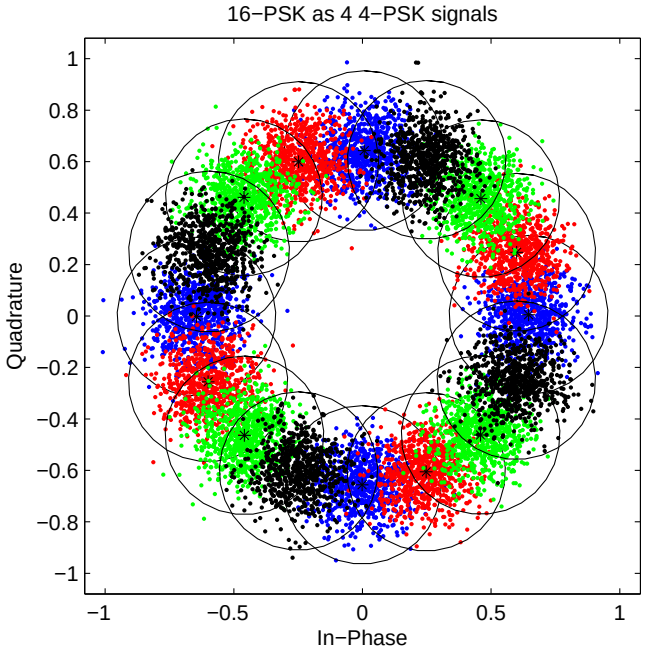

Figure 19: Generalized multiplexing process

7.2 Applying AI algorithms

Now, we are placed in the decoder module. This was the module in charge of deciding what symbol is the current sample belonging to. The normal way to do that is to set some decision boundaries and define some regions where the symbols should be. We will make it more complex and split the decision process into two steps: applying hidden Markov theory, and building the decision boundaries dynamically.

7.2.1 Hidden Markov Model

To perform the demultiplexation process, we first encounter one problem: which QPSK signal does the current symbol belong to? To solve this, we use the Hidden Markov Model. In order to use it, our model has to meet one important requirement: the current symbol must depend on the previous symbols. We know that there is no relationship between the symbols generated by a random process, but we can use the multiplexation feature to make it.

In an environment with 2 multiplexed QPSK, we know that there is a phase offset between them, and we can use that for the hidden Markov model. Everytime we receive a symbol, it can be classified in one region, but it is not enough to determine which QPSK it belongs, so we check where the previous n symbols fall, and make an estimation from their phase offset.

We define a Hidden Markov Model as $M = \{S, \pi(1), A, B, Y_K\}$ where:

- $S = \{1, 2\}$: States of the model, the number of qpsk in our system, 2.
- $\pi(1)$: Initial probability vector.
- A: The state transition probability matrix, the probability of jumping from one qpsk to another one.
- B: The emission probability matrix, the probability of jumping from one symbol to another.
- Y_K: The observed vector, the received signal.

To calculate the A Matrix, we know that the multiplexation process is constant, if we have 2 qpsk multiplexed constituting a 8-PSK system, we always send them linearly. Hence, A is a S by S secondary diagonal matrix as follows:

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The B Matrix tells us the probability of one observed symbol to jump into another one. As we consider 8 possible different symbols, they hide 2 QPSK constellations. Hence, this matrix will be a M by S matrix as follows:

$$B^T = \begin{pmatrix} b & 0 & b & 0 & b & 0 & b & 0 \\ 0 & b & 0 & b & 0 & b & 0 & b \end{pmatrix}, b = 1/N$$

$\pi(1)$ ells us which QPSK is probably the first one. Since this information is not available, we assume a uniform distribution.

$$\pi(1) = \begin{pmatrix} p & p \end{pmatrix}, p = 1/Q$$

Let us explain the same process more intuitively. In the encoder module, we created the symbols following some rule, which was multiplexing Q N-PSK signals to generate one M-PSK signal. Now we are going to separate these signals. In the 8-PSK signal spitted by two QPSK, the decoder only has to separate the even symbols and the odd symbols. Since we have created these symbols as two QPSK, the result of the process are two QPSK. We have maximized the decision region to the double, as it is shown on figure 20.

