A Dirty Paper Coding MODEM

Tomás Law, Student, Instituto Superior Técnico

***Abstract*—Dirty Paper Coding (DPC) is a set of techniques for reducing the effects of interference on a signal across the channel by using precoding techniques at the transmitter to shape and transform the original message. It allows the cancelation of arbitrary known interference at the transmitter while keeping the power required to transmit the signal constrained. The main objective of the dissertation was to implement a complete end-to-end transmission system which employs DPC. Obviously, both ends of the system need to be coordinated and compatible. This involves implementing the encoding techniques performed at the transmitter and the respective decoding techniques at the receiver. The implemented system was used to perform several simulations where the noise parameter in the channel will vary. From these simulations, the Bit Error Rate (BER) and the total number of errors on the decoded message will be measured for different Signal to Noise Ratio (SNR) conditions. The simulations were performed using *Matlab* software.**

***Index Terms*—DPC, Transmitter, Receiver, Interference, SNR.**

I. Introduction

In a transmission system, usually we can consider three main components: a transmitter, a channel and a receiver. The transmitter is responsible for encoding the original message W into an appropriate format which is suitable for transmission across the channel. At the channel, the transmitted signal X will suffer the effects of other interfering signals which are seen as noise to the system. The receiver is responsible for decoding the received signal Y into the original decoded message W'. However, the decoded message at the receiver might not match the original message sent at the transmitter because of the interference in the channel. The received signal Y can be represented as

$$Y = X + S + N \tag{1}$$

where S is arbitrary interference known at the transmitter and N is a statistically independent Gaussian random variable representing noise. This scenario is illustrated in Figure 1.

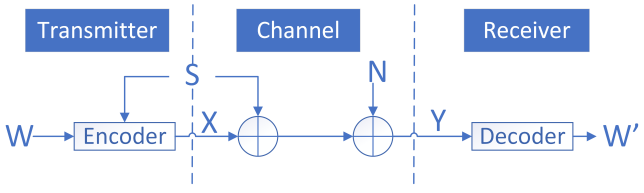

Fig. 1. Transmission steps in a communications system

Signal interference problems are a recurring issue on many people's lives. Ranging from connections droppping off due to a low received power of the signal to erroneous messages being delivered to the receiver. The usage of multiple-input multiple-output (MIMO) systems has been significantly increasing. So has the use multiple user MIMO (MU-MIMO) systems, where the same medium is shared by several users. In MU-MIMO systems interference between users can be reduced by using DPC. The signal sent from a base station to one user can be seen as known interference to the signal sent to another user. With this project, we intend to increase the quality and reliability of nowadays digital transmission services.

The effects of interference on the signal can be reduced through several different techniques. Dirty Paper Coding (DPC) is a technique for transmitting data efficiently across a channel with interference. It uses several methods to precode the data at the transmitter side so that it becomes less vulnerable to the effects of interference across the channel. It allows the cancellation of the effects of a portion of interference known at the transmitter, without increasing the overall power necessary to transmit the signal. This cancellation will allow us to reach a close-to-capacity transmission rate. That is, a transmission rate close to the maximum transmission rate possible at which information can be reliably transmitted. Therefore, we achieve a more efficient, but still reliable, signal transmission system.

One of the applications of DPC is in digital watermarking which consists of embedding one signal, embedded signal or watermark, within another signal, host signal. This method allows the transmission of auxiliary information where the embedded signal causes no serious degradation to its host signal.

The concept of DPC was originated way before the rise of research in digital watermarking in 1983 where Costa postulated the following problem:

"imagine we have a piece of paper covered with independent dirty spots of Normally distributed intensity, and we write a message on it with a limited amount of ink. The dirty paper, with the message on it, is then sent to someone else, and acquires more Normally distributed dirt along the way. If the recipient cannot distinguish between the ink and the dirt, how much information can we reliably send?", [5]

The problem exposed by Costa makes an analogy to a real communications channel scenario where the dirt represents the noise found in the channel. As the amount of noise (dirt) found across the channel increases, the amount information that we can reliably send will be lower. However, Costa proved that the same isn't true for interference known at the transmitter (dirt added before sending the message) and showed that the capacity C of the system depends solely on the signal power P and the channel noise N as given by

$$C = \frac{1}{2}\log_2\left(1+\frac{P}{N}\right) \tag{2}$$

where B represents the sampling rate. This is true as long as the transmitted signal X was power constrained as shown in Equation (3).

$$\frac{1}{n}\sum_{i=1}^{n} X_i^2 \leq P \tag{3}$$

A. Paper Outline

This dissertation will start by giving an overall explanation regarding data transmission across channels and pointing out the problems caused by interference along with some examples of solutions. Then, the dirty paper coding solution will be introduced along with the challenges associated with each element of it. Also, a brief presentation regarding the evolution of this approach throughout the history such as the major breaking points and developments in this area. Most importantly, before explaining the system implemented to achieve our goal, some background information will be provided about the main concepts involved in this project such as *lattices*, *shaping gain* and log-likelihood.

In the implementation section, the MODEM is described from 2 different point of views: the transmitter side and the receiver side. For each side, we explain what is objective, how it will be achieved and the objective of the modules such as Repeat-Accumulate Code, Viterbi and Bahl-Cocke-Jelinek-Raviv algorithms. After the whole implementation process is explained, results obtained from the simulations with the system we implemented will be presented along with a proper explanation and analysis. Finally, a brief summary about the execution of the project and some thoughts about applications of DPC in existing systems will be given. Special relevance regarding the relationship between DPC and MIMO systems will be taken into consideration and future work to be done in this process will be mentioned.

