UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Vanessa Brischi Olivatto

Analysis of LDPC decoders for DVB-S2 using LLR approximations

Análise de decodificadores LDPC do padrão DVB-S2 utilizando aproximações de LLR

Campinas
2016

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Vanessa Brischi Olivatto

Analysis of LDPC decoders for DVB-S2 using LLR approximations

Análise de decodificadores LDPC do padrão DVB-S2 utilizando aproximações de LLR

Thesis presented to the School of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of Master in Electrical Engineering. Concentration area: Telecommunications and Telematics

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestra em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Supervisor: Prof. Dr. Renato da Rocha Lopes
Co-Supervisor: Dr. Eduardo Rodrigues de Lima

Este exemplar corresponde à versão final da dissertação defendida pela aluna, e orientada pelo Prof. Dr. Renato da Rocha Lopes

Campinas
2016

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

OL4a

Olivatto, Vanessa Brischi, 1991-
Analysis of LDPC decoders for DVB-S2 using LLR approximations / Vanessa Brischi Olivatto. – Campinas, SP : [s.n.], 2016.

Orientador: Renato da Rocha Lopes.
Coorientador: Eduardo Rodrigues de Lima.
Dissertação (Mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Códigos corretores de erros (Teoria da Informação). 2. Teoria da codificação. 3. Modulação digital. 4. Televisão digital. I. Lopes, Renato da Rocha,1972-. II. Lima, Eduardo Rodrigues de. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Análise de decodificadores LDPC do padrão DVB-S2 utilizando aproximações de LLR
Palavras-chave em inglês:
Error-correcting codes (Information Theory)
Coding theory
Digital modulation
Digital television
Área de concentração: Telecomunicações e Telemática
Titulação: Mestra em Engenharia Elétrica
Banca Examinadora:
Renato da Rocha Lopes [Orientador]
Cristiano Magalhães Panazio
Michel Daoud Yacoub
Data da defesa: 01-07-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidata: Vanessa Brischi Olivatto RA: 096002
Data da Defesa: 1º de julho de 2016
Título da Tese: "Analysis of LDPC decoders for DVB-S2 using LLR approximations"

Prof. Dr. Renato da Rocha Lopes (Presidente, FEEC/UNICAMP)
Prof. Dr. Cristiano Magalhães Panazio (EPUSP/USP)
Prof. Dr. Michel Daoud Yacoub (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica da aluna.

To André Fioravanti, with
all my love and admiration.

Acknowledgments

First, I would like to express my gratitude to my supervisor, Prof. Dr. Renato Lopes, who gave me a lot of helpful advices along this work. His support was essential for the conclusion of my work.

I would also like to thanks all my colleagues from Instituto de Pesquisas Eldorado. A special thanks to Dr. Eduardo R. de Lima from DHW, for his valuable guidance and for his compelling willingness to work.

My thanks to my family who always inspires me showing the way of kindness and generosity. To my parents, my brother, my grandmothers, my grandfather and my uncles, for their engaging trajectory of strength and perseverance.

My thanks to all my teachers from School of Electrical and Computer Engineering, above all, Prof. Romis Attux, who was the first to introduce me into the research world. I appreciate all the skills they passed me along these years and sincerely thank them all.

Finally, I am deeply thankful to my beloved boyfriend, André, for his constant patience and support, for his endless love and encouragement. More than anyone, he always know how to drive me forward when I need to be driven. Without him, this work would certainly not have been finally concluded. For all he has been for me, I totally dedicate this thesis to him.

No one wants to learn by mistakes, but we cannot learn enough from successes to go beyond the state of the art.

Henry Petroski

Abstract

As one of the most powerful error-correcting codes, Low-Density Parity Check (LDPC) codes are widely used in many digital communication systems. The encoding scheme of the Digital Video Broadcast Satellite, Second Generation (DVB-S2) systems is based on the concatenated LDPC code and the Bose-Chaudhuri-Hocquenghem (BCH) code. Together, they are responsible for delivering excellent performance, coming close to the Shannon limit. The actual complexity of these LDPC decoders depends on the algorithm chosen as well as on several implementation aspects such as the floating-point representation and the estimation of the reliability values from the channel, i.e., the probabilistic arguments that optimise the channel decoder decisions. Here, we are more interested in providing alternative simplified methods for calculating the approximated channel information and evaluating their performances at the output of soft-decision DVB-S2 LDPC decoders.

The DVB-S2 systems require very large LDPC block sizes. This results in a huge number of mathematical operations, decoding delays and the increasing of the required hardware area. Because of this, it becomes necessary to investigate alternative approximations for reducing the number and the complexity of the operations that are involved in each step of the decoding flow. This work specially addresses the simplification of the input parameters for soft-decision LDPC decoding algorithms. Two original proposals for simplified approximation of the channel Log-Likelihood Ratio (LLR) in higher-order modulation are investigated and compared to a consolidated method commonly referred to as Max-Log.

Keywords: Error-correcting codes (Information Theory), Coding Theory, Digital modulation, Digital Television.

Resumo

Um dos códigos corretores de erros mais poderosos, os códigos *Low-Density Parity Check* (LDPC) são amplamente utilizados em diversos sistemas de comunicação digital. O esquema de codificação dos sistemas *Digital Video Broadcast Satellite* da Segunda Geração (DVB-S2) baseia-se em uma estrutura LDPC concatenada à estrutura Bose-Chaudhuri-Hocquenghem (BCH). Juntos, este códigos são responsáveis por um excelente desempenho, aproximando-se do limite de Shannon. A complexidade real desses decodificadores LDPC depende do algoritmo escolhido bem como de vários outros aspectos de implementação, tais como a representação de ponto flutuante e a estimativa dos valores de confiabilidade do canal, isto é, os argumentos probabilísticos que otimizam as decisões do decodificador de canal. Neste trabalho, estamos mais interessados em fornecer métodos simplificados para o cálculo aproximado da informação do canal, avaliando seus respectivos desempenhos na saída dos *soft-decoders* LDPC.

Os sistemas DVB-S2 utilizam blocos de codificação e decodificação LDPC de comprimento longo. Isso resulta em um grande número de operações matemáticas, atrasos de decodificação e no aumento da área de hardware necessária. Por estas razões, torna-se necessário investigar aproximações alternativas que reduzam o número e a complexidade das operações envolvidos em cada etapa do fluxo de descodificação. Este trabalho aborda especialmente a simplificação dos parâmetros de entrada para algoritmos de decodificação LDPC baseados em *soft-decision*. Duas propostas originais de aproximação simplificada das Razões Logarítmicas de Verossimilhança (*Log-Likelihood Ratio*, LLR) em modulações de ordem superior são investigados e comparados com uma simplificação bem consolidada, usualmente denominada *Max-Log*.

Palavras-Chave: Códigos corretores de erros (Teoria da Informação), Teoria da Codificação, Modulação digital, Televisão digital.

List of Figures

List of Tables

Contents

Introduction

Low-Density Parity-Check (LDPC) codes were first discovered by Robert G. Gallager in 1962 in his doctoral dissertation [1]. For almost 30 years, the LDPC codes had not drawn enough attention until MacKay and Neal rediscovered them in the 1990s [2]. Given its excellent decoding capability, the performance of LDPC codes allows the noise threshold to be set very close to the Shannon limit. Moreover, compared to the Turbo Codes, LDPC codes have a higher error correcting performance, better error floor and lower complexity. These advances boosted LDPC codes to be applied in the Digital Video Broadcast Satellite Second Generation (DVB-S2) [3]. In fact, LDPC codes have also been used in many other standards, such as WiMAX (802.16e) and WiFi (802.11$\mathfrak{n}$ and 802.11$\mathfrak{ac}$).

In the particular case of DVB-S2, LDPC is designed for two very long frame lengths: the normal frame length, which contains 64800 bits and the short frame length, which contains 16200 bits. These frame lengths entail a computational effort that requires decoding algorithms someway adapted and simplified in order to reduce their complexity without significant performance loss. In this sense, there are many LDPC decoding algorithms to be exploited. The Belief Propagation is a message passing family of algorithms that operates over the messages associated with edges of bipartite graphs, and recursively updates them through calculations done at the vertices of the graph. In the context of coding-theory, this algorithm, which is also known as the sum-product message passing [4], achieves good performance but still demands high computation complexity. On the other hand, the suboptimal Min-Sum [5] and its variants [6] achieve a slightly worse performance but demand less computational efforts.

This thesis will not straightly focus on the decoding algorithms but instead, on the simplified estimation of their required inputs, i.e., the soft-information from the channel in the form of Log-Likelihood ratios (LLRs). Computing the LLR may also demand high computational cost, especially for higher-order constellations. For this reason, one of the main purposes of this work is to study and describe new and efficient methods for estimation of LLR approximations.

This thesis is organized as follows:

- Chapter 1: the notation is introduced and a small review is presented for the understanding of this work. Selected topics on the theory of the digital communication systems are

discussed and some concepts on the Galois Theory are highlighted. The most important theorems from this theory are presented in order to address the concepts and properties involving the Linear Block Codes such as the BCH and the LDPC codes. At the end of this chapter, an overview on the DVB-S2 standard functional blocks are presented, as well as the general scheme of the proposed and implemented Matlab model.

- Chapter 2: the most important LDPC decoding algorithms are presented. The Belief-Propagation algorithms and the sub-optimal algorithms such as the Min-Sum and its variants are exploited.

- Chapter 3: explains the contribution of this work. Two new approaches for the LLR simplification are introduced. These approaches are based on splitting the DVB-S2 constellations and the design of suitable and optimised areas of decision. We will show how the choice of convenient areas can significantly reduce the number of symbols taken into account for the approximated calculations.