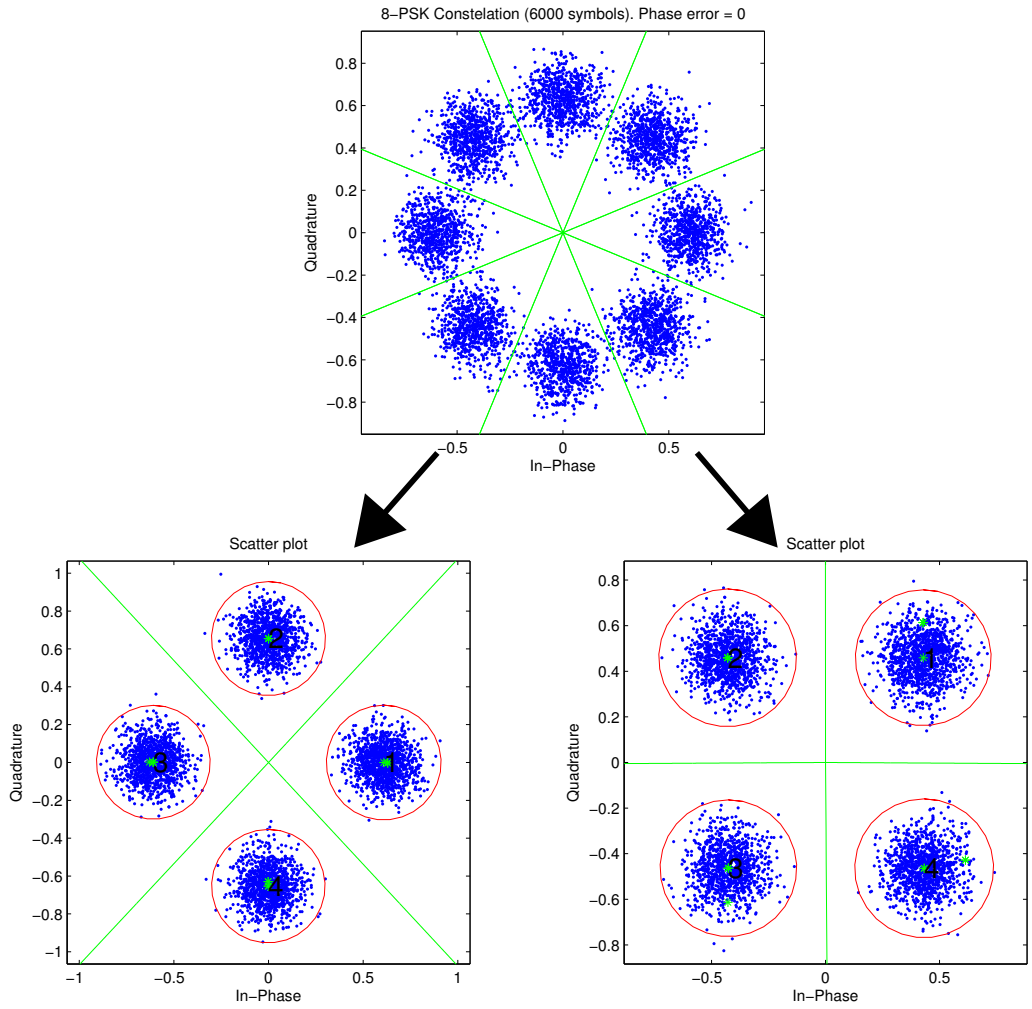

Figure 20: Hidden Markov model process

7.2.2 Expectation Maximization

With the previous algorithm we have solved problems originating from the dispersion and the AWGN. Now we have to solve another type of problem like the phase offset. As we described in the section 6, the current decoders make their decisions based on decision boundaries built in. But, what if the decoder makes its decision using a probabilistic model based on where the symbols are, instead of where the symbols are supposed to be? This means that the decider module can adjust the decision boundaries in order to how the channel influences in where fall the symbols. In other words, we can make the decider module learn how to model the channel.

To achieve that, we are going to use the Expectation - Maximization algorithm, explained in the section 4. The concept is simple, we have a bench of points (symbols) spread all over the space, following some probabilistic density functions. Since we know that the symbols are grouped according to where they should be in a Gaussian form, we have only to learn the means and the variances of these clusters of points. An important point is that we don't know where these points are grouped. The algorithm has to learn

these means, so if the channel is rotating the constellation, the probabilistic model will adjust the means; even if the channel is causing any asymmetry in the constellation as well. The means can be far away from the original symbols. The same happens with the variances. We only assume that the co-variance matrix of each Gaussian is spherical (equation 17), but σ is learned by the algorithm.

$$\Sigma = \begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix} \tag{17}$$

We now have defined the means and variances of each Gaussian symbol output from the channel. With these data, we are able to construct a Probabilistic model based on the posterior probabilities of the data $P(N/x)$, that is, the probability of given x_i,it belongs to N_j. We can use the maximum likelihood method to perform that. Therefore the demodulation process is a question of what N_j has the higher probability for each x_i.

You should not forget that all this process is applied to each sub-psk signal, not in the global one. When each QPSK is demodulated separately, the remaining task is to multiplex the symbols again in time domain, and reconstruct the original 8-PSK signal by demapping each symbol and reassigning the number of bits.

This algorithm has only one disadvantage: the computing time. Since it is an iterative algorithm, it will not finish until all the symbols are correctly classified, and for real time applications it is a big inconvenience. But we can reduce this time by performing a correct initialization of the means. To initialize the algorithm, we will use the features of each sub-qpsk. Since the system knows the N parameter (number of symbols per sub-psk), we know how many clusters we have and also how they are they supposed to be spread. Performing a correct initialization of the points, if the Gaussian are far enough from each other (which is performed by the demultiplexing process), the number of iterations goes to 2 or 3, which it is acceptable, and it perfectly fixes anomalies such a phase offset. Figure 21 shows an example of that.

Another problem that the EM algorithm can fix was about the phase offset and dispersion effects, because the constellation were turned elliptically and the distribution of the points changed radically. Comparing figures 15 and 22 you can see the dynamism of the decision boundaries, showing how some of them are bigger than others.

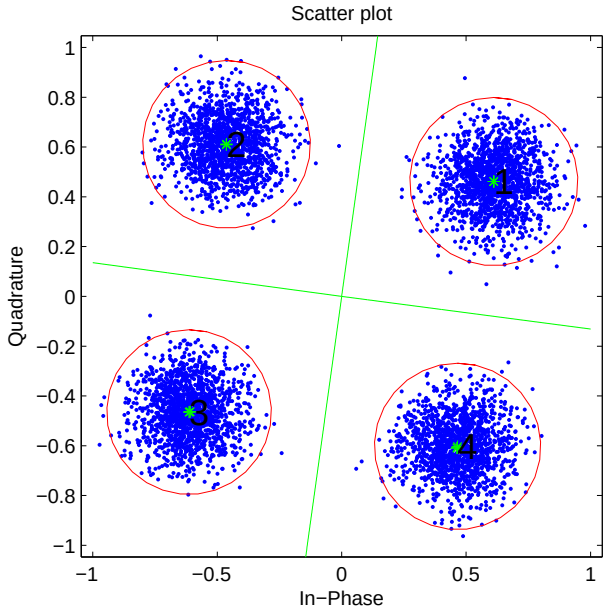

Figure 21: Correction of phase offset by EM algorithm

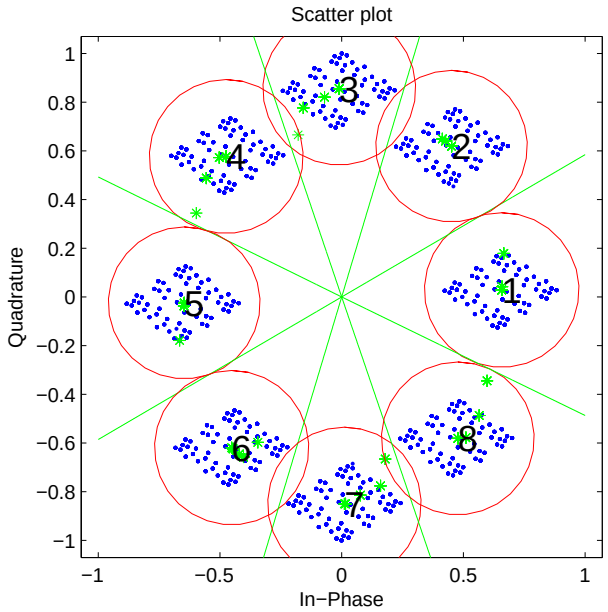

Figure 22: Dispersion and phase offset learned

Taking a look of the whole process, figure 23 illustrates how the two algorithms work together and correct the phase offset in a 8PSK spitted by 2 QPSK.