II. Background

A. Evolution

Initially, in [2], Max Costa studied the DPC with Gaussian noise and interference and showed that the capacity is equal to $\frac{1}{2}\log_2\left(1+\frac{P_X}{P_N}\right)$. However, the paper didn't address the relevance of the results to common communication problems so it didn't draw much attention. Only recently, as mentioned in [1], *"the connection of the DPC model to precoding for interference cancellation was established, and Costas result was extended to arbitrary interference, deterministic or random"*. A scheme based on lattice quantization and minimum mean-squared error (MMSE) scaling was suggested and was extended by using capacity-approaching codes, iterative detection and iterative decoding. After further researches, a complete end-to-end dirty paper transmission system that offers considerable gains under low conditions of signal-to-noise ratio (SNR) was designed.

B. Lattices

As defined in [1], *"a lattice Λ is a discrete subgroup of the Eucliden space $\mathbb{R}^n$"* and it can be regarded as a linear code defined over the real numbers. To ease the process of understanding the concept of a lattice, we can work with the definition that a lattice is represented by a linear combination of basis vectors in N-dimensional space. Therefore, each lattice point $\lambda \in \Lambda$ can be represented by Equation (4).

$$\lambda = \mathbf{g_1}b_1 + \mathbf{g_2}b_2 + ... + \mathbf{g_n}b_n \tag{4}$$

where $\mathbf{g_1}, \mathbf{g_2}...\mathbf{g_n}$ are basis vectors and $b_1, b_2, ..., b_n$ are integers. We can immediately see that if we take all integers $b_i = 0$, then $\lambda = \mathbf{0}$. That is, the **origin is always a lattice point**.

The ordinary vector addition operation states that given two points $\mathbf{x^T} = (x_1, x_2, ..., x_n)$ and $\mathbf{y^T} = (y_1, y_2, ..., y_n)$, the addition of these two points is given by Equation (5). Additionally, since the lattice is linear, if these two points belong to the lattice, then their sum also belongs to the lattice. That is, if $\mathbf{x}, \mathbf{y} \in \Lambda$, then $\mathbf{x}+\mathbf{y} \in \Lambda$.

$$\mathbf{x^T} + \mathbf{y^T} = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n) \tag{5}$$

The Squared Euclidean Distance represents the straight-line squared distance between two points in Euclidean space defined by Equation (6).

$$||\mathbf{x}-\mathbf{y}||_2 = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{6}$$

An example of a two-dimensional lattice is shown in Figure 2 where we have two generator vectors $\mathbf{g_1}$ and $\mathbf{g_2}$.

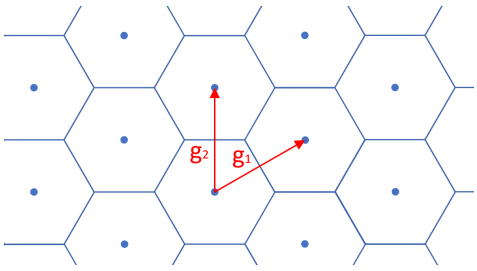

Fig. 2. Lattice representation

The lattice Λ can also be translated by $\mathbf{x} \in \mathbb{R}^n$ so, consequently, all lattice points λ are translated by $\mathbf{x}$. The translated version of Λ defined by the set $\mathbf{x}+\Lambda$ represents a *coset*. An example coset for a two-dimensional lattice is shown in Figure 3.

$$Q_\Lambda(\mathbf{x}) = \lambda \in \Lambda \quad \text{if} \quad ||\mathbf{x}-\lambda|| \leq ||\mathbf{x}-\lambda'||, \quad \forall\, \lambda' \in \Lambda \tag{7}$$

The nearest neighbor quantizer $Q_\Lambda(\mathbf{x})$ represents the lattice point that is the closest to the sequence $\mathbf{x}$. The distance between $Q_\Lambda(\mathbf{x})$ and $\mathbf{x}$ is calculated using the modulo lattice operation.

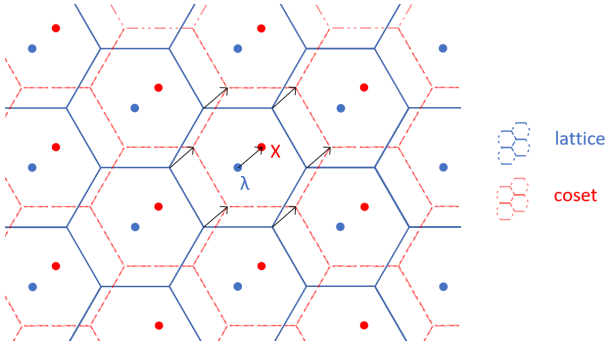

Fig. 3. Coset of the lattice Λ generated by $\mathbf{x}$

1) Modulo lattice: The modulo lattice operation is used to compute the difference between the lattice point λ that is the closest to a certain sequence $\mathbf{x}$ and the sequence $\mathbf{x}$ itself. Finding the closest lattice point is accomplished by computing the minimum squared Euclidean distance between $\mathbf{x}$ and any of the surrounding lattice points $\lambda_\mathbf{i}$. However, a problem arises: how does the algorithm know which lattice points are surrounding the sequence $\mathbf{x}$ and which aren't. The solution for this problem relies on the fact that lattices are periodic in $n\mathbb{Z}^n$. Therefore, we can translate any sequence $\mathbf{x}$ to the proximity of the lattice point centered at the origin by performing a modulo operation as shown in Equation (8).