- Chapter 4: this chapter is dedicated to the presentation of results, discussion and comparison of performance and complexity between the proposed methods of simplification.

- Chapter 5: this chapter is about the conclusion and the perspectives at the end of this work.

Chapter 1

Background

This chapter presents an overview of the main foundations required for the understanding of this thesis. First, a brief review on the basic concepts involving digital communication systems is presented. These concepts are essentially found in [7] and [8]. Then we provide a general presentation of the linear block codes. We also present the fundamentals of Galois Theory [9], which is an essential topic for discussing the subjects that come next: the BCH [9] and LDPC [10, 11, 2] codes. The last section from this chapter sketches the basic structure of the DVB-S2 standard [3], as well as some general features from a Matlab implemented model for simulations.

1.1 Digital Communications

In 1948, Claude Shannon developed the first mathematical theory of information [12] which established fundamental boundaries within which the communication systems can operate. Thanks to his contribution, nowadays we can compare the performance of proposed codes by using these limits.

Figure 1.1 depicts a general diagram containing the basic elements in a digital communication system. In this diagram, the source may be either an analog signal, such as video and audio, or a digital signal, such as the signals exchanged between digital devices. The first step is to convert the source of data into a binary sequence through the source encoder. In this step, the data may be compressed by using the entropy of the source as the measure of information. The source encoding aims to somehow reduce the redundancy in the source. In other words, the main idea is to minimise the amount of bits required for representing the source, keeping the fidelity on the original information. The Huffman algorithm [7] is an example of optimal source encoder when the input symbols are independent and identically distributed random variables whose probabilities are dyadic [13], i.e., the probabilities p_i are given by 2^{-u}, where u is a positive integer. The process of data compression involved on the Huffman algorithm is straightly based on the set of probabilities of each source element. Nevertheless, there are enhanced algorithms that can extract more valuable aspects from the particular structure of the source than only the probabilities associated to the alphabet when the source is not dyadic, independent and identically distributed [13].

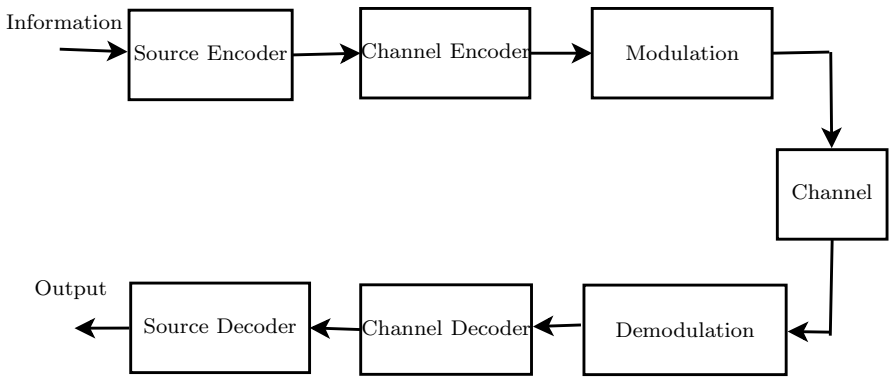

Figure 1.1: Basic elements of a digital communication system.

The channel encoder, on the other hand, aims to introduce some redundancy in the binary sequence in order to overcome the effects of the channel and the noise. In this case, the final goal is to approach the information rate that the channel is able to transmit (the channel capacity) with sufficient reliability, i.e, with the smallest probability of error. The encoding process involves the mapping of k information bits into a new bit sequence of length n that is called codeword. The code rate of a channel code is defined by $r = k/n$.

A reliable communication system is not achievable with information rate R greater than the channel capacity C [13]. The information rate is defined as the product between the code rate and the gross bit rate, i.e., $R = rR_b$. As usual, the gross bit rate is defined as the quotient between the number of bits per symbol, b, and the symbol duration such that $R_b = b/T_r = 1/T_b$. This remarkable result was demonstrated by Shannon in 1948 [12]. In other words, he showed that there will always exist an error-correction code of any rate $r < C/R_b$, that will be able to provide arbitrarily reliable communication or small probability of failure to recover the information bits. It still remains unclear how to find such codes and how to encode and especially decode them in practice with reasonable complexity. However, some modern codes, such as LDPC, offer some promise that Shannon's limits may be achieved.

The digital modulator provides the interface to the communication channel by mapping the binary information into signal waveforms. An M-ary modulation ($M \geqslant 2$) transmits b bits/symbol by using $M = 2^b$ distinct signal waveforms $s_i(t), i = 0,\ 1,\ 2,\ \cdots,\ M-1$.

The channel is the physical medium that is used to transmit the signal, such as cable, optical-fiber and broadcast radio. The digital demodulator estimates the transmitted data symbols based on the channel parameters as well as the channel output. These symbols are demapped to an associated sequence of b bits. This sequence of bits is passed to the channel decoder, which attempts to recover the original bit sequence from the redundancy contained in the received data. Finally, the source decoder decompresses the data to recover the original information provided

by the source. Next section explores the digital demodulator by focusing on the generation and detection of digital modulated signals.

1.2 Optimum Receiver for digitally modulated signal in AWGN channels

In this section, we define the optimum receiver and the discrete-time model of AWGN channels by adopting the quadrature amplitude modulation (QAM) along the particular demonstrations. Through the derivations presented next, we also describe the steps of the digital simulation platform we developed, implementing the basic principles of an optimal receiver. Our main goal is to review the discrete AWGN features as well as the demodulation approach based on the matched filter [14]. In our system, the transmitted signal is corrupted by AWGN, whose two-sided power spectral density S_n is defined as $S_n(f) = \frac{N_0}{2}$. From a received signal $r(t)$ (real and complex signal are both treated in parallel) and a noise signal $n(t)$, we aim to show the structure of an optimum filter which is capable of recovering the transmitted symbols a_k by minimising the probability of error on reception.

Subsection 1.2.1 presents a discrete-time model for Additive White Gaussian Noise. The time and frequency response of the matched filter, which is derived in subsection 1.2.2, is assumed to be known in advance. Thus, the matched filter is denoted as the conjugated time-reversed version of the transmitter pulse shape filter. The main purpose of this subsection is to derive an equivalent AWGN discrete-time model, despite of the continuous-time AWGN models that are intensively discussed in most of references on the digital communications theory [8] [7].

1.2.1 AWGN Discrete-time Model

The Additive White Gaussian Noise model is frequently included in wireless communications and terrestrial channels for representing the effects of general random processes. The AWGN is characterised by the following properties:

- Additive channel: the noise is *added* to the information sent from the transmitter;
- White: the noise has *uniform* power across the system spectrum;
- Gaussian: the noise has a *normal* probability distribution in the time domain.

The noise is characterised as a white Gaussian random process with zero mean and two-sided power spectral density $\frac{N_0}{2}$.

Several important system parameters can be explicitly calculated in terms of discrete-time (digital) parameters of an AWGN. To see this, note that in a continuous-time model with AWGN, as represented in Figure 1.2, the received signal is given by $r(t) = s(t) + n(t)$. We define the SNR, at the output of the receiver, as the ratio between the received signal power P_r and the noise power, limited by the system band, which in turn depends on the channel bandwidth and

spectral properties of $n(t)$. Remind that a baseband channel has the frequency response of a low-pass filter, whereas a passband channel has the frequency response of a bandpass filter [8].

The continuous-time transmitted signal of a PAM baseband system is given by

$$s(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT_r) \tag{1.1}$$

where T_r is the inverse of the symbol rate R, $g(t)$ is the transmitter pulse shape filter and a_k is a real value referred to as a symbol. For arbitrary QAM modulations of order $M > 2$ Equation (1.1) remains the same but in this case we would have $a_k \in \mathbb{C}$. The bandwidth of the complex envelope from baseband modulation will be referred to as B, also corresponding to the bandwidth of $g(t)$. Thus, the corresponding passband signal bandwidth will be referred to as $2B$. Mathematical derivations presented next are related to one single dimension of the symbols (the real or imaginary).

To retrieve the symbols a_k from a continuous-time PAM baseband transmitted signal, $s(t)$, one might sample $s(t)$ at multiples of the symbol period T_r so that

$$s(kT_r) = a_k \star g(kT_r) = \sum_{m=-\infty}^{\infty} a_m g(kT_r - mT_r) \tag{1.2}$$

Splitting the discrete-time convolution into two parts yields

$$s(kT_r) = g(0)a_k + \sum_{m \neq k} a_m g(kT_r - mT_r) \tag{1.3}$$

Equation (1.3) is our first step of transition from continuous to discrete-time model, which is illustrated in Figure 1.3. The first term of this equation is the desired information whereas the second term represents the interference from the adjacent symbols (ISI). Clearly, this last term must be zero so that we can fully eliminate the interference. This constraint imposes that $g(kT_r) = \delta(k)$ so that, in the frequency domain, we must have

$$\frac{1}{T_r} \sum_{m=-\infty}^{\infty} G\left(f - \frac{m}{T_r}\right) = 1 \tag{1.4}$$

Equation (1.4) is the Nyquist criterion in the frequency domain. This result implies the existence of a minimum bandwidth, B, at a certain symbol rate R or, equivalently, a maximum symbol rate for a given bandwidth for an ISI free transmission.