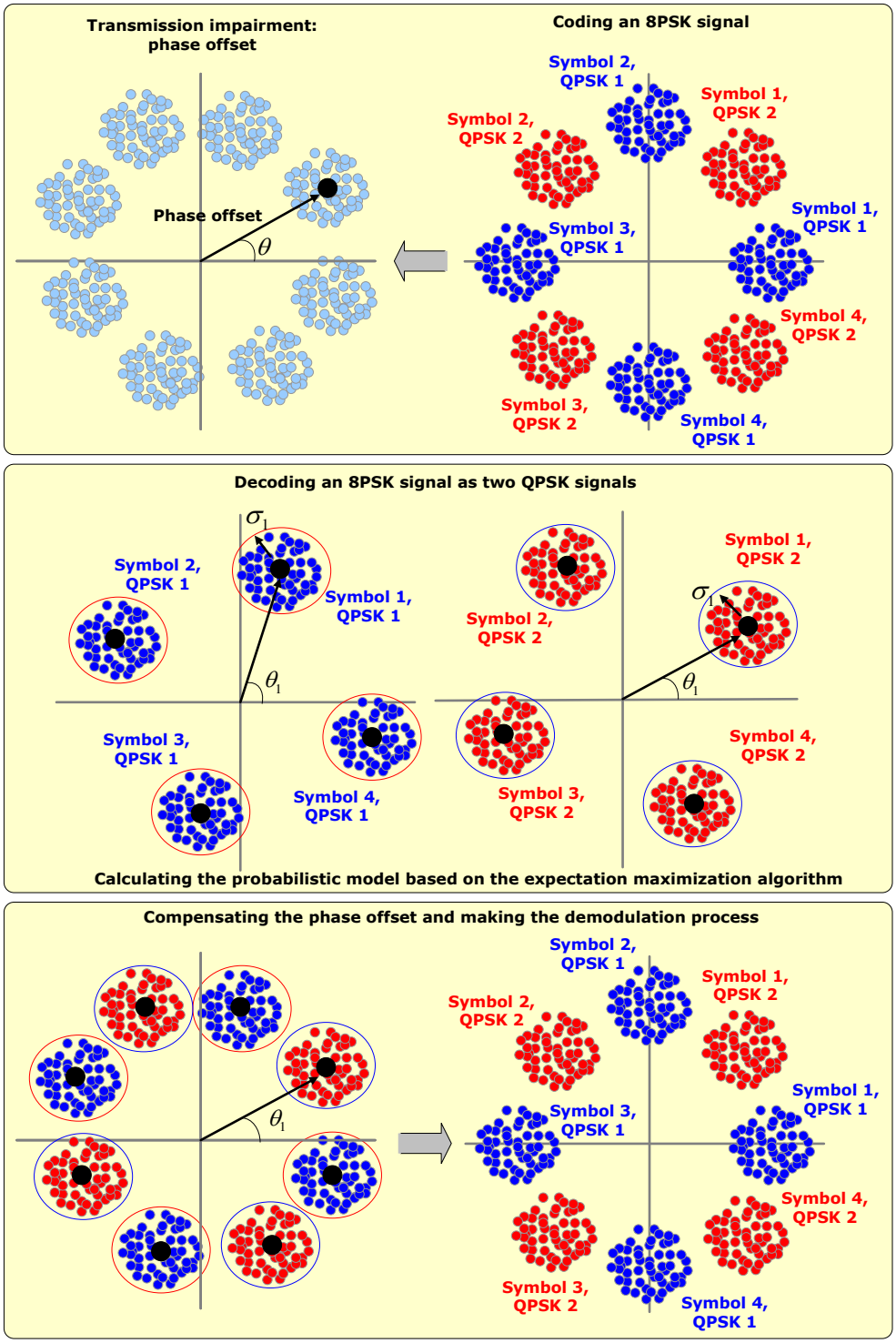

Figure 23: Classification process

7.3 Advantages and Disadvantages

7.3.1 Phase offset correction

As we have commented previously, the EM algorithm can easily correct small phase offsets. If the phase offset is too hard, then we have the symbol ambiguity. This means that the receptor can't identify correctly the symbols unless there is a training period. If ones everything is established the receptor keep tracking by the EM algorithm and updating the boundaries, then the problem is solved, and we have one more advantage: Self phase-tracking since centroids are updated continuously.

7.3.2 Modulation format flexibility

The principle is applicable to any modulation format (PSK and QAM) if frequency downconversion and timing recovery are perfect. The only change we should take into account is the initialization of the centroids, but we can use the same principle if we know the modulation format used.

7.3.3 Multiplexation flexibility

The system can be easily set up for the number of subpsk's multiplexed. It means that either you can set two QPSK composing one 8-PSK or you can compose one 128-PSK by multiplexing 16 8-PSK. It is clear that as much sub-Psk you have, the less is the BER, the bit rate, and the higher is the complexity of the system.

7.3.4 Real-time implementation

This method works fine and achieve everything we have talked about in a simulation environment. In real environments that method can increase the computing time resulting very harmful in real time transmissions such audio or video.

8 Implementation

The simulation framework is a set of scripts made in MATLAB^{TM}of which propose is to show the performance of the normal M-PSK system in comparison with the system described in this project. The process is described in the previous section, and the code is available in the CD-ROM [5] attached to this document. Instructions of how the program should be used are in the `Readme.txt` file.

The program is divided in three main scripts:

- `main1.m` is in charge of the simulations for the normal process of digital modulations.
- `main2.m` simulates the system described in the previous section.
- `ber.m` uses the two previous scripts and shows the simulated BER curves of each system.

The variables of the system are located in `variables.m` file. We will explain the implementation of the code as an example of one 8-PSK generated as two QPSK multiplexed signals.The variables can be classified in two categories:

- **General variables:** the number of symbols (M), the data length (L), the bit rate (Rb), the sampling frequency (Fs), and the SNR.
- **Optical variables:** the wavelength (lambda), the power of the output wave (power), the initial phase (phase_ini), the dispersion (D) and the distance (z).

Also concrete variables of the multiplexation system can be modified, like num_psk that is the number of multiplexed sub psk. The output of the plots and other simulation boolean values can be found in the simulation variables section.

Since `main2.m` is an extension of `main1.m`, we will talk about the first one, explaining how is all the process mentioned achieved.