$$\mathbf{x}' = \mathbf{x}\ mod\ \mathbb{Z}^n \tag{8}$$

Then, we can compute the Euclidean distance between the translated sequence $\mathbf{x}'$ and the surrounding lattice points to find which one is the closest as shown in Equation (9). After we perform this operation, we have found the *coset leader* $\mathbf{y}$ which is defined as the unique member of the coset lying in the fundamental Voronoi region.

$$\mathbf{y} = \mathbf{x}'\ mod\ \Lambda = (\mathbf{x}\ mod\ \mathbb{Z}^n)\ mod\ \Lambda \tag{9}$$

2) Voronoi region: The fundamental Voronoi region of $\Lambda \subset \mathbb{R}^n$ denoted by $\mathcal{V}$ is defined as the set of minimum Euclidean norm coset representatives of the cosets of Λ. The Voronoi region of a lattice point λ is the space which is closer to λ than to any other lattice point as represented in Figure 4

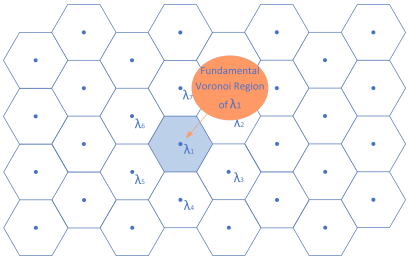

Fig. 4. Representation of the Fundamental Voronoi Region of λ_1

The fundamental Voronoi region of a lattice can be computed by performing the modulo lattice operation over $\mathbb{R}^n$ as shown in Equation (10).

$$\mathcal{V}_0 = \mathbb{R}^n\ mod\ \Lambda \tag{10}$$

The fundamental Voronoi region $\mathcal{V}_0$ defined by Equation (11) corresponds to the set of all points that are closer to the origin lattice point than to any other lattice point. That is, the sequences $\mathbf{x}$ whose nearest neighbor $Q_\Lambda(\mathbf{x})$ is the origin.

$$\mathcal{V}_0 \triangleq \{\mathbf{x} \in \mathbb{R}^n : Q_\Lambda(\mathbf{x}) = 0\} \tag{11}$$

The fundamental Voronoi region is the key element in setting the power constraint on the transmitted channel. The objective of the modulo lattice operation is to map the information sequences into the fundamental Voronoi region $\mathcal{V}$ because this means that transmitted sequences correspond to the minimum energy sequence. This gain translates to a gain in signal power when we use modulo-lattice coding for the DPC.

3) Shaping gain: A useful way of measuring the gain from using the lattice method is the shaping gain $g_s(\Lambda)$:

$$g_s(\Lambda)|_{dB} = 10\log_{10}\frac{G(\mathbb{Z})^n}{G(\Lambda)} = 10\log_{10}\frac{1}{12G(\Lambda)} \tag{12}$$

where $G(\mathbb{Z})^n$ is the normalized second moment of a hypercube of any dimension (no shaping) and $G(\Lambda)$ is the normalized second moment of the lattice (using $\mathcal{V}$ for shaping). It measures how much more power is needed when using an input uniformly distributed over a cube, rather than a distribution uniform over the Voronoi region $\mathcal{V}$, in order to obtain the same entropy.

C. Viterbi algorithm

The Viterbi algorithm is a dynamic programming algorithm which provides an efficient way of finding the most likely state sequence of a finite-state discrete-time Markov process in memoryless noise. The estimated state sequence is called **Viterbi path**. State x_k represents one of a finite number M of states m at one of K discrete time instants k where $1 \leq m \leq M$ and $0 \leq k \leq K$. The underlying key for estimating the state sequence involves taking advantage of a Markov property which affirms that the future state of a process depends solely on the present state, not on the sequence of events that preceded it. More specifically, the probability of state x_{k+1} depends solely on the state x_k, not on the previous states $x_{k-1}, x_{k-2}, ..., x_0$, as described in Equation (13). The state transitions will be dependent on the input of the Markov process.

$$p(x_{k+1}|x_0, x_1, ..., x_k) = p(x_{l+1}|x_k) \tag{13}$$

In the context of our problem, Viterbi will be used to find the lattice codeword that most closely matches the received information sequence. This information sequence corresponds to the sequence of input bits after several transformations that have been performed. Our lattice points correspond to codewords generated from the convolutional code and we will map each information sequence to the closest lattice point. In other words, the Viterbi algorithm finds what state sequence causes the most similar codeword sequence to the one actually received. The convolutional code can be modeled by a shift-register in which the shift-register bits represent the state

and the shift-register input represents the input bit u_k. An explanation with a step-by-step description and evolution of the algorithm we used will be further detailed in the next chapter.

D. Log-Likelihood

In the algorithms described in this system, the use of Soft-In Soft-Out (SISO) decoders is highly important. These type of decoders allow the input and output of each operation to take values other than 0 or 1.

More specifically, all the bits will be represented by a log-likelihood value (L-value) which is calculated as.

$$L(u) = \ln \frac{p(u=1)}{p(u=-1)} \tag{14}$$

where $p[u=1]$ is the probability of the bit u to be equal to '0' and $p[u=-1]$ is the probability of the bit u to be equal to '1'. In this notation, the sign of $L(u)$ indicates the hard decision (bit '0' or '1') and the magnitude of $L(u)$ indicates the reliability of this decision (how likely the hard decision is true).

III. Implementation

A. Designed system

We modeled a system which allows the simulation of the three main components in a general transmission system: transmitter, channel and receiver. The overview of the implemented system is depicted in Figure 5.