As seen in Equation (1.3), the sampling of $g(t)$ places a replica of $G(f)$ at multiples of R. Hence, in order to ensure the condition of no ISI in (1.3), we must always keep the pulse shape bandwidth less than the half of the symbol rate, i.e, $B \leqslant \frac{1}{2T_r}$. The minimum bandwidth pulse is a sinc function in the time domain. This satisfies the bandwidth constraint as well as Equation (1.3), though the pulse filter design becomes impractical. For this reason, the

bandwidth usually adopted for real applications is larger than its minimum by a factor of $1+\alpha$ so that

$$B \leqslant \frac{1+\alpha}{2T_r} \tag{1.5}$$

The most important overall pulse response that satisfies the Nyquist criterion and additionally may have a non-zero excess of band is called the raised cosine (RC) pulse. For $\alpha = 0$, this pulse is a sinc, in the time domain, and a rectangular window, in the frequency domain. This is the case of the ideally bandlimited pulse. These pulses can be approximated using FIR filters by truncating the pulse at some multiple of T_r. Note that the Nyquist criterion must be satisfied for the overall system response, i.e., $H(f) = G(f)C(f)G(-f)$, which is the frequency response from the transmitter pulse shape filter, the channel and the receiver pulse shape filter, respectively. Since we assume $C(f) = 1$, the transmitter and receiver filters can be obtained by splitting the RC filter response into two parts, so that the entire system response is in accordance to the Nyquist criterion. As a result, the combination of a transmitter pulse shape filter known as root raised cosine (RRC) and a receiver pulse shape filter with the same response yields the expected overall response.

The pulse shaping choice is related to the required spectral characteristics and occupied bandwidth of the baseband modulated signal. There are some well-known family of baseband pulses such as the RRC, the rectangular pulse (unfiltered modulation) and the RC pulse. Their individual responses or their combined responses fulfill the Nyquist criterion. In real systems, the RRC filter is usually preferred due to its configurable excess bandwidth. This feature allows an interesting tradeoff between a more complex filter implementation and the desired system bandwidth efficiency. It is often implemented as a FIR structure, where a more complex implementation stands for an increase in the number of coefficients. Additionally, we note that only the effective response of this filter, (i.e., $g(t) \star g^*(-t)$, not $g(t)$) indeed exhibits zero crossings at the symbol instants.

Figure 1.2 shows a general continuous system configuration scheme. Assuming the transmitter pulse shape filter response $g(t)$ is, in fact, an RRC or any other filter whose combined responses result in an overall system response which is in accordance to the criterion presented in Equation (1.4), we intend to derive the SNR for discrete-time model, at the receiver output. First of all, we introduce the average transmitted symbol energy, which is given by

$$E_s' = E[|a_k|^2] \int_{-\infty}^{\infty} |G(f)|^2 df \tag{1.6}$$

and the average received symbol energy, i.e, the energy obtained from the output of the receivers's pulse shape filter, so that

$$E_s = E[|a_k|^2] \int_{-\infty}^{\infty} \left(|G(f)|^2\right)^2 df \tag{1.7}$$

Equation (1.7) results from the assumption that the signal in the output of matched filter is given by $a_k g(t) \star g^*(-t)$. As we mentioned before, subsection 1.2.2 is dedicated to the analysis

of the matched filter response. On the other hand, the convolution, in the frequency domain, is simply given by $|G(f)|^2$ and the equivalent average energy is $\int_{-\infty}^{\infty} \left(|G(f)|^2\right)^2 df$.

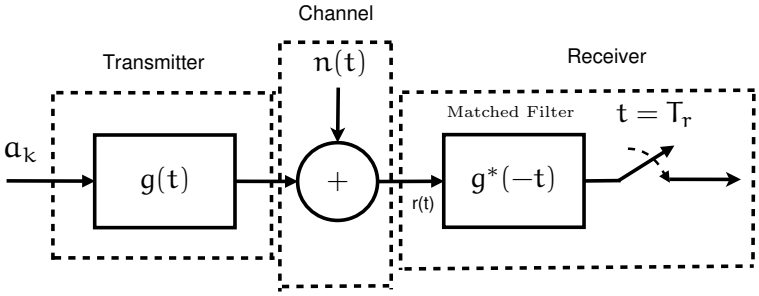

Figure 1.2: General system configuration for continuous-time analysis.

For complex signals, there are two independent noise channels. The variance in each independent dimension (the power of a real signal when the signal mean is zero) is given in terms of N_0, the spectral density (power per unit bandwidth), and B, the bandwidth. In the case of real noise, $\sigma^2 = N = \frac{N_0 B}{2}$. Similarly, in the case of complex noise, the corresponding random variable has variance $2\sigma^2 = N = \frac{N_0 B}{2} + \frac{N_0 B}{2} = N_0 B$.

Typically, for continuous-time signals, we define $SNR = S/N$, where S is the signal power and N is the noise power. For a complex and discrete signal, the SNR might be calculated in terms of the average energy per symbol, E_s and the sample rate, F_s. For M-ary modulations with b bits per symbol without using any channel coding, we assume $E_s = bE_b$

$$\text{SNR} = \frac{S}{N} = \frac{E_s F_s}{N_0 B} = \frac{E_b b F_s}{N_0 B} \tag{1.8}$$

In the case of real signals, we obtain

$$\text{SNR} = \frac{S}{N} = \frac{E_s F_s}{\frac{N_0}{2} B} = \frac{E_b b F_s}{\frac{N_0}{2} B} \tag{1.9}$$

On the other hand, if a channel code with code rate r is included to the system, Equations (1.8) and (1.9) must incorporate the product between the code rate r and the number of bits per symbol of the constellation, $b = \log_2 M$. In these cases, a typical equivalence between the bit and the symbol energy is $E_s = brE_b$.

The SNR is usually adopted as a simple reference measure of channel quality. It compares the level of a desired signal to the level of background noise. But so far, we have only derived its parameters in the continuous-time scenario, i.e., in accordance to our initial goal, it still remains necessary to show how to make the transition to the discrete-time scenario which is represented in Figure 1.3. The continuous-time AWGN model demands the use of signal waveforms when implemented in a digital simulation. By using this model we would certainly increase simulation time since the processing of these waveforms represents a high computational cost. A complete

description of a discrete AWGN model would require the use of the signal space theory. For convenience, in this thesis, we adopt the equivalent discrete-time vectors rather than the waveforms.

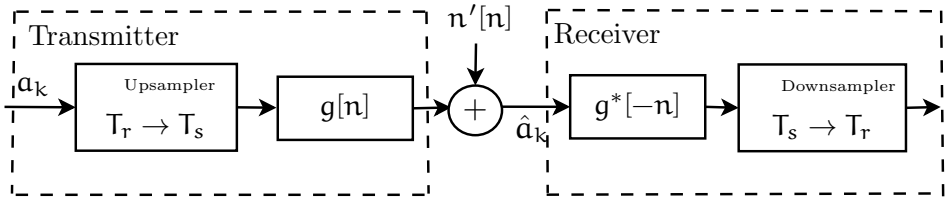

Figure 1.3: General system configuration for discrete-time analysis.

We aim to define $\frac{E_b}{N_0}$ at the output of the discrete-time receiver model. To do so, we start by defining the received signal for one single symbol at $t = 0$ (output of the downsampler) as

$$r(0) = a_k h(0) + n(0) \tag{1.10}$$

where

$$h(0) = g(t) \star g^*(-t)|_{t=0} = \int_{-\infty}^{\infty} |g(t)|^2 dt \tag{1.11}$$

in which $g(t)$ is the transmitter pulse shape filter response, $g^*(-t)$ is the matched filter response and a_k is the transmitted symbol. Similarly, the noise signal at the output of the downsampler is defined as

$$n(0) = n'(t) \star g^*(-t)|_{t=0} \tag{1.12}$$

If the complex signal is discrete, the SNR might be defined in terms of the discrete symbol energy. If energy per complex symbol is E_s and the sample rate is F_s, then the power of the complex signal is $S = E_s F_s$[15]. Furthermore, if we assume a bandwidth B, then the complex noise power is given by $N = N_0 B$. Once we define $F_s = B$, we obtain the following

$$\text{SNR} = \frac{S}{N} = \frac{E_s F_s}{N_0 B} = \frac{E_s}{N_0} \tag{1.13}$$

In the case of real signals

$$\text{SNR} = \frac{E_s}{\frac{N_0}{2}} \tag{1.14}$$

It follows, from a simple analysis of the discrete-time system in Figure 1.3, that the received signal power at the output of the downsampler is given by

$$S = E_s = E[|a_k h(0)|^2] \tag{1.15}$$

as well as the noise power for a real signal is

$$\begin{aligned} E[|n^2(0)|] &= E[|\, n'(t) \star g^*(-t)|_{t=0}\,|^2] && (1.16)\\ &= E\left[\int_{-\infty}^{\infty} |n'(t) g^*(-t)|^2 dt\right] = E\left[\frac{N_0}{2}\int_{-\infty}^{\infty} |g(t)|^2 dt\right] && (1.17)\\ &= E\left[\frac{N_0}{2} h(0)\right] = \frac{N_0}{2} h(0) && (1.18) \end{aligned}$$

Thus we can finally obtain the SNR for a discrete-time system with real signals as

$$\frac{E_s}{N_0} = \frac{E[|a_k|^2]h^2(0)}{\frac{N_0}{2}h(0)} = \frac{2E[|a_k|^2]h(0)}{N_0} \tag{1.19}$$

Note that for the family if systems which do not include any channel encoder, such as the one represented in Figure 1.3, $b = \log_2(M)$ is the final equivalent number of bits per symbol for a constellation of order M. Therefore,

$$\frac{E_b}{N_0} = \frac{E[|a_k|^2]h^2(0)}{b\frac{N_0}{2}h(0)} = \frac{2E[|a_k|^2]h(0)}{bN_0} \tag{1.20}$$

On the other hand, if we include an error-correction code whose code rate is r, we must take into account that the final equivalent number of information bits per symbol of will become br, i.e., $r\log_2 M$. Thus, the variance of the noise we need to generate per dimension in a discrete-time system can be determined from the parameters $\frac{E_s}{N0}$ or $\frac{E_b}{N0}$, the average power of the symbols, and the power of the discrete coefficients from the overall pulse response, $h(t)$, such that

$$\sigma^2 = \frac{N_0}{2} = \frac{E[|a_k|^2]h(0)}{\frac{\frac{E_s}{N_0}}{2}} = \frac{E[|a_k|^2]\frac{1}{L}\sum_{i=1}^{L} g_i^2}{\frac{2E_s}{N_0}} = \frac{E[|a_k|^2]\frac{1}{L}\sum_{i=1}^{L} g_i^2}{2br\frac{E_b}{N_0}} \tag{1.21}$$

where g_i corresponds to the i-th coefficient of the employed transmitter pulse shape filter and L is the total number of filter coefficients. Note that if no channel code block is included, we can simply consider $r = 1$ in Equation (1.21).