8.1 Multiplexation

In this process we prepare the data to create the time dependency used in the HMM algorithm. The process consists in taking the symbols (M=8) and mix them as 2 x (M=4) so we can construct a normal QPSK. The trick is that we are going to multiplex them in time domain, so two consecutive symbols will never belong to the same QPSK.

8.2 Modulation

After generating a pseudo-random bit sequence as a data to transmit, we proceed to encode each K_2 bits in symbols, where $K_2 = log_2 N$, it means the number of bits per subpsk,in this case, 2. Since we are encoding them as a two QPSK, a phase offset = pskindex$\cdot 2\pi/M$ is applied for each symbol in order to preserve the multiplexation process. In that case it is just apply $2\pi/M$ of phase offset each 2 symbols. The graphical view of the process can be seen in the figure 18. After, a pass band modulation with a

carrier of F_c Hz is applied. In this process, a signal like in the figure 24 is generated. The content of this signal is the result of subtracting the SI and SQ channels:

$$SI = Im\{x_{mod}\} \cdot \cos(2\pi \cdot F_C \cdot t) \quad (18)$$

$$SQ = Re\{x_{mod}\} \cdot \sin(2\pi \cdot F_C \cdot t) \quad (19)$$

Where x_{mod} contains the symbols of data.

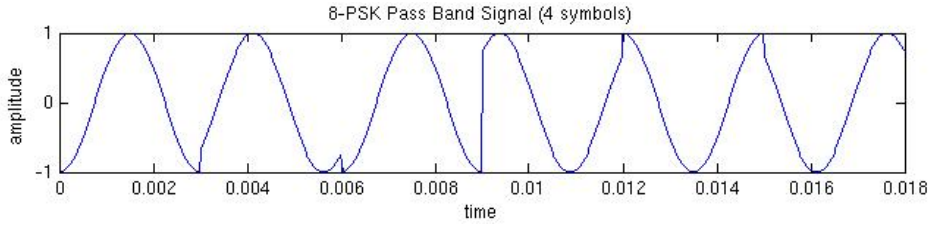

Figure 24: Complete QPSK Transmission

8.3 Fiber channel model

That part of the simulation is model in two steps: Gaussian noise and Dispersion. Gaussian noise (AWGN) is governed by the SNR parameter. It is a linear distortion of the signal that can be seen as a sum of random values to the signal. This random values are produced by a Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 0.5 \cdot 10^{\frac{E_b/N_0}{10}}$.

As a result of the chromatic dispersion a pulse broadening in the envelope of the signal will be model. To achieve that, we are going to use a Gaussian FIR filter with many taps as samples per period has the envelope. To calculate the mean and the variance of our filter, we will begin from the model proposed by Seb J. Savory's model [4]:

$$g(z,t) = \sqrt{\frac{c}{jD\lambda^2 z}} exp\left(j\frac{\pi c}{D\lambda^2 z}t^2\right) \quad (20)$$

The expression of a Gaussian distribution is defined as

$$N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \leftarrow \sigma^2 = j\frac{D\cdot z\cdot\lambda^2}{2\pi c} \quad (21)$$

$$= \frac{1}{\sqrt{2\pi j\frac{Dz\lambda^2}{2\pi c}}} exp\left(-\frac{(x-\mu)^2}{2j\frac{Dz\lambda^2}{2\pi c}}\right) \leftarrow x = t, \mu = 0 \quad (22)$$

$$= \sqrt{\frac{c}{jD\lambda^2 z}} exp\left(j\frac{\pi c}{D\lambda^2 z}t^2\right) \quad (23)$$

Since we are applying the model directly to the envelope in our simulation, we use the real part of the σ.

$$N(\mu,\sigma^2)$$

$$\mu = 0$$

$$\sigma^2 = \frac{D\cdot z\cdot\lambda^2}{2\pi c}$$

Where D is the dispersion parameter, z is the distance, and λ is the wavelength.

8.4 Receiver

The receiving process is implemented in the function `mpskdemod.m`. It takes the signal, already photo-detected, it means, the envelope; and demodulates signal and decodes it by using the Markov and EM theory.

8.4.1 Demodulator

Assuming that frequency estimation and synchronization are perfect, we leave the carrier recovery problem in terms of phase offset estimator for the decider module. In this first step, we apply the usual steps to demodulate the signal: Filtering, frequency down-conversion, timing recovery, signal demodulation... The whole process can be illustrated in the figure 25

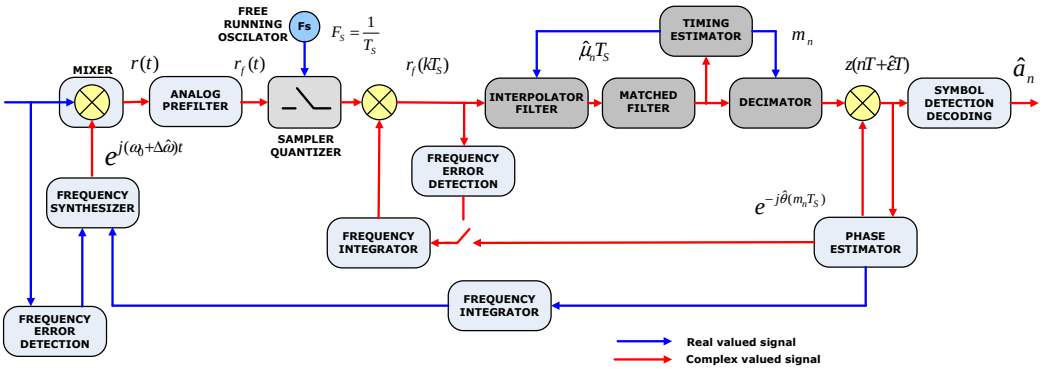

Figure 25: PAM receiver

8.4.2 Decider

The demultiplexing process is inspired in the **Hidden Markov Model** theory. As we have seen, there is a lot of equations and probability matrix linked to this process, but in a simple view, all we have to do is to create a SubPsk matrix with all the symbols ordered by number of sub psk as a rows, and number of symbols (data) as a columns. The method used to create this matrix uses the multiplexing feature that we forced in the encoder module. Thus, what we really do is to separate the sequence of symbols in Q different rows corresponding to Q different time slots:

$$B = (b_1, b_2, b_3, b_4 \cdots b_n)$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{pmatrix} \tag{24}$$

Where B is the original sequence data, $a_{ij} = s_{ij} + n$ where i determines the SubPsk and j determines the current symbol. a is therefore a point in the complex plane; and n represents the AWGN model as a $N(\mu, \sigma^2)$.