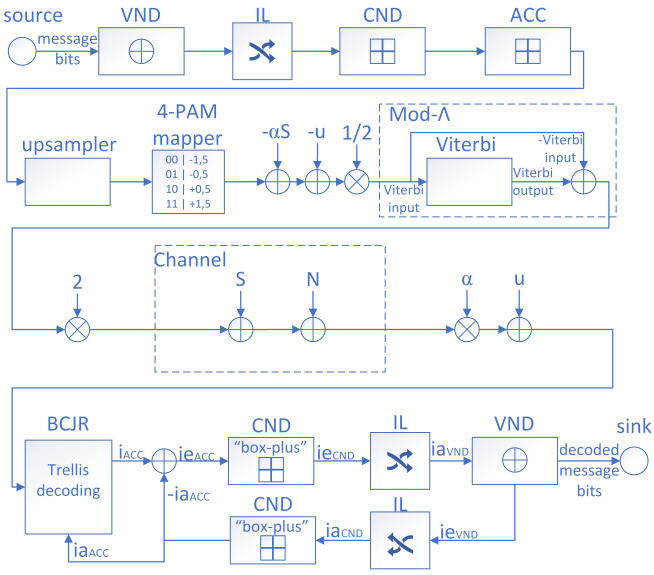

Fig. 5. Full overview of the designed system

1) Transmitter: At the transmitter side, we start with the message we wish to send. The coding of the original message is performed by several elements which can be simplified into two main modules. The first module includes an error-correcting code which consists of a nonsystematic repeat-accumulate (RA) encoder. After the repeat-accumulate code has been applied, the second module of the transmitter further transforms the signal by applying a trellis shaping code. This module consists of several signal shaping operations with the main goal of minimizing the energy of the signal before transmitting across the channel.

For this shaping operation, the Viterbi algorithm is used which will provide us the closest lattice codeword to each signal symbol but translated to the fundamental Voronoi region. This is performed by transmitting only the distance between each signal symbol and the closest lattice codeword so we will subtract Viterbi's output by Viterbi's input. This way, we guarantee that the transmitted signal is inside the fundamental Voronoi region and ready to be transmitted.

2) Channel: Across the channel, the signal will be affected by two sources of interference:

- The state interference S which was known at the transmitter and already pre-subtracted with a scale factor α;
- An Additive White Gaussian Noise signal n

3) Receiver: At the receiver side, after rescaling the signal by factor α and removing the dither u, we follow a different approach. The Viterbi algorithm "minimizes the probability of word error for convolutional codes. The algorithm does not, however, necessarily minimize the probability of symbol (or bit) error" [3]. The solution involves using the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm which minimizes the symbol error probability. More accurately, the Viterbi algorithm only allows the estimation of the probability of the state of the Markov process, while the BCJR will allow to determine the probability of the input bits that drive the state changes in the Markov process. This decoder is used to estimate the bit probabilities by using the folded Euclidean metric which will be described ahead.

A Bahl-Cocke-Jelinek-Reviv (BCJR) *a posteriori* probability (APP) decoder is used. This decoder is used to estimate the bit probabilities by using the folded Euclidean metric which will be described ahead. The BCJR algorithm receives the *a-priori* L-values and provides the *a-posteriori* L-values. Then, the extrinsic information from the values is forwarded to the channel decoder which will refine the estimations using the redundancy in the coded bit stream. The decoder will iterate over these steps until the bit probabilities converge. These iterations are responsible for conducting the Belief Propagation algorithm which consists of a sum-product message passing algorithm. That is, we are computing the marginal probabilities of the L-values at each module based on L-values originated from other modules. These L-values represent messages that are transmitted between the different modules of the algorithm which increase the amount of information known at each module until it converges to a solution.

B. Repeat-Accumulator Code

1) Introduction: Repeat-accumulator codes are a class of error correcting codes and they have some similarities to turbo codes in the sense that they are also decoded iteratively and allow to come close to capacity. RA codes can be systematic or non-systematic and they are generated by convolutional coders. As described in [4], irregular RA codes consist of an outer code that is a mixture of repetition codes, an inner

code that consists of parity checks and an accumulator. More specifically, it consists of a variable node coder (VND), an interleaver (IL), a check node coder (CND) and an accumulator (ACC).

2) Structure: Our encoder receives a stream of *k* information bits which enter the first part of our repeat-accumulate coder: the repetition codes. These codes are represented by variable nodes which have different degrees depending on the rate of the repetition codes. After the repetition codes, there's an interleaver, represented by an edge interleaver, which will foward the bits to the parity check codes. These codes are represented by check nodes which, similarly to the variable nodes, have different degrees depending on the rate of the single parity check code. Finally, after the single parity check codes, there's an accumulator represented by a chain of check nodes.

At the decoder side of the RA code, more specifically, the inner decoder side, we perform the "BCJR decoding process" which will be detailed later in Section III-E. This algorithm receives as input the a-priori probabilities of the accumulator's input bits in the L-value format. Then, the a-posteriori L-values are fed back to the check nodes. The objective of this RA decoder is to use the redundancy in the bits, which was intentionally created with the variable nodes, to improve the accuracy of the L-values. An iteration of this decoding algorithm will be detailed in the next section.

3) Decoding algorithm: An iteration of this algorithm starts by subtracting the resulting *a-posteriori* L-values of the BCJR decoding algorithm by the *a-priori* L-values to obtain the extrinsic L-values. These extrinsic L-values are forwarded from the accumulator to the check nodes. The next step of this process corresponds to forwarding the output L-values from the check nodes to the edge interleaver.