1.2.2 Matched Filter

The primary purpose of a receiver is to guess, from the signal $r(t)$, what symbol $a_k \in \mathcal{A} \subset \mathbb{C}$ was originally sent by the transmitter. The minimum distance criterion chooses the alphabet symbol that best suits to the received waveform signal in the sense of minimum distance (ℓ_2 norm) between them. The ℓ_2 norm consists on estimating the energy between the difference signal given by

$$\hat{a}_k = \arg\min_{a_k \in \mathcal{A}} \int_{-\infty}^{\infty} |r(t) - a_k m(t)|^2 dt \tag{1.22}$$

where $m(t)$ is the unknown receiver pulse shape filter.

Deriving the cost function, $J = \int_{-\infty}^{\infty} |r(t) - a_k m(t)|^2 dt$, yields the minimisation of one single term y, at the sample output indicated in Figure 1.5, that is given by $y = \int_{-\infty}^{\infty} r(t)h^*(t)dt$. It follows that y is a *sufficient statistic* to determine the transmitted symbol a_k. This term might be interpreted as the inner product or a simple correlator from which the slicer can take a decision. Figure 1.4 shows the final structure of the correlator.

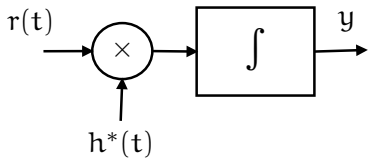

Figure 1.4: Correlator structure.

The Matched Filter (MF) and the correlator are equivalent structures. Nevertheless, the latter presents some disadvantages on its implementation so that, in practice, MF is usually preferred. From now on, we focus on the derivation of the matched filter depicted in the Figure (1.5). According to this scheme, the output of the MF is sampled at each multiple of the symbol period, $t = kT_r$ and then these samples pass through a decision block which selects the most suitable signal. The optimum filter has also another important feature, as the structure responsible for maximising the SNR at the output of the presented discrete-time AWGN model. Our math development is mainly based on this property. We dedicate the next calculations to find the unknown time-response of the filter $m(t)$, from Figure 1.5, that results in this maximisation.

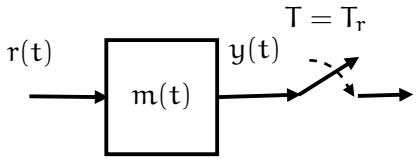

Figure 1.5: The sampled Matched Filter, whose time-response $m(t)$ is being derived in order to maximise the SNR and minimise the probability of error on demodulation.

We start deriving the MF response by taking as an example an hypothetical modulation system for which the probability of error is given by

$$P_e = Q\left(\sqrt{\frac{A}{\sigma_n^2}}\right) \tag{1.23}$$

where $Q(.)$ is the normal cumulative distribution function given by $Q(x) = P(X > x)$ or $Q(x) = 1 - P(X \leqslant x)$, A is the energy of the desired signal at the output of the receiver filter and σ_n^2 is the variance of the noise. In contrast to the probability density function (PDF), $Q(x)$ is a decreasing function of its argument. For a more convenient notation, we refer to the argument of P_e by simply $\frac{A}{\sigma_n^2}$. We note that, in order to minimise P_e, we must maximise its argument.

The output of the filter, $y(t)$, can be written in terms of the time convolution between the

unknown receiver filter response and the received signal as

$$r(t) = a_k g(t) + n(t) \tag{1.24}$$

$$y(t) = \int_0^t s(u)m(t-u)du + \int_0^t n(u)m(t-u)du \tag{1.25}$$

Thus, the argument $\frac{A}{\sigma_n^2}$ might be written as

$$\begin{aligned}
\frac{A}{\sigma_n^2} &= \frac{\left[\int_0^{T_r} s(u)m(t-u)du\right]^2}{E\left[\int_0^{T_r} n(u)m(t-u)du\right]^2} \\
&= \frac{\left[\int_0^{T_r} s(u)m(t-u)du\right]^2}{E\left[\int_0^{T_r} n(u)m(t-u)du \int_0^{T_r} n(v)m(t-v)dv\right]} \\
&= \frac{\left[\int_0^{T_r} s(u)m(t-u)du\right]^2}{\int_0^{T_r}\int_0^{T_r} E\left[n(u)n(v)\right] m(t-u)m(t-v)dudv} \\
&= \frac{\left[\int_0^{T_r} s(u)m(t-u)du\right]^2}{\int_0^{T_r}\int_0^{T_r} \frac{N_0}{2}\delta(u-v)m(t-u)m(t-v)dudv} \\
&= \frac{\left[\int_0^{T_r} s(u)m(t-u)du\right]^2}{\frac{N_0}{2}\int_0^{T_r} m^2(t-u)du}
\end{aligned}$$

By handling the statements above we need to obtain an impulse response $m(t)$ such that the SNR, i.e., $\frac{A}{\sigma_n^2}$, is maximised, as mentioned before. The classical approach is obtained by applying the Cauchy-Schwarz inequality theorem, which is stated in Theorem 1

Theorem 1 (Cauchy-Schwarz Inequality)**.** Let γ and ψ be any two real integrable functions in $[a, b]$. The Cauchy-Schwarz inequality is given by

$$\left[\int \gamma(x)\psi(x)dx\right]^2 \leqslant \int \gamma^2(x)dx \int \psi^2(x)dx$$

The equality is only reached when, for a constant c, we have $\psi(x) = c\gamma(x)$

Hence, the numerator of the SNR derived above may be maximised if $m(t-u) = cs(u)$. Applying this result to the SNR numerator, we obtain

$$\left[\int_0^{T_r} s(u)m(t-u)du\right]^2 = \int_0^{T_r} s^2(u) \int_0^{T_r} c^2 s^2(u)du \tag{1.26}$$

for any constant c.

If $s(t)$ has a finite duration of T_r (one symbol period), then the SNR is maximised by simply setting $t = T_r$. In this scenario, we have

$$\frac{A}{\sigma_n^2} = \frac{\left[c\int_0^{T_r} s^2(u)du\right]^2}{\frac{N_0}{2}\int_0^{T_r} c^2 s^2(u)du} = \frac{\int_0^{T_r} s^2(u)du}{\frac{N_0}{2}} \tag{1.27}$$

Finally, the filter impulse response derived from the Cauchy-Schwarz inequality theorem is given by

$$\begin{aligned} m(t-u) &= cs(u) \\ m(u) &= cs(t-u) \end{aligned}$$

while the sampled MF impulse response is

$$\begin{aligned} m(T_r-u) &= cs(u) \\ m(u) &= cs(T_r-u) \end{aligned}$$

The output of the matched filter is the time convolution between $m(t)$ and $r(t)$. On the other hand, the output after the sampling at $t = T_r$ is given by the following

$$\begin{aligned} y'(T_r) &= \int_0^{T_r} m(u)r(T_r-u)du \\ &= \int_0^{T_r} cs(T_r-u)r(T_r-u)du \\ &= c\int_0^{T_r} s(u)r(u)du \\ &= c\langle s(u), r(u)\rangle \end{aligned}$$

For a complex signal, the conjugate is required so that the impulse response of the MF has the conjugate and the time flipping response of $s(t)$. We can ideally interpret the signal $s(t)$ as the output of the transmitter pulse shape filter $g(t)$ such that $s(t) = a_k \star g(t)$. In this case, the MF response would be $g^*(-t)$ while the overall response of the system would be $g(t) \star g^*(-t)$ following the permanent assumption that channel response $C(f) = 1, \forall f$.

Figure 1.6 illustrates the theoretical aspects covered in the last two subsections. This figure results from a discrete-time simulation involving the implementation of a digital MF filter and the generation of an AWGN in a 4-QAM baseband system. Note that the M-QAM systems can be viewed as two independent PAM systems in quadrature to each self. For example, the 4-QAM system mentioned before may be treated as two 2-PAM independent systems.