8.4.3 EM algorithm

Ones separated the sequence of symbols in Q different SubPsk (Matrix 24), we are able to apply **EM**. Our aim is to compute the means and the variances of each cluster of symbols for each sub psk (for each j row). To make it, we will use the *K-means* algorithm in order to identify the clusters, and then we will calculate the variances of each cluster, amusing that the noise is given by the AWGN , we can use the spherical covariance matrix (equation 17). To better understand how this process work, we now link the theory studied in section 4.0.1 by identifying the parameters of the equation 7

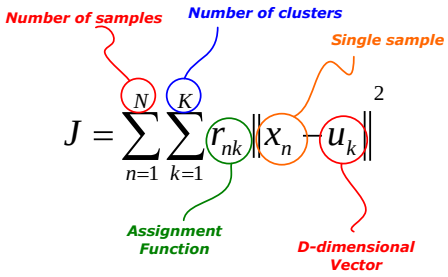

Figure 26: The principle of *k-means*

All this process is performed in the function `kmeans.m`, that receives a set of bi-dimensional (from the complex data) data, the number of K clusters, and the phase of the first symbol. Figure 27 illustrate how this algorithm works for the case of one sub QPSK.

At that point we have a problem: we have identified the clusters but we don't know which symbol is each cluster, since our algorithm only separate them. To order the clusters according to the symbols, we will pass a new parameter to power `kmeans.m` function. The phase of the first symbol. We use it to sort the means from this phase. This means that the first symbol (1) will be the closest cluster next to this phase parameter. The calculation of this phase is done while the demultiplexing process is running, and given by $\phi = Q \cdot 2\pi/M$. where Q represents the sub psk we want to calculate the means and variances, and M refers the complete number of symbols (not only for each sub psk).

The result of this process are two vectors: one contains the means of each cluster ordered by the symbol identification (since we are working in a bi-dimensional plane, it is actually a matrix of $M_{psk} \times 2$) and the other contains the variances of each cluster also ordered. This is all the information we need to build our own probabilistic model based in Gaussian mixtures.

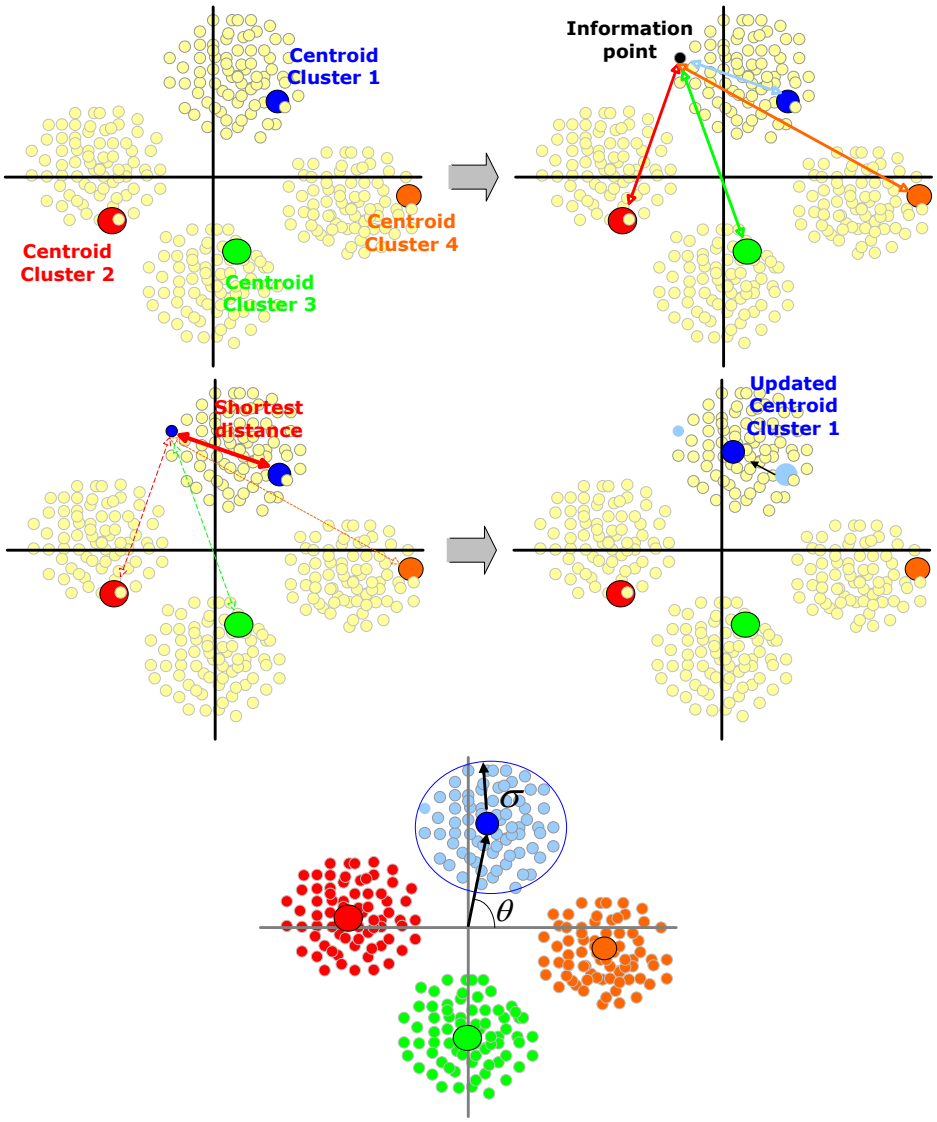

Figure 27: The initialization of the clusters is performed by picking one random point and calculating its M_{psk} points displaced $2\pi/M_{psk}$.

8.4.4 Probabilistic model

We now use all the information to classify all the points in our SubPskMatrix. For each subPsk we evaluate all the points for M_{psk} spherical Gaussian obtained from the EM algorithm, resulting a multidimensional matrix of $L_q \times N \times Q$ where L_q is the length of each subpsk, N is M_{psk} and Q the number of sub psk. In that matrix, the number of columns for each plane (M_{psk}) is ordered by the symbol identification; so all we have to do now is, for each sub psk and for each row, pick the highest value, and check the column where it is placed, which will indicate the symbol.