An iteration of this algorithm starts by the forwarding of the resulting a-posteriori L-values of the BCJR decoding algorithm from the accumulator to the check nodes. The next step of this process corresponds to forwarding the output L-values from the check nodes to the edge interleaver. Each check nodes has d_c+1 L-values associated where d_c L-values come from the interleaver and 1 comes from the accumulator. These L-values are used as inputs to the CND and they will be used to calculate the output L-values of the CND

$$L_{i,out} = ln \frac{1 - \prod_{j \neq i} \frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}}{1 - \prod_{j \neq i} \frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}} \tag{15}$$

which can be denoted as a "box-plus" operation using the L-value notation of [6].

$$L_{i,out} = \sum_{j \neq i} \boxplus L_{j,in} \tag{16}$$

Now, the CND output L-values are forwarded to the interleaver which will permutate them and forward them to the VND where we will calculate the VND output L-values. Similarly to the calculation of the CND output L-values, the VND output L-values are obtained from the VND input L-values. Each VND has d_c edges: d_c edges coming from the IL and 1 from the source. However, the *a-priori* L-value of the source edge is always zero because we have no *a-priori* information about the coded bits. Each VND output L-value is calculated using Equation (17).

$$L_{i,out} = \sum_{j \neq i}^{d_c} L_{j,in} \tag{17}$$

Since each L-value contains information about the hard decision and the reliability of that decision, by adding the input L-values, all this information combined will produce a more complete and reliable L-value. At this point, we've computed an estimate of the input bit probability but, in order to further improve the accuracy of this value, we should perform more iterations of this decoding algorithm. Now, the following L-values correspond to the flow of the L-values in the opposite direction: the IL-to-ACC direction. The VND output L-values are fed back to the IL which will permutate them and forward them to the CND. Then, the "box-plus" operation will be performed again with the newly received L-values from the IL to provide the *a-priori* L-values on which the BCJR algorithm will operate.

C. 4-PAM Mapping, Dither and Interference

After the RA code has been applied to the input bits, we now have a signal composed of a sequence of coded bits at the output of the accumulator. However, before applying the Viterbi algorithm to find the corresponding codeword, we need to perform some transformations to the signal which mainly consist of two important operations: *map* and *dither addition*. This sequence will be upsampled and then mapped to real scalar values. The upsampler generates 4-PAM symbols by grouping every 3 bits into two pairs of bits. The 2nd bit of each group of 3 bits is repeated in both pairs and corresponds to the most significant bit in each pair. Each pair of bits represents a 4-PAM symbol. Then, each of these symbols will be mapped to a real scalar value using a mapper centered at the origin which generates values in the set $\{-1.5, -0.5, 0.5, 1.5\}$. The mappings from the 4-PAM symbols to the real values are described in Table I.

TABLE I
MAPPING OF THE 4-PAM SYMBOLS TO REAL VALUES

Symbol	Mapped value
00	-1.5
01	-0.5
10	0.5
11	1.5

The resulting sequence of real values will now be subtracted by the *scaled interference* and the *dither*. This scaled interference corresponds to the arbitrary interference known at the transmitter which will be added in the channel.

The dither consists of a random variable that is intentionally subtracted to the signal. Its main characteristic is the uniform distribution centered at the origin and it's used to shape the signal with one main objective: to make the signal uniformly distributed in the Voronoi region.

We want to transform the signal into an uniformly distributed signal before sending across the channel because the mutual information of the channel is maximized for an uniform input. A deeper explanation on this is in [1] but what we need to infer is that, this way, instead of sending the input signal directly, we send the sum of the signal with the dither. In other words, this will allow us to describe the transmitted signal (which will be sent across the channel) by an uniform signal that is independent of the input signal. This is a key characteristic of the signal which will allow the methods we implemented to operate correctly.

One more operation will need to be performed to transform the signal. The lattice points are represented by $2(\mathcal{C} + 2\mathbb{Z}^N)$ and not $(\mathcal{C} + 2\mathbb{Z}^N)$ because we are shaping with the sign-bit which has a weight of 2. Since the 4-PAM operation generates values in $\{-1.5, -0.5, 0.5, 1.5\}$, after they are replicated by adding multiples of 2, an infinite sequence of values spaced by 1 will be obtained $\{..., -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, ...\}$. Therefore, Viterbi's fundamental Voronoi region is in $[-1 \quad 1]$. The signal values need to be reduced by half before executing the algorithm. The Viterbi algorithm can now be applied to obtain the lattice codewords corresponding to the *dithered information sequence with scaled interference*.

D. Viterbi Algorithm

1) Introduction: The Viterbi element in our system is placed after the dither and scaled interference addition and receives the information sequences. It serves a goal different than the original objective of the Viterbi decoder. Instead of finding the input bits, we use it to generate the closest codeword. It will generate codeword sequences based on the generating polynomials to build a convolutional code. These codeword sequences represent the lattice points.

$$b_i(D) = g_i(D) * u_{vq}(D) \tag{18}$$

where $g_i(D)$ is a generating polynomial and $u_{vq}(D)$ ranges over all binary input sequences.

In our scenario, this distance of the codewords corresponds to the modulo Euclidean distance between the two sequences. This minimization process is represented by Equation (19) where x_k represents an information sequence and b_k represents a codeword from the lattice.

$$min \left\{ \sum_{k=1}^{n} ((x_k - b_k) \, mod \, 2)^2 \right\} \tag{19}$$

As earlier introduced in the previous chapter, this algorithm estimates what was the most likely sequence of input bits, based on some observed output, by computing all possible outputs for each state and input bit combination. Yet, in our system, the observed output sequence corresponds to a sequence of bits, grouped in pairs. These pairs don't represent a codeword, they are just arbitrary information sequences outputted after the addition of the dither. We will denote these information sequences as **observed output**. To estimate the most likely state sequence we will be comparing each pair of bits from the observed output with the pair of bits generated from the convolutional code. That is, we will be finding the lattice point, represented by a codeword, that is the closest to the observed output. These lattice points were generated from a specific convolutional code which consists of the polynomials 5_8 and 7_8 to create the codewords. These polynomials were chosen by Uri Erez as mentioned in [1]. Most likely, there will be information sequences from the observed output that don't match any of the possible codewords generated. The purpose of the Viterbi algorithm is to find the codeword that is the closest to the observed output.