1.3 Galois Fields

The general theory of finite fields began with the work of Carl Friedrich Gauss (1777-1855) and Evariste Galois (1811-1832) [16]. Nevertheless, it only became of interest for engineers in recent decades because of its many applications in communication theory. This section introduces

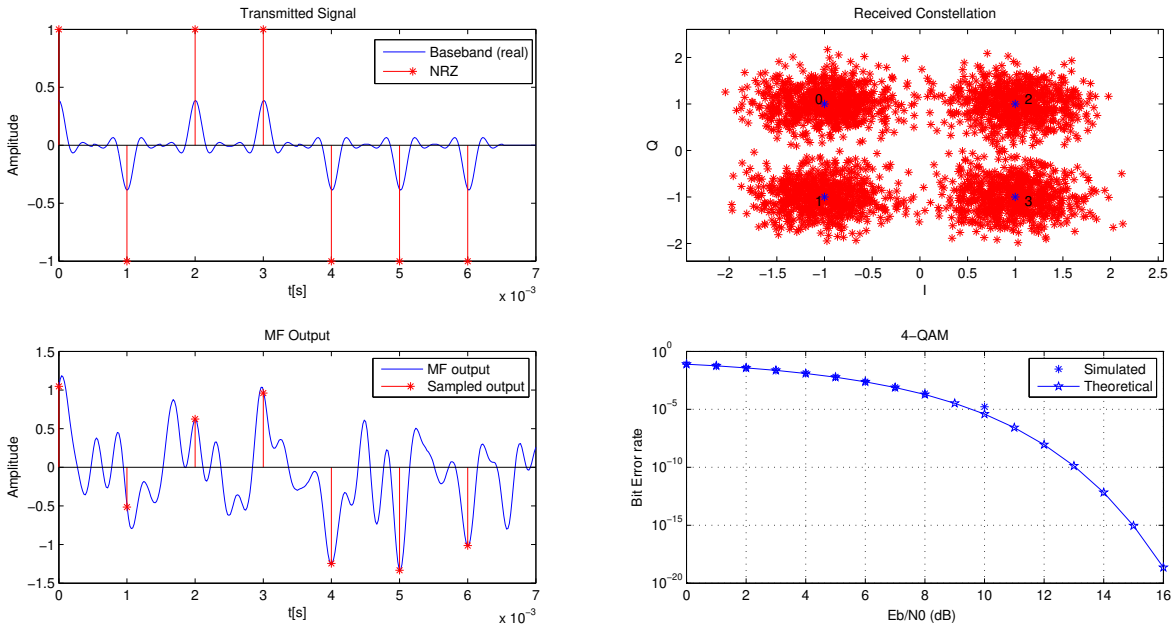

Figure 1.6: Simulation results for a 4-QAM baseband system: From left to right, from top to bottom, (a) NRZ and baseband signal at the output of the transmitter pulse shape filter (RRC); (b) Received constellation at the output of the discrete-time AWGN model with $\frac{E_b}{N_0} = 6$dB; (c) MF Output and sampled values; (d) Theoretical and Simulated bit-error rate.

the key definitions and properties from the general theory of finite fields. These mathematical concepts will be relevant for the next chapters, where we will present the *Bose-Chaudhuri-Hocquenghem* (BCH) and the LDPC codes. We start by the fundamental definitions of Groups, Rings and Fields.

The number of elements in a set is called *cardinality* and this parameter can be either finite or infinite. In the context of algebraic theory, two operations, heretofore undefined, can be applied over the elements of a given set: the multiplication and the addition. Depending on the conditions imposed over this set, we can classify it into some well-defined algebraic structures as follows:

- **Group**: A group is a nonempty set for which only the multiplicative operations are defined. Multiplying any two elements in a group must result in a third element that also belongs to the same group. This property is known as *closure*. Furthermore, a group must satisfy the following conditions:

  1. Associativity under multiplication: For any elements $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ in the group, it follows that $(\mathfrak{a} \times \mathfrak{b}) \times \mathfrak{c} = \mathfrak{a} \times (\mathfrak{b} \times \mathfrak{c})$

2. Identity under multiplication: For any element $\mathfrak{a}$ in the group, there must exist an element $\mathfrak{b}$, such that: $\mathfrak{a} \times \mathfrak{b} = \mathfrak{a}$
3. Inverse: For any element $\mathfrak{a}$ in the group, there must exist an inverse element $\mathfrak{a}^{-1}$ such that: $\mathfrak{a} \times \mathfrak{a}^{-1} = \mathfrak{b}$, where $\mathfrak{b}$ is the identity element.

The group may also present the commutativity under multiplication i.e., $\mathfrak{a} \times \mathfrak{b} = \mathfrak{b} \times \mathfrak{a}$. In this case, the set is called *Abelian group* or *Commutative group*

- **Rings**: Rings are sets for which the operations of addition and multiplication are both defined. A ring must also satisfy the following three sets of properties:

  1. A ring must be an *abelian* group under addition, meaning that
     - Associativity under addition: For any elements $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ in the ring, it follows that $(\mathfrak{a} + \mathfrak{b}) + \mathfrak{c} = \mathfrak{a} + (\mathfrak{b} + \mathfrak{c})$.
     - Commutativity under addition: For any two elements $\mathfrak{a}, \mathfrak{b}$ in the ring, it follows that $(\mathfrak{a} + \mathfrak{b}) = (\mathfrak{b} + \mathfrak{a})$.
     - Additive identity: There must exist an element 0 in the set such that $\mathfrak{a} + 0 = \mathfrak{a}$ for all $\mathfrak{a}$ in the set.
     - Additive inverse: For each element $\mathfrak{a}$ in the set there must exists a second element $-\mathfrak{a}$ in the set such that $\mathfrak{a} + (-\mathfrak{a}) = 0$
  2. A ring must satisfy the following property under multiplication:
     - Associativity under multiplication: For any elements $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ in the ring, it follows that $(\mathfrak{a} \times \mathfrak{b}) \times \mathfrak{c} = \mathfrak{a} \times (\mathfrak{b} \times \mathfrak{c})$. A ring satisfying this property is sometimes denoted as an associative ring.
  3. A ring must satisfy the distributivity under multiplication, i.e.:
     - Distributivity under multiplication with respect to addition: For any elements $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ in the ring, it follows that $\mathfrak{a} \times (\mathfrak{b} + \mathfrak{c}) = (\mathfrak{a} \times \mathfrak{b}) + (\mathfrak{a} \times \mathfrak{c})$ and $(\mathfrak{b} + \mathfrak{c}) \times \mathfrak{a} = (\mathfrak{b} \times \mathfrak{a}) + (\mathfrak{c} \times \mathfrak{a})$.

  A ring may also present the property of commutativity under multiplication i.e., $\mathfrak{a} \times \mathfrak{b} = \mathfrak{b} \times \mathfrak{a}$. In this case, the set will be called *Commutative Ring*. Additionally, if a ring presents a multiplicative identity, then this set will be called *Ring with Identity*. Nevertheless, some authors depart from the presented properties and define, under their definition, additional properties. For example, in [17] the multiplicative identity is added as a necessary condition.

Example 1.3.1: Ring of integers

An example of ring is the ring of integers, $\mathbb{Z}_6$, defined by the set $\{0, 1, 2, 3, 4, 5\}$ under modulo-6 operations. It is often convenient to describe these algebraic structures in terms of an addition and a multiplication table. Such tables are called Cayley tables. In our example we include both tables.

Through the tables presented below, we can easily note that $\{\mathbb{Z}_6, +, \times\}$ obeys the three basic conditions of a ring. This set also satisfies the properties of commutativity under addition and multiplication. Besides, the identity element of this ring is 1, meaning that $\mathbb{Z}_6$ shall be classified as a Commutativity-Identity Ring.

$+$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Table of addition

$\times$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Table of multiplication

We remark that $\{\mathbb{Z}_6, \times\}$ can not be a *group* since the element 0 has no inverse element. The set of elements $\{2, 3, 4\}$ are called *zero divisor* in $\mathbb{Z}_6$ because they also do not present inverse elements. On the other hand, the elements 1 and a 5 have inverses 1 and 5, respectively.

- **Fields**: As well as the algebraic structure of the rings, the fields are also defined for operations of addition and multiplication. This structure also satisfies the following conditions, in addition to those of the rings:

  1. Commutativity under addition: for any two elements $\mathfrak{a}$ and $\mathfrak{b}$, $\mathfrak{a} + \mathfrak{b} = \mathfrak{b} + \mathfrak{a}$.
  2. Commutativity under multiplication for all elements but 0: for any two non-zero elements $\mathfrak{a}$ and $\mathfrak{b}$, $\mathfrak{a} \times \mathfrak{b} = \mathfrak{b} \times \mathfrak{a}$
  3. Distributivity: for any three elements $\mathfrak{a}$, $\mathfrak{b}$ and $\mathfrak{c}$ of the set, it follows that $\mathfrak{a} \times (\mathfrak{b} + \mathfrak{c}) = (\mathfrak{a} \times \mathfrak{b}) + (\mathfrak{a} \times \mathfrak{c})$.
  4. Identity: for any element $\mathfrak{a}$ in the set, there is an multiplicative identity element $\mathfrak{b}$, such that $\mathfrak{a} \times \mathfrak{b} = \mathfrak{a}$ and also an additive identity element $\mathfrak{c}$, such that $\mathfrak{a} + \mathfrak{c} = \mathfrak{a}$. The additive identity and the multiplicative identity are required to be distinct, i.e., $\mathfrak{c} \neq \mathfrak{b}$.

5. Inverse: for any element a in the field, the element a^{-1} must be in the set such that $\mathsf{a} \times \mathsf{a}^{-1} = \mathsf{b}$, where b is the multiplicative identity element defined before.
6. Commutativity under multiplication: when the additive identity element 0 is removed, the set must be commutative so that for any two elements a, b in the field, we have $\mathsf{a} \times \mathsf{b} = \mathsf{b} \times \mathsf{a}$.

Some set of integers can not be classified as a field because they include integers which do not present a multiplicative inverse, as illustrated in the previous example.

A *finite field*, which is also known as a *Galois Field*, is commonly represented as $\mathsf{GF}(\mathsf{q})$. It can be shown that, for a finite set to be a field, the parameter q, which represents the cardinality, can only be a prime number or even a power of a prime number greater than 1 [9]. It can also be shown that, to be a finite field, the elements $\{1,\ 2,\ 3,\ \cdots,\ \mathsf{q}-1\}$ must form a group under multiplication modulo-q. For instance, the integers $\{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\}$ form a field in $\mathsf{GF}(7)$ with the operations of sum and product defined as modulo-7, and indeed, also form a group under the product modulo-7. On the other hand, the set of integers from zero to nine is not an example of a finite field since the subset $\{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\}$ is classified as group over the multiplication modulo-10. For instance, there are some non-zero elements in the set $\{0,\ 2,\ \cdots,\ 9\}$ whose multiplication results in 0, such as 5×6.