After it, we have the signals demodulated in a matrix of $L_q \times Q$. And the only remaining thing to do is multiplex them again in time domain. The code used for all these actions is attached below; where `ydown` variable is the data set already demodulated.

```
% Extracting Sub Psk Matrix
len = length(ydown)/num_psk;
SubPskMatrix = zeros(len, num_psk);
for q=1:num_psk
    SubPskMatrix(:,q) = ydown(q:num_psk:end);
end

% Calculating means and variances by EM.
mean = zeros(M_psk,2,num_psk);
sigma2 = zeros(M_psk,num_psk);
for q=1:num_psk
    [mean(:,:,q) sigma2(:,q)] = Kmeans(SubPskMatrix(:,q), M_psk,
        phase_order,enable_plots);
end

% Building the probability models for each subpsk
pK_x = zeros(len,M_psk,num_psk);
for q=1:num_psk
    subPsk = [real(SubPskMatrix(:,q)) imag(SubPskMatrix(:,q))];
    for k=1:M_psk
        sig = [sigma2(k,q),0;0,sigma2(k,q)];
        pK_x(:,k,q) = mvnpdf(subPsk ,mean(k,:,q), sig);
    end
end

% Demodulating each subpsk using P(K/x)
xrx=zeros(len,num_psk);
for q=1:num_psk
  for l=1:len
      [col row] = find (pK_x == max(pK_x(l,:,q)));
      xrx(l,q) = row - (q-1)*M_psk -1;
  end
end

% Recontructing the original Psk
xrx = xrx';
xrx = xrx(:)';
```

9 Simulations

Next, several sub-systems are simulated to show the performance of the system.

9.1 8PSK with no phase offset or dispersion

In the first case, we tested an 8PSK system with no phase offset or dispersion. It can be seen in figure 28 that the simulation points meet the theoretical BER of the 8PSK (red) and QPSK (blue).

```
N bits      : 18000 bits
Bit rate    : 1e+09 bps
Dispersion  : 0 ps/nm
Phase offset: 0 rad
```

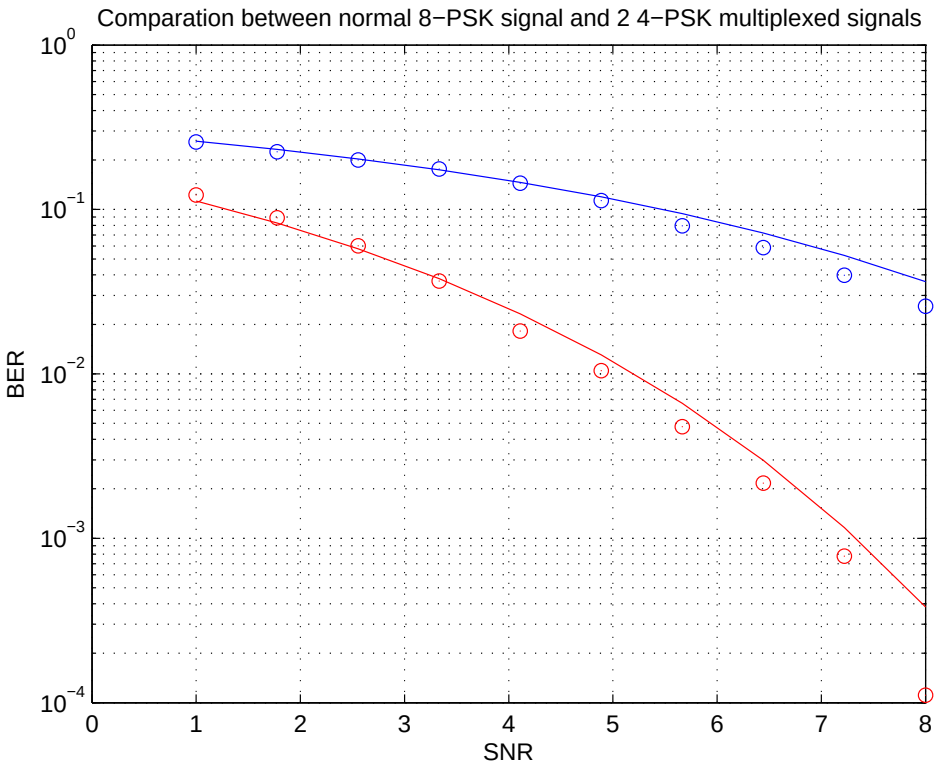

Figure 28: BER curves

9.2 8PSK with phase offset and no dispersion

In this case, the EM algorithm is tested to see how it can solve problems like phase offset by making the demodulator start demodulating with 0.65 radians of offset. It can be seen in figure 29 that almost half of the points are out of the decision region which makes BER extremely high. However, using the probabilistic model based in EM (figure 30), it can be seen that the decision boundaries are corrected as a function of the phase offset.

```
N bits       : 12000 bits
Bit rate     : 1e+09 bps
Tx time      : 9e-10 sg
SNR          : 8 dB
Dispersion   : 0 ps/nm
Phase offset: 0.65 rad
```

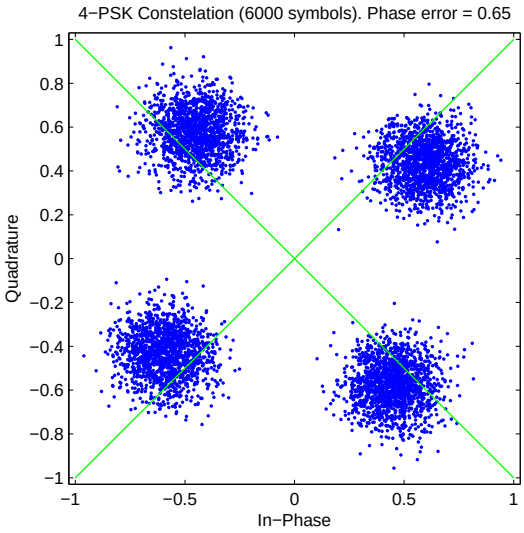

Figure 29: $BER = 0.15392$

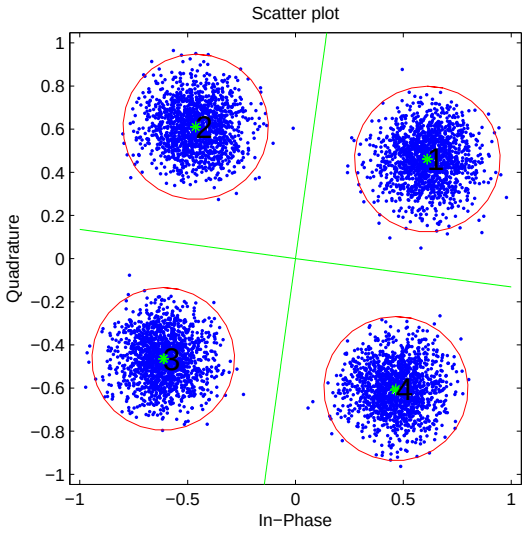

Figure 30: $BER = 0$

9.3 8PSK with dispersion

Now the system is tested in a dispersive environment, with SNR and dispersion. Since it has been demonstrated that it works well with the coupling of phase offset, it will be skipped in order to make other comparisons with the current system.

```
N bits       : 18000 bits
Bit rate     : 1e+03 bps
Dispersion   : 2.6 ps/nm
Phase offset: 0 rad
```