2) Mapping to the Voronoi region: The core property of this dirty-paper-coding technique, regarding the minimization of the interference, rests on the transmission of the signal with limited power. As described in the previous section, regarding lattices quantization, we need the signal to be inside the Voronoi region. This region is the region inside each cell of the lattices grid. And, if we recall the introduction about the lattices, each point of the grid corresponds to a certain codeword from a convolutional code. Therefore, returning back to the Viterbi algorithm, if we subtract off the codeword from the observed output performing the modulo lattice operation, we guarantee that the result is inside the Voronoi region. Then, we are ready to transmit the power constrained coded signal across the channel.

3) Implementation code: As mentioned before, the lattice codewords can be modeled by a shift-register from which we can describe the state evolution of the process using a trellis diagram. This trellis displays all possible state transitions for each time instant. It also describes what conditions are necessary for each state transition and what is the output codeword that was generated in that transition. The convolutional code used the polynomials 5_8 and 7_8. Therefore, to match the number of bits from the polynomials (3 bits), the shift-register was designed with 2 state bits which, along with the input bit, equal a total of 3 bits. One important property of this trellis is that it starts at the all-zero state and it must converge to a terminal all-zero state. This trellis method was implemented using a set of matrices representing the output codeword as a function of the state and the input of the shift register.

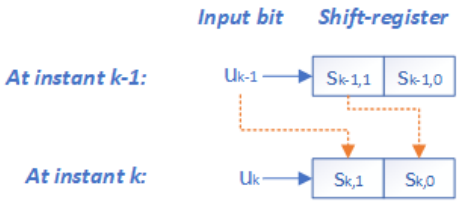

Fig. 6. State transition in a shift-register

4) Calculation of the lattice codewords: For creating our lattice codewords, we used a convolutional code which was generated by the polynomials $(g_1, g_2) = (05_8, 07_8)$ of memory 2 where $g_1 = [g_{1,2} \; g_{1,1} \; g_{1,0}] = [1 \; 0 \; 1]$ and $g_2 = [g_{2,2} \; g_{2,1} \; g_{2,0}] = [1 \; 1 \; 1]$. The lattice codeword will be computed as shown in Figure 7 where u represents the virtual input bit into Viterbi algorithm.

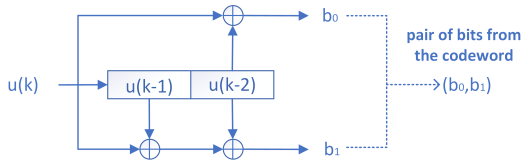

Fig. 7. Structure of the shift-register and the generator polynomials to obtain the lattice codewords

Then, these two bits of the codeword are compared to the information sequence of the observed output. The result of this comparison represents the metric used to estimate which of the possible previous states, is the most likely one. This metric corresponds to the total number of errors propagated for the respective path. The operations described above are executed for all instants of the algorithm. Figure 8 shows a trellis diagram for the initial and terminal phases of the algorithm. In this trellis, we represent the **transitions** between each state (0 - red, 1 - green), the **input bit** necessary to cause that transition and the **output** generated from that transition.

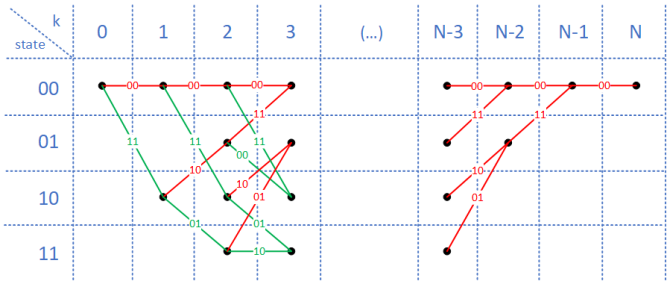

Fig. 8. Representation of the Viterbi algorithm in a trellis diagram

After all iterations of the algorithm have been computed, we now can easily obtain the Viterbi path by following the sequence of most likely previous states, starting at the terminal all-zero state. Along with the state sequence, the associated codeword is obtained. As previously mentioned, the true advantage of this system relies on the transmission of a signal which is inside the Voronoi region. So, if we subtract the resulting sequence of the map and interference and dither addition by the output codeword of the Viterbi algorithm, then the result will be inside the Voronoi region. So, now the resulting signal can be transmitted across the channel and guarantee that it satisfies the power constraint on the signal power.

E. Bahl-Cocke-Jelinek-Raviv Algorithm

1) Introduction: The Bahl-Cocke-Jelinek-Raviv (BCJR) decoder allows the estimation of the *a-posteriori* probability (APP) of the states and transitions of a Markov source through a discrete and memoryless channel (DMC). Unlike the Viterbi algorithm which minimizes the probability of word error, BCJR minimizes the probability of symbol/bit error probability. That's the reason for the designation of "bit-wise quantization detector". Since our system can be described in a space state, the output of the system is a Markov process. The number of possible states M is defined by the number of state bits and they are indexed by the integer $m = 0, 1, ..., M-1$. The state of the coder at a time t is denoted by S_t and the transitions between them are described by the transition probability $p_t(m|m') = Pr\{S_t = m|S_{t-1} = m'\}$. The initial and terminal states of the trellis must be the all-zero state. The output of the coder is denoted by X_t and it's described by the probability $q_t(X|m', m) = Pr\{X_t = X|S_{t-1} = m'; S_t = m\}$. This output X^t is transmitted through a noisy DMC resulting in the output Y^t. The transition probability of the DMC between the X_t and Y_t is denoted by $R(Y_t, X_t)$. Since the noise of the DMC is approximately AWGN, the transition probabilities will be given by the Normal distribution.