We call $\mathsf{GF}(\mathsf{p}^{\mathsf{m}})$ an extension field of $\mathsf{GF}(\mathsf{p})$ for any positive integer $\mathsf{m} \geqslant 1$ where $\mathsf{p}^{\mathsf{m}} = \mathsf{q}$. Additions and multiplications between the elements in the field $\mathsf{GF}(\mathsf{p})$ from the base of these fields are defined modulo-p. The rest of the elements have these operations defined from a *primitive polynomial*, that we will define next.

Every element in a finite field has a parameter called *order*. The order of a given element β is the number of times it must be multiplied by itself until we obtain the element 1, i.e., the order, o, is such that $\beta^{\mathsf{o}} = 1$. The elements which present their order equal to $\mathsf{q} - 1$ are known as the *primitive elements*. For instance, in $\mathsf{GF}(3)$ the element 2 is a primitive element because all the non-zero elements in this finite field can be written in terms of the powers of 2, as presented in the Table 1.1. We note that every finite field contains at least one primitive element.

A key property involving the *primitive elements* stems from the fact that all the non-zero elements in a finite field can be expressed in terms of the powers of the primitive elements.

All finite fields of order given by a prime number p have the same properties to the field $\{0,\ 1,\ 2,\ \cdots,\ \mathsf{p}-1\}$ with modulo-p addition and multiplication. All the non-zero elements from these fields have order p. The finite fields of order p^{m} are called extension fields, and their elements can also be represented in terms of primitive elements. In other words, if α is a primitive element, then the full set is

$$\{0\ 1\ \alpha\ \alpha^2\ \cdots\ \alpha^{\mathsf{p}^{\mathsf{m}}-2}\} \tag{1.28}$$

Table 1.1: Representation of the non-zero elements in the finite field $GF(3)$ in terms of its primitive element.

Element	**Representation (Primitive Element)**
1	2^2
2	2^1

The elements in $GF(p^m)$ shall be represented as an m-tuple vector in terms of the elements from $GF(p)$. In this case, addition and multiplication are called ordinary addition and multiplication modulo-p, respectively. The addition of any two m-tuple is obtained by the element-wise addition in $GF(p)$.

To better represent the extension field $GF(p^m)$, we need to introduce two important instances from the field theory: the irreducible and the primitive polynomials. A polynomial $f(x)$ is said to be irreducible if it has a degree m with coefficients defined over $GF(p)$, and it can not be factored into a product of polynomials of lower degree. In other words, $f(x)$ can not have any root in the set $GF(p), \{0,\ 1,\ 2,\ \cdots,\ p-1\}$. On the other hand, an irreducible polynomial $g(x)$ of degree m defined over a base field $GF(p)$ is said to be primitive if the smallest integer n for which $g(x)$ divides $x^n - 1$ is $n = p^m - 1$. A primitive polynomial is a polynomial that is able to generate all the elements of a given extension field from its base field [18].

Considering a polynomial with coefficients in $GF(p)$, the addition of any two polynomials corresponds to the usual polynomial addition in $GF(p)$. On the other hand, the scalar multiplication of this polynomial by a field element of $GF(p)$ corresponds to the multiplication of each polynomial coefficient by the field element, carried out in $GF(p)$.

Theorem 2. The roots of an m-th degree primitive polynomial, $p(x) \in GF(p)[x]$, are primitive elements in $GF(p^m)$, i.e., any root can be used in order to generate the non-zero elements of the extension field $GF(p^m)$.

For further clarification, we will present an example by building up an extension field from a base field $GF(2)$. In order to completely describe and define the extension field $GF(2^5)$, we need to obtain their irreducible and at least one of its primitive polynomial. The irreducible polynomials are obtained by choosing those polynomials which have no roots over the set $GF(2)$. For didactic purposes, as a first step, we might simply select those polynomials which contemplate this constraint from an entire list containing all the possible polynomials with degree m and coefficients over the base field $GF(2)$.

In our case, we must evaluate the elements $\{0, 1\}$ by replacing them into the list of each possible polynomial of degree 5 and apply the operations in $GF(2)$. If the evaluation results in zero for any element of the set $\{0, 1\}$, then that polynomial can not be classified as an irreducible

polynomial since it could be factored in terms of these *root elements*. After testing the full list of polynomials of degree 5, we obtain the irreducible polynomials list which is given by Table 1.2.

Table 1.2: Irreducible polynomials over $\mathsf{GF}(2^5)$.

x^5	x^4	x^3	x^2	x	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1

In order to obtain the primitive polynomials based on the list of irreducible's, we can explore the aspect of the representation of each element of $\mathsf{GF}(p^m)$. For instance, we can take the first irreducible polynomial from the Table 1.2, i.e., $x^5 + x + 1$. If we assume this is a primitive polynomial, the roots would be expressed in terms of a generic primitive element, α, so that $\alpha^5 = \alpha + 1$. This would lead to an specific representation of the field elements as shown in the Table 1.3

Table 1.3: Representation of the field $\mathsf{GF}(2^5)$

Elements in $\mathsf{GF}(2^5)$	**Representation in terms of lower power of** α	m**-tuple vector over** $\mathsf{GF}(2)$
0	0	00000
1	1	00001
α	α	00010
α^2	α^2	00100
α^3	α^3	01000
α^4	α^4	10000
α^5	$\alpha + 1$	00011
α^6	$\alpha^2 + \alpha$	00110
α^7	$\alpha^3 + \alpha^2$	01100
α^8	$\alpha^4 + \alpha^3$	11000
α^9	$\alpha^4 + \alpha + 1$	10011
α^{10}	$\alpha^2 + 1$	00101
α^{11}	$\alpha^3 + \alpha$	01010
α^{12}	$\alpha^4 + \alpha^2$	10100
α^{13}	$\alpha^3 + \alpha + 1$	01011
α^{14}	$\alpha^4 + \alpha^2 + \alpha$	10110
α^{15}	$\alpha^2 + 1$	00101
α^{16}	$\alpha^3 + \alpha$	01010
$\vdots$	$\vdots$	$\vdots$

As we can note in Table 1.3, this polynomial is not able to define all the elements of our extension field since the representation of the elements within the set starts to repeat for distinct elements. This repetitive behaviour is a sufficient proof that the polynomial under evaluation is not primitive. The key point in this toy-example refers to the straightforward observation that the element $\alpha^{2^5-2} \times \alpha$, which is expected to result in $1 \in \mathsf{GF}(2^5)$ since the non-zero elements of a finite field must form a cyclic group, in this case will not (check (1.28)). This observation might be useful every time we need to identify the primitive polynomials from a list of irreducible polynomials. Therefore, as a first verification, we can test primitive polynomials for any extension field by simply selecting those polynomials which lead to the element 1 when we evaluate $\alpha^{p^m-2} \times \alpha$. Note that this verification is necessary but not sufficient to determine if polynomial is actually primitive. Nevertheless, through this observation, we are able easily check the primitive polynomials in $\mathsf{GF}(2^5)$, for instance. The coefficients of these polynomials are listed in the Table 1.4 for illustrative purposes.

Table 1.4: Primitive polynomials over $\mathsf{GF}(2^5)$.

x^5	x^4	x^3	x^2	x	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1

The main motivation of using finite fields to construct error correcting codes is that we can prove several important and practical properties of the resulting codes, such as their error-correcting capabilities. Furthermore, the parallel with finite field theory allows the implementation of very efficient decoding algorithms. One particularly important class of codes that arise from finite field theory is the BCH code, which will be discussed in the subsection 1.4.1. Proceeding in the same illustrative case, we can adopt the first primitive polynomial from the Table 1.4 in order to introduce an example involving the definition of the *minimal polynomial* and the *cyclotomic cosets*. For any element $\beta \in \mathsf{GF}(2^m)$, we define its conjugates by β^{2^i}, $i = 1,\ 2,\ 3,\ \cdots,\ (m-1)$. Henceforth, we will keep $p = 2$ to become easier the association with the binary information. For any element $\beta \in \mathsf{GF}(2^m)$, the minimal polynomial of β is defined as

$$M(x) = (x + \beta)(x + \beta^2)(x + \beta^4) \cdots (x + \beta^{2^{(m-1)}}) \tag{1.29}$$

The conjugates of a given finite field $\mathsf{GF}(q)$ compose a set known as *cyclotomic coset*. The cyclotomic cosets and its associated minimal polynomial of $\mathsf{GF}(2^5)$ are shown in the Table 1.5.

Table 1.5: Cyclotomic cosets over $GF(2^5)$.