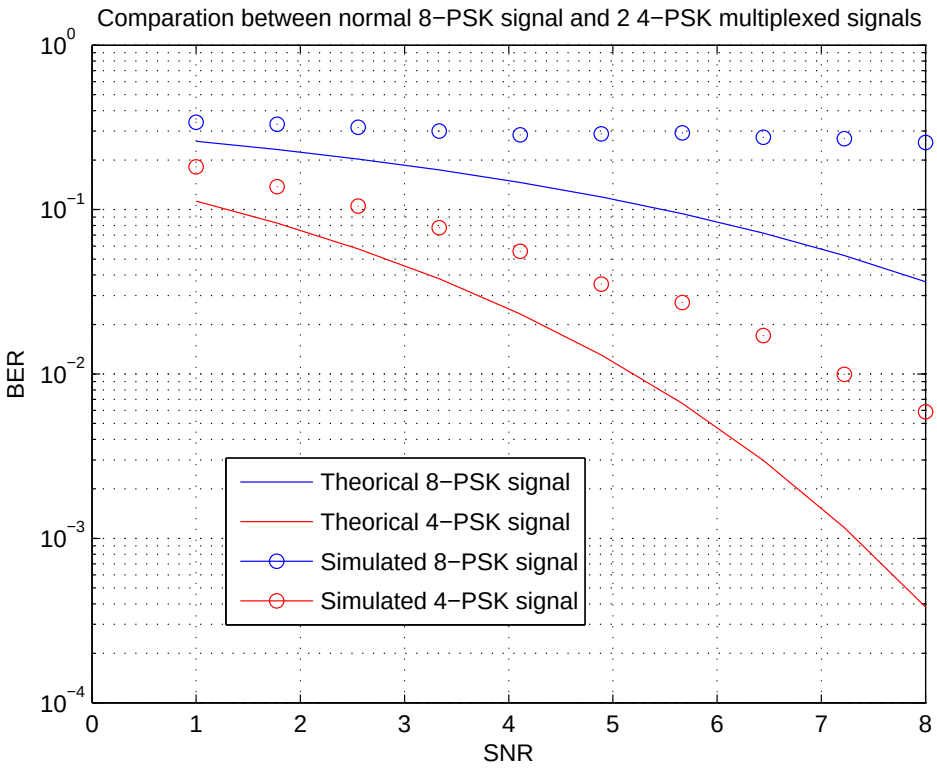

Figure 31: BER curves

9.4 64PSK with dispersion, and phase offset

A failed simulation was encountered. It is clear that the EM algorithm works fine when the clusters are relatively defined. The effects of the dispersion in one 64PSK constellation are shown in figure 32. In that figure, a phase offset of $\pi/4$ is applied; and it is at those points (and its conjugate) where the effect shows its most devastating consequences. Thus, the EM algorithm will not be able to identify which points are belonging to which correct symbol.

```
N bits       : 36000 bits
Bit rate     : 1e+03 bps
Tx time      : 0.002625 sg
SNR          : 100 dB
Dispersion   : 2.6 ps/nm
Phase offset: 0.7854 rad
BER          : 0.62164
```

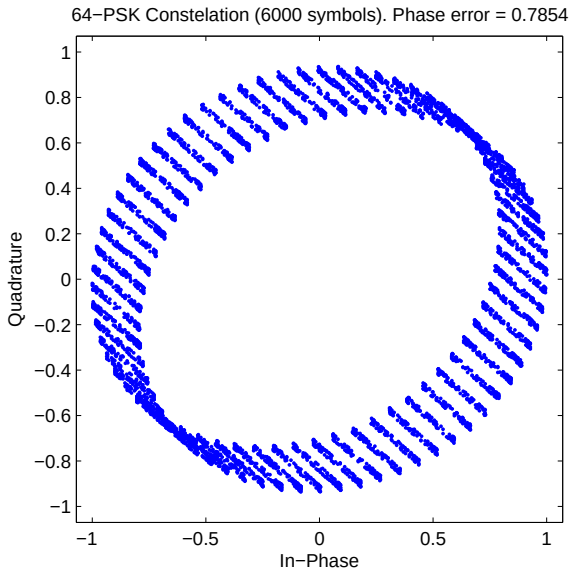

Figure 32: Dispersion effect in 64PSK constellation

Even if the constellations are split into 4 sub 16PSK, the algorithm has several problems to identify the symbols 2 and 10. (Figure 33). Higher splitting could be used, but then symbol ambiguity problem would be encountered. Since the clusters are too close, the K-means algorithm will assign the wrong symbol identifier to each cluster. As mentioned before, it could be solved with a pre-training set, but then the time computing would be too high.