2) Behaviour: The algorithm estimates the APP of the states and transitions by comparing the outputs Y^t with the produced output by each combination of state bits for all possible states in the trellis. It performs a soft demapping of signed and unsigned bits. Let $Y_{t_1}^{t_2}$ be the set formed by the values of Y_t from $t = t_1$ to $t = t_2$ and consider a set of measured values from $t = 1$ to $t = \tau$. For each node, there's the APP $Pr\{S_t = m|Y_1^\tau\}$ associated and for each branch there's the APP $Pr\{S_{t-1} = m'; S_t = m|Y_1^\tau\}$. To calculate these values, expressions from [3] were used. These expressions use some auxiliary probabilities which allow us to calculate the desired probabilities. The state space model used for this decoding process can be inferred from Figure 9 which will be explained ahead.

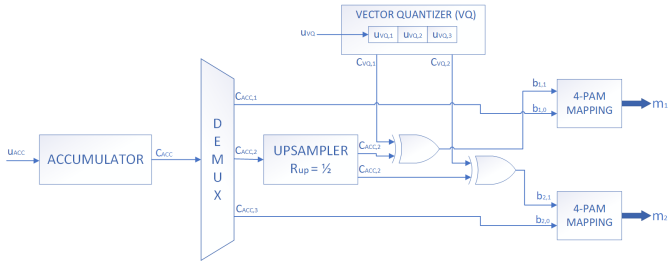

Fig. 9. Illustration of the trellis processing at the receiver side

3) Operations: The decoder has 4 state bits: 1 coded bit from the accumulator c_{ACC} and 3 state bits from the vector quantizer (VQ) $[u_{VQ,1} \quad u_{VQ,2} \quad u_{VQ,3}]$. These 3 state bits are used to create 2 bits of the codewords using the same generator polynomials as the ones used at the transmitter side $(05_8, 07_8)$. The accumulator consists of a memory one differential encoder with a stream of bits (1 input bit u_{ACC} at each iteration) as input. The BCJR determines the *a-posteriori* probabilities of the input bits of the accumulator. These *a-posteriori* probabilities are obtained using the *a-priori* L-values at the accumulator. The VQ can be seen as a 3-bit shift register that receives an input bit u_{VQ} every 3 accumulator input bits. This input bit is denoted by "virtual bit" because its values are undetermined, that is, they make the VQ codeword range over all possible values. The VQ codeword $(c_{VQ,1}, c_{VQ,2})$ is obtained from the convolution of the virtual bits u_{VQ} with the polynomials $(g_1(D), g_2(D))$. The decoder that was designed performs the operations in a cycle of 3, that is, in each one of the 3 iterations, it behaves differently. The output is calculated by performing a 4-PAM

mapping of the bits $(b_{1,1}, b_{1,0})$ for m_1 and $(b_{2,1}, b_{2,0})$ for m_2.

4) Initial and terminal states of the decoder: As previously stated, we need the initial and terminal state of the decoder to be the all-zero state. That is, the coded bit from the accumulator $c_{ACC} = 0$ and all 3 vector quantizer state bits $[u_{VQ,1} \quad u_{VQ,2} \quad u_{VQ,3}] = [0 \quad 0 \quad 0]$ should be zero. For the initial state we simply initialize the decoder at the all-zero state. For the terminal state, since only 1 input bit is shifted into the vector quantizer at every 3 iterations, we need the terminating process to start 9 iterations from the end.

5) Computation of the L-values: After the algorithm has iterated over all bits, we are able to compute the *a-priori* probabilities of the accumulator input bits. These probabilities will be converted into the log-domain, resulting in the *a-posteriori* L-values which are defined as:

$$L_{a\text{-}posteriori}(u_{ACC,k}) = \ln \frac{P[u_{ACC,k} = 1]}{P[u_{ACC,k} = -1]} \tag{20}$$

where $P[u_{ACC,k} = 1]$ and $P[u_{ACC,k} = -1]$ are the probabilities of the input bit u_{acc} to be equal to '0' or '1' at time instant k, respectively.

IV. Results

Our simulations concluded with success. The correct sequence of input bits after transmission over a channel with interference was decoded. As expected, the decoder is not perfect because, if the noise is too great, the algorithm is not capable of decoding the correct sequence. Nonetheless, for a reasonable amount of noise, our simulations showed that the dirty-paper technique does work and the received signal can be decoded correctly. Our simulation results will be based on comparing the bit values between the encoder and decoder. First, the original message will be compared with the final decoded message and then, the evolution of our system throughout each iteration will be analyzed.

As introduced previously, our system was simulated for different noise values by modifying the noise coefficient σ between 0 and 1 with a step of 0.1. For each of those noise values, using Equation 21, the corresponding SNR values were obtained which are displayed in Table II.

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 10\log_{10}\left(\frac{P_X}{\sigma^2}\right) \tag{21}$$

Figure 10 displays the number of errors between the original message and the decoded message where the message contains a total of 6000 bits. Each curve represents a whole run for a specific SNR value and the evolution can be tracked by observing the x-axis which indicates the iteration number.