Finite field elements	**Minimal Polynomial**
0	x
1	$x+1$
$\{\alpha,\ \alpha^2,\ \alpha^4,\ \alpha^8,\ \alpha^{16}\}$	x^5+x^2+1
$\{\alpha^3,\ \alpha^6,\ \alpha^{12},\ \alpha^{17},\ \alpha^{24}\}$	$x^5+x^4+x^3+x^2+1$
$\{\alpha^5,\ \alpha^9,\ \alpha^{10},\ \alpha^{18},\ \alpha^{20}\}$	$x^5+x^4+x^2+x+1$
$\{\alpha^7,\ \alpha^{14},\ \alpha^{19},\ \alpha^{25},\ \alpha^{28}\}$	$x^5+x^3+x^2+x+1$
$\{\alpha^{11},\ \alpha^{13},\ \alpha^{21},\ \alpha^{22},\ \alpha^{26}\}$	$x^5+x^4+x^3+x+1$
$\{\alpha^{15},\ \alpha^{23},\ \alpha^{27},\ \alpha^{29},\ \alpha^{30}\}$	x^5+x^3+1

Being $\beta \in GF(2^m)$ a root of the polynomial $g(x) \in GF(2)[x]$, i.e., the polynomial which has coefficients in the base field $GF(2)$ and the roots from the extension field $GF(2^m)$, then β^{2^n} is also a root of $g(x)$. Hence, the conjugates of β with respect to a subfield $GF(q)$, $\beta,\ \beta^2,\ \beta^{2^2},\ \beta^{2^3},\ \cdots$, also form a class of roots from $g(x)$. For convenience, we state this well-known result by the following theorem:

Theorem 3. If $\beta \in GF(p^m)$ is a root of $g(x) \in GF(p)[x]$, then β^{p^n} is also a root of $g(x)$. The conjugates of β with respect to $GF(p)$ form a set called the *conjugacy class* of β.

The proof of this theorem can be found in [18]. In the next section, we discuss an important class of error-correcting codes, the linear block codes, and show how the concepts of finite field theory introduced in this section can be used to construct an important family of the linear block codes.

1.4 Linear Block Codes

In this section we will present the most relevant properties from the linear block codes. In practice, this is the most important class of codes and includes LDPC, BCH and also Reed Solomon codes.

A block code is a code in which k bits of information are encoded into n bits that are called *codeword*. These codes are referred to as an (n, k) code. Assuming a binary message in $GF(2)$, for each k bits in the input of an encoder, there will exist 2^k different possible messages. Historically, these codes were intensively studied in the specific case of Binary Symmetric Channels (BSC). The BSC channel is defined from a probability p of bit in error, which corresponds to the power noise density $\frac{N_0}{2}$ and a BPSK modulator, whose symbol energy is E_s. Therefore, the crossover probability p becomes $p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right)$, which in turn corresponds to the error probability of the equivalent AWGN channel. The BSC reduces the possible values from the channel output to only two symbols and due to this, the BSC is a hard-decision model for the underlying of AWGN [19].

Definition 1.4.1. A block code with 2^k codewords of length n is a linear (n, k) code if and only if the collection of codewords form a k-dimensional subspace of the vector space of the n-tuples over the field $GF(2)$. More generally, for a field with q elements, a block code $\mathcal{C}$ of length n with q^k codewords is called a linear (n, k) code if, and only if, its q^k codewords form a k-dimensional subspace in the vector space of all n-tuples over the field $GF(q)$.

Corollary 3.1. The sum of any two codewords in the block code $\mathcal{C}$ will also result in a new codeword of $\mathcal{C}$.

The characteristic of linear vector space also ensures that there will always exist a vector base from which all the codewords can be represented as linear combinations of these vectors. In this sense, the encoding operation requires a simple matrix multiplication. We shall introduce a *generator* matrix $\mathbf{G}$ from the linearly independent *base vectors* $\{\mathbf{g}_1\ \mathbf{g}_2\ \cdots\ \mathbf{g}_k\}$ of dimension $1 \times n$ given by

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ . \\ . \\ . \\ \mathbf{g}_k \end{pmatrix} \tag{1.30}$$

where $\mathbf{G}$ is a $k \times n$ matrix. If the message we expect to encode is $\mathbf{u} = (u_1\ u_2\ ...\ u_k)$, the associated codeword of length n is given by $\mathbf{c} = \mathbf{uG} = u_1 g_1 + u_2 g_2 + ... + u_k g_k$.

We remark that performing elementary row operations in $\mathbf{G}$ does not change the row span and thus, the same code is produced. If two columns of $\mathbf{G}$ are interchanged, the corresponding codewords indices are interchanged the same way. Nevertheless, the distance between the codewords from this new structured code is kept, as stated in Theorem 4, which underlies the equivalent codes.

Theorem 4. Two $k \times n$ generator matrices generate equivalent codes if one can be obtained from the other by a sequence of operations of the following types:

1. Permutation of rows
2. Multiplication of a row by a non-zero scalar
3. Addition of a scaled multiple of one row to another
4. Permutation of columns
5. Multiplication of any fixed column by a non-zero scalar

Proof. The items 1, 2 and 3 preserve the linear independence of the rows of $\mathbf{G}$ by simply replacing one basis by another from the original code. In fact, item 1 merely reorders the basis of $\mathcal{C}$. Items 4 and 5 are both permutations of the code, i.e., it still keeps the same structure and preserves

the minimum distance from the original code. Indeed, the set of codewords remains the same as before the matrix operations. Nevertheless, these operations will result in a new mapping between a given message, u, and its corresponding codeword. We note that equivalent codes will also keep its original performance. □

The generator matrix $\mathbf{G}$ of a linear block code is also related with a parity check matrix $\mathbf{H}$, whose rows are orthogonal to every codeword, i.e.,

$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \tag{1.31}$$

This implies that

$$\mathbf{G}\mathbf{H}^T = \mathbf{0} \tag{1.32}$$

It is often convenient to have the original data explicitly embedded in the final generated codeword. This sort of structure can be seen in what we call *systematic* code. The codewords from this sort of codewords present the following sequence

$$\mathbf{c} = \mathbf{u}\mathbf{G} = \left(p_1\ p_2\ ...\ p_{(n-k-1)}\ u_1\ u_2\ ...\ u_k\right) \tag{1.33}$$

where $\mathbf{u}$ is the set of bits from the message to be encoded and $\mathbf{p}$ is the incorporated set of parity check bits.

For a systematic code, it is always possible to obtain a convenient matrix $\mathbf{G}$ for which it becomes easy to obtain the parity check matrix $\mathbf{H}$. In fact, performing row operations and column reordering on $\mathbf{G}$, we can write it equivalently as

$$\mathbf{G} = \begin{pmatrix} \mathbf{P} & \mathbf{I_k} \end{pmatrix} \tag{1.34}$$

where $\mathbf{I_k}$ is an identity matrix of dimension $k \times k$ and $\mathbf{P}$ is a matrix of dimension $k \times n-k$ which distinguishes one code from another of the same dimension, i.e., it represents the identity of the code. As a straightforward consequence, obtaining the parity check matrix for systematic codes is simply write

$$\mathbf{H} = \begin{pmatrix} \mathbf{I_{n-k}} & -\mathbf{P}^T \end{pmatrix} \tag{1.35}$$

Note that, due to the nature of binary fields, $-\mathbf{P} = \mathbf{P}$ for fields of the form $GF(2^m)$.

The syndrome vector of a linear block code is given by the product between a binary received vector $\mathbf{x}'$ and the parity check matrix $\mathbf{H}$, i.e.,

$$\mathbf{s} = \mathbf{x}'\mathbf{H}^T \tag{1.36}$$

For all the codewords which belong to $\mathcal{C}$, the syndrome is equal to a zero vector. This way, it is possible to determine if any received codeword is a valid codeword $\mathbf{c} \in \mathcal{C}$. The Example (1.4.1) illustrates the structure of a systematic code.

Example 1.4.1: Hamming Code

A Hamming code $(7, 4)$ is generated by the following matrix

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.37}$$

For a message sequence $\mathbf{u} = (u_1\ u_2\ u_3\ u_4)$, the codeword is

$$\mathbf{c} = (p_1\ p_2\ p_3\ u_1\ u_2\ u_3\ u_4) \tag{1.38}$$

where p_i corresponds to the i-th parity check bit given by

$$p_1 = u_2 \oplus u_3 \oplus u_4 \tag{1.39}$$
$$p_2 = u_1 \oplus u_3 \oplus u_4 \tag{1.40}$$
$$p_3 = u_1 \oplus u_2 \oplus u_4 \tag{1.41}$$
$$\tag{1.42}$$

Since we know this is a systematic code, the parity check matrix, $\mathbf{H}$, which is orthogonal to the generator matrix $\mathbf{G}$, might be easily obtained from the observation of $\mathbf{G}$ and identification of $\mathbf{P}$ from Equation (1.34). Thus, the parity check matrix can be reordered according to Equation (1.35) so that we obtain

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{1.43}$$

Theorem 5. For a linear block code $\mathcal{C}$, the minimum hamming distance d_{min} among their codewords, $c_i,\ i \in \{0,\ 2,\ \cdots,\ 2^k - 1\}$, is equal to the minimum weight w_{min} of its non-zero codeword [18]. The weight of a codeword is given by the number of non-zero elements.

Proof. This result relies on the fact that for any linear combination of codewords from $\mathcal{C}$, the resulting codeword still resides in the same set of codewords from $\mathcal{C}$. The hamming distance calculation of any linear combination $a_1 c_i + a_2 c_j,\ i,\ j \in \{0,\ 1,\ \cdots,\ 2^k - 1\}$ might be reassigned to the origin, i.e.,

$$d_{min} = \min_{c_i, c_j \in \mathcal{C}, c_i \neq c_j} d_H(c_i, c_j) = \min_{c_i, c_j \in \mathcal{C}, c_i \neq c_j} d_H(a_1 c_i + a_2 c_j) = \min_{c \in \mathcal{C}, c \neq 0} w(c) \tag{1.44}$$

□

The minimum distance, d_{min}, is sometimes included in the code notation as an (n, k, d_{min}) code. This metric quantifies the minimum number of bit flips required to change a binary message into other, i.e., the minimum number of errors that could transform one codeword into other. Intuitively, a code with minimum Hamming distance d_{min} between its codewords can detect at most $d_{min} - 1$ errors. The Theorem (7) presents the relationship between d_{min}, n and k.