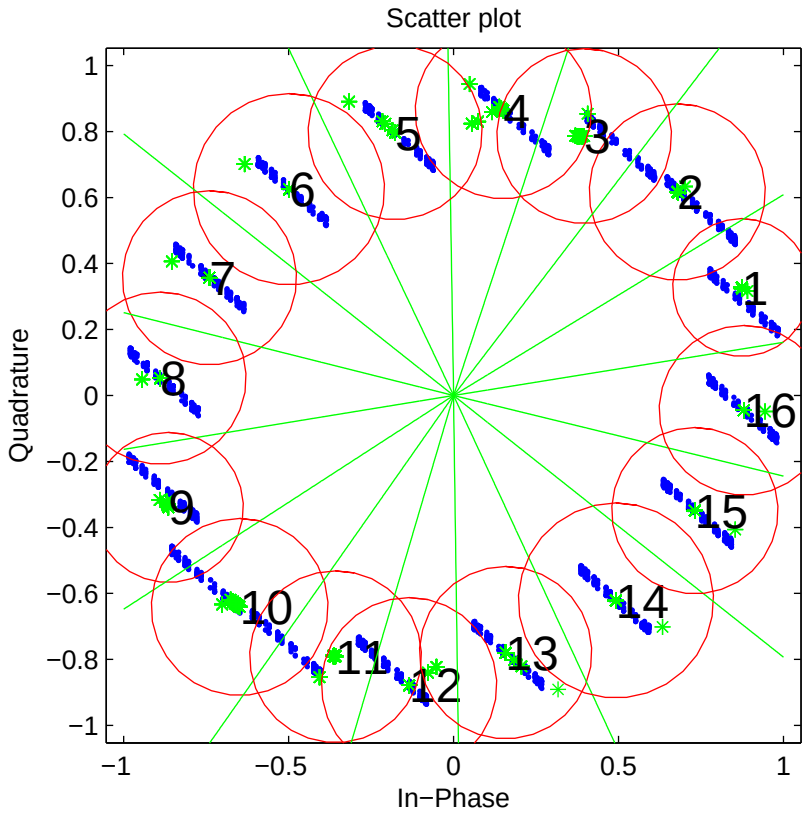

Figure 33: Sub 16PSK from 64PSK

9.5 16PSK with SNR, dispersion, and phase offset

The final simulation tries to show how it would work in a real environment. In this case all the impairments combined with a high symbol capability will be used (figure34).

```
N bits       : 4000 bits
Bit rate     : 1e+03 bps
Dispersion   : 2.6 ps/nm
Phase offset: 0.2 rad
```

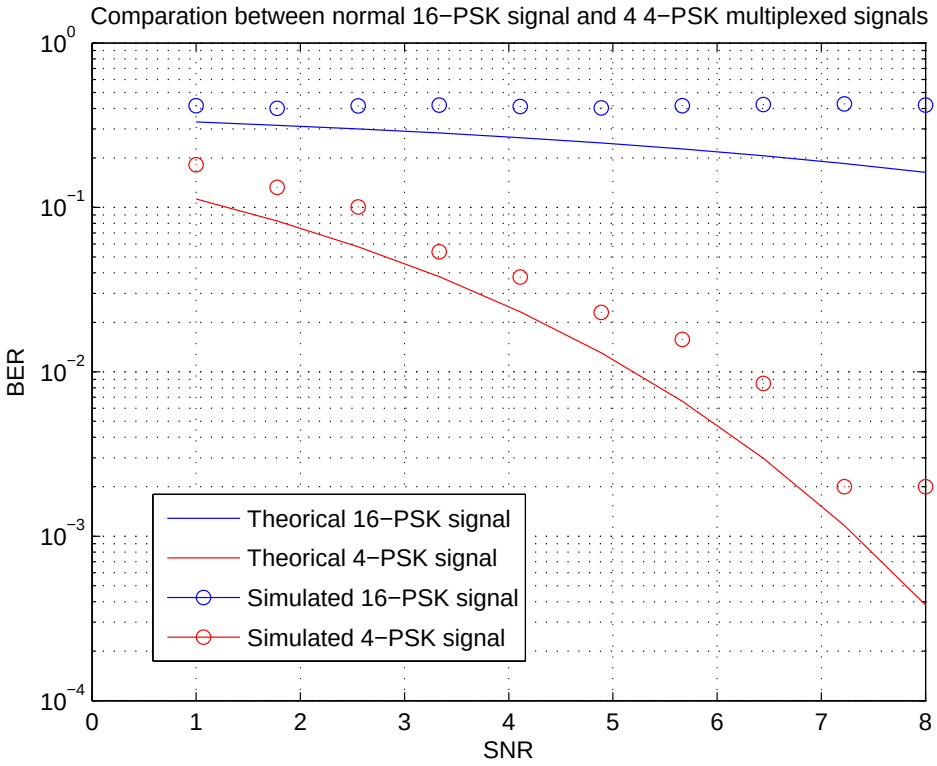

Figure 34: BER curves

One interesting conclusion from this simulation, is that either from the phase offset or from the constellation widening due to the dispersion and the phase offset, the points classified by our model best fit its theoretical BER curve (red). However, the simulated points classified by the conventional decider don't improve with higher SNR values.

10 Conclusions

This paper has explained the central importance of the decider module in digital communications. We now can take the power of the digital signal processing and combine it with artificial intelligence algorithms to construct robust systems able to rest importance to the channel. Our goal has been to create an understanding between the transmitter and the receiver, and let them split the problem into smaller ones. The methodology used to solve the smaller problems has been based in a different point of view taken from the decider module. We based the decisions made in that of module in the posterior probabilities, instead of the priori probabilities. In other words, we used the information coming from the channel to make a better decision.

This project was undertaken to design new kinds of receptors and evaluate how the intelligence algorithms can contribute to the digital communications. The solution here discussed is not only available to work in the scenario here studied, but also works for a lot of other impairments not linked to fiber channels.

The results of this investigation show an improvement in the BER curves over existing systems. Under this scenario, it seems that the system is able to solve a big amount of Gaussian noise, phase constellation coupling, and dispersion problems.

The following conclusions can be drawn from the present study. First, a difference of the current systems, a mistake taken from the EM algorithm could be disastrous as we saw in the simulation 9.4. Also, any misunderstanding in the sub psk identifier at the time of the demultiplexing process can flow into a complete bad resignation of the original bits transmitted. However, if we test the system in scenarios not higher than 16 symbols this algorithm would unlikely fail. This research will serve as a base for future studies, and taken into account the progress of the technology nowadays, this branch of investigation could begin to gain importance.

Furthermore research might explore a better assignation of the bit map in order to improve the hidden Markov background in this study. In this case, the symbol was coded lower than M-PSK, and applied the phase differences between Q symbols, flowing in a penalty in the bit rate. But in a better use of the Markov theory, a better information capability in the whole system can be achieved, getting a real improvement both in the bit rate as in the bit error rate.

References

[1] **Optical Fiber Telecommunications V: B Systems and Networks**, *Ivan P. Kaminow*, 2008

[2] **Digital Communications-Fundamentals and applications**, *Bernard Sklar*

[3] **Pattern Recognition and Machine Learning**, *Christopher M. Bishop*

[4] **Digital filters for coherent optical receivers**, *Seb. Savory*, 2008.

[5] **Simulation Framework**, Miguel Iglesias Olmedo, Neil Guerrero González `http://www.sundara.es/data/simulation_framework.zip`