At high SNR, the noise does not distort the signal too much so the received signal doesn't have too many erroneous bits. The amount of information needed to obtain the correct sequence is low so very few iterations of the algorithm are needed. Also, the information acquired in each iteration shows the necessary quality to further improve the decoding process. Slightly reducing the SNR, the noise starts to have a higher effect on the received signal by causing many bits to be received incorrectly. Consequently, many iterations for improving the received signal will need to be performed, each of them consecutively adding useful information to the signal. Last, at extremely low SNR, the noise distorts the signal greatly causing many of the bits to be received incorrectly so many iterations would be needed to acquire the needed information to obtain the correct bit sequence. However, since the noise is too high, the algorithm's accuracy gets compromised. It might perform incorrect decisions about the bit sequence and cause the information acquired to be imprecise. This leads to a propagation of the errors and induces the algorithm into converging to the wrong solution. In this situation, the decoder is unable to decode the message correctly.

TABLE II
SNR VALUES FOR EACH RUN

Run number	σ	SNR(dB)
1	0,1	17,259
2	0,2	11,238
3	0,3	7,717
4	0,4	5,218
5	0,5	3,28
6	0,6	1,696
7	0,7	0,357
8	0,8	-0,803
9	0,9	-1,826
10	1	-2,741

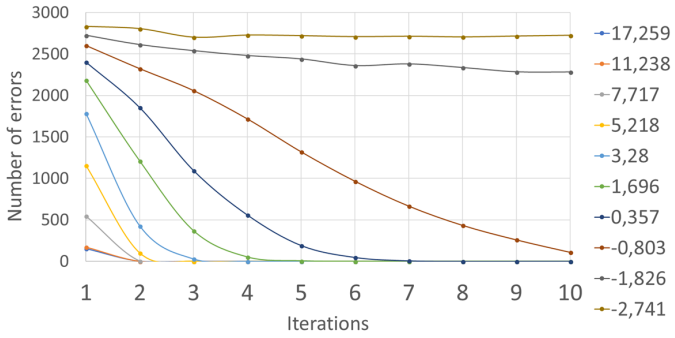

Fig. 10. Evolution of the number of errors along the iterations for each signal-to-noise ratio

The bit error rate (BER) can also be computed for each SNR value in Figure 11 using Equation (22).

$$\text{BER} = \frac{\textit{number of bit errors}}{\textit{total number of bits}} \tag{22}$$

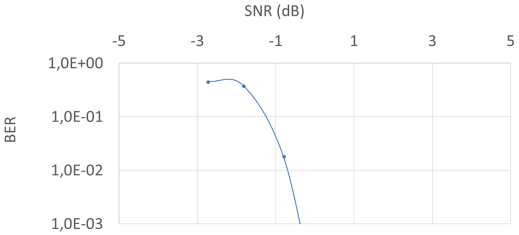

Fig. 11. Evolution of the Bit-Error-Rate (BER) for different Signal-Noise Ratio (SNR) values

Figure 11 displays the BER in a logarithmic scale and, although we dont have many data points in this chart, we detect a sudden drop in the BER curve at around SNR = -1.7 dB. This huge increase in the steepness of the curve is denoted by turbo cliff and defines the SNR values at which the system starts converging to the correct solution.

V. Conclusion

The objective of this thesis was to implement an end-to-end MODEM which employed the Dirty Paper Coding technique. The capability of this system to transmit a message and correctly decode it at the receiver end was tested with different conditions. Overall, the implemented system worked correctly and each component worked as expected by meeting the objectives we had formulated. That is, for relatively low SNR conditions, the system still managed to correctly decode the received signal into the correct original message. From the analysis of the results, we can conclude that the iterative decoding process describes three singular behaviors:

1) Converging into the correct solution after very few iterations;
2) Requiring several more iterations and slowly converging into the correct solution;
3) Converging into the wrong solution.

One of DPC's application is in Multi-User Multiple-Input Multiple-Output (MIMO) systems. These systems allow an increase in the transmission rate by transmitting different streams to different users. However, these users share the same communication medium and cause interference between them. The interference can be reduced by using DPC. It allows the removal of noncausally known interference at the transmitter. Regarding the extent of the implemented system to other situations such as digital watermarking, since the concepts behind both systems are very similar, one can conclude that watermarking can greatly benefit from using side information during the watermark coding process.

For future work, considering the relatively high complexity involved in implementing DPC, the question to ask is: how big of a performance boost does DPC offer against other strategies?

Acknowledgment

I would like to thank my family for supporting me throughout my studies. I would like to thank Prof. Paulo Lopes for the incredible support during the thesis. I would also like to thank my friends for the encouragement throughout the entire course and the thesis.

References

[1] U. Erez and S. T. Brink, *A Close-to-Capacity Dirty Paper Coding Scheme*. In *IEEE Transactions on Information Theory*, pages 3417-3432. IEEE Information Theory Society, 2005.

[2] M. Costa, *Writing on dirty paper*. In *IEEE Transactions on Information Theory*, pages 439-441. IEEE Information Theory Society, 1983.

[3] L. R. Bahl and J. Cocke and F. Jelinek and J. Raviv, *Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate*. In *IEEE Transactions on Information Theory*, pages 284-287. IEEE Information Theory Society, 1974.

[4] Stephan ten Brink and Gerhard Kramer, *Design of Repeat-Accumulate Codes for Iterative Detection and Decoding*. In *IEEE Transactions on Signal Processing*, pages 2764-2772. IEEE IEEE Signal Processing Society, 2003.

[5] M. M. Ingemar Cox and J. Bloom, *Digital Watermarking*. Morgan Kaufmann Publishers, 2006.

[6] J. Hagenauer, E. Offer, and L. Papke, *Iterative decoding of binary block and convolutional codes*, IEEE Trans. Inform. Theory, vol. 42, pp. 429445, Mar. 1996.