Theorem 6. For a linear block code $\mathcal{C}$ with parity check matrix $\mathbf{H}$, the minimum distance between the codewords is given by the smallest positive number of linearly dependent columns of $\mathbf{H}$.

Proof. Adopting $\mathbf{h_0},\ \mathbf{h_1},\ \cdots,\ \mathbf{h_{n-1}}$ the collection of columns from $\mathbf{H}$, since $\mathbf{cH}^{\mathsf{T}} = \mathbf{0}$, for any codeword $\mathbf{c} = (c_0,\ c_1,\ \cdots,\ c_{n-1})$, we have

$$c_0\mathbf{h_0} + c_1\mathbf{h_1} + \cdots + c_{n-1}\mathbf{h_{n-1}} = \mathbf{0} \tag{1.45}$$

Being $\mathbf{c}$ the codeword of smallest weight $w(\mathbf{c}) = d_{min} = w$, as stated in Theorem (5), let us consider the non-zero position at indices $i_1,\ i_2,\ \cdots,\ i_w$ such that

$$c_{i1}\mathbf{h_{i1}} + c_{i2}\mathbf{h_{i2}} + \cdots + c_{iw}\mathbf{h_{iw}} = \mathbf{0} \tag{1.46}$$

In this case, the columns of $\mathbf{H}$ corresponding to the non-zero indices of $\mathbf{c}$ are linearly dependent, i.e., if we had a linear set of columns $u < w$, there would be a codeword of weight u, which contradicts the Theorem (5). □

Theorem 7 (The Singleton Bound)**.** The minimum distance for a (n, k) linear code is bounded by

$$d_{min} \leqslant n - k + 1 \tag{1.47}$$

Proof. The parity check matrix of a (n, k) linear code has $n-k$ linearly independent rows. The row rank of a matrix is equal the column rank. Hence, any set of $n-k+1$ columns is expected to be linearly dependent, according to the Theorem (6).

When Equation (1.47) is an equality, the code is called *Maximum Distance Separable* (MDS) code [18]. □

Obtaining the original collection of codewords from a given matrix $\mathbf{G}$ transformed by a row and a column permutation given by the matrices $\boldsymbol{\eta}$ and $\boldsymbol{\Omega}$ respectively, might be useful when, for some reason, we need to create a more convenient structure for this matrix. For instance, we can write an equivalent code, $\mathbf{G}'$, by using the set of operations listed in the Theorem (4), as

$$\mathbf{G}' = \boldsymbol{\eta}\ \begin{matrix} k \end{matrix}\!\!\begin{pmatrix} \overset{n-k}{\mathbf{P}} & \overset{k}{\mathbf{I}} \end{pmatrix}\ \begin{matrix} \overset{n-k}{\mathbf{I}} & \overset{k}{\mathbf{0}} \\ \mathbf{0} & \boldsymbol{\Omega} \end{matrix}\!\!\begin{matrix} n-k \\ k \end{matrix}$$

For convenience, we state that our transformation over the matrix $\mathbf{G} = (\mathbf{P}\ \ \mathbf{I})$ will generate a new matrix $\mathbf{G}'$ still in the usual form $\mathbf{G}' = (\mathbf{P}'\ \ \mathbf{I})$ by enforcing that $\boldsymbol{\eta} = \boldsymbol{\Omega}^{\mathsf{T}}$. From this assumption we can note that

$$\mathbf{c} = \mathbf{u}\begin{pmatrix} \mathbf{P} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{uP} & \mathbf{uI} \end{pmatrix} \tag{1.48}$$

and the equivalent codeword $\hat{\mathbf{c}}$ is

$$\hat{\mathbf{c}} = \mathbf{u}\begin{pmatrix} \boldsymbol{\eta}\mathbf{PI} & \boldsymbol{\eta}\mathbf{I}\boldsymbol{\Omega} \end{pmatrix} \tag{1.49}$$

Thus, when the process of encoding is performed from the matrix $\mathbf{G}'$, we know how to reassign the codewords, $\hat{\mathbf{c}}$, for those from the original equivalent code associated to $\mathbf{G}$ by simply applying the following transformation:

$$\mathbf{c} = \mathbf{u}\begin{pmatrix} \boldsymbol{\eta}\mathbf{P}\mathbf{I}\mathbf{T}_2 & \boldsymbol{\eta}\mathbf{I}\boldsymbol{\Omega}\mathbf{T}_1 \end{pmatrix} \tag{1.50}$$

where $\mathbf{T}_1$ and $\mathbf{T}_2$ are the transformation matrices given by

$$\mathbf{T}_1 = \mathbf{I} \tag{1.51}$$

$$\mathbf{T}_2 = \mathbf{P}^{-1}\boldsymbol{\eta}^{-1}\mathbf{P} \tag{1.52}$$

1.4.1 BCH Codes

The BCH codes are named for their creators Bose, Ray-Chaudhuri and Hocquenghem, who published their work in 1959 and 1960. Binary BCH codes were firstly proposed by Hocquenghem [20], in 1959 and independently by Bose and Chaudhuri, in 1960 [21]. The first decoding algorithm was devised by Peterson in 1960 [22] Peterson's algorithm was then generalised and improved by Chien [23], Forney [24], Berlekamp [25], Massey [26] and others.

These codes introduced, for the first time, a mean for designing codes over $\mathsf{GF}(2)$ from specified minimum distances [18]. As we will see next, they can be specified by a generator polynomial whose coefficients come from the finite field of two elements, $\mathsf{GF}(2)$. Nonetheless, the computation used by the error correction algorithms is usually performed over a larger finite field, $\mathsf{GF}(2^m)$. The two most important parameters for a binary BCH code are usually referred to as k and t. The parameter k expresses the number of data bits to be encoded. The parameter t establishes the maximum number of bit errors this code is capable of correcting.

Encoding

A BCH code over $\mathsf{GF}(q)$ is usually defined by a generator polynomial $g(x)$. A binary BCH code of length n is capable of correcting at least t errors. The generator polynomial of this code is specified in terms of its roots in a finite field $\mathsf{GF}(2^m)$. Being α a primitive element of $\mathsf{GF}(2^m)$, the generator polynomial $g(x)$ for a t-error correcting BCH code of length $n = 2^m - 1$ is the lowest degree polynomial over $\mathsf{GF}(2)$ that has $\alpha, \alpha^2, \cdots, \alpha^{2t}$ as its roots, i.e., $g(\alpha^i) = 0, \;\; i \in \{1, 2, \cdots 2t\}$. Let $M_i(x)$ be the minimal polynomial of α^i. Then the generator polynomial, $g(x)$, is given by the least common multiple (LCM) of $M_1(x),\ M_2(x),\ \cdots,\ M_{2t}(x)$. Due to the repetition of conjugate roots, it can be shown that $g(x)$ is formed by only the odd indices of the minimal polynomials, i.e, every even power of α in $\{\alpha,\ \alpha^2,\ \cdots,\ \alpha^{2t}\}$ has the same minimal polynomial as the preceding odd power of α. As a result, $g(x)$ is reduced to

$$g(x) = \mathrm{LCM}\{M_1(x),\ M_3(x),\ \cdots,\ M_{2t-1}(x)\} \tag{1.53}$$

and the codeword is obtained from $c(x) = g(x)m(x)$.

Because the minimal polynomial has degree at most m, the generator polynomial has degree at most mt, i.e., a t-error correcting BCH code requires $n - k \leqslant mt$ parity check bits.

For any positive integer $m \geqslant 3$ and $t \leqslant 2^{m-1}$, there always exists a binary BCH code (n, k) with the following parameters [18]:

- $n = 2^m - 1$ *code length*
- $n - k \leqslant mt$ *number of parity check bits*
- $d_{min} \geqslant 2t + 1$ *minimum Hamming distance*
- error correcting capacity *t errors/codeword*

Example 1.4.2: BCH code design

In this example we aim to design a binary BCH code for a codeword length $n = 30$, i.e, we will adopt the extension field $GF(2^5)$, for this code. Given the code capacity, $t = 3$, the set of roots is originally given by $\{\alpha,\ \alpha^2, \alpha^3,\ \alpha^4,\ \alpha^5,\ \alpha^6\}$ and then reduced to $\{\alpha, \alpha^3,\ \alpha^5\}$, due to the reasons described before. By taking the minimal polynomials from the Table 1.5, we are able to design the following generator polynomial.

$$\begin{aligned} g(x) &= (x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1)(x^5 + x^4 + x^2 + x + 1) \\ &= x^{15} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1 \end{aligned}$$

The length of the message is given by $k = n - \text{degree}\{g(x)\} + 1$. In this case, $k = 16$ and the code rate is $r = \frac{k}{n} = \frac{16}{30} = 0.533$.

Decoding

Let us define a codeword polynomial $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$, where the set $\{c_0,\ c_1,\ \cdots,\ c_{n-1}\}$ corresponds to the polynomial coefficients in $GF(2)$. This codeword is transmitted over a channel impaired by noise. Because of that, some errors are introduced in the original polynomial and, as a consequence, the received polynomial becomes

$$r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1}x^{n-1}$$

The introduction of errors by the channel can be modelled as the addition of a general error polynomial $e(x)$. Then

$$r(x) = c(x) + e(x)$$

The first step for decoding a BCH code is to compute the syndrome polynomial $s(x)$ from the received signal $r(x)$. For a t-error correcting BCH code, the syndrome has $2t$ elements

$$\mathbf{s} = (s_1,\ s_2,\ \cdots,\ s_{2t})$$

Assuming $e(x)$ contains ν errors ($\nu < t$) at the hypothetical locations specified by $\left(x^{j_1},\ x^{j_2},\ \cdots,\ x^{j_\nu}\right)$, then

$$e(x) = x^{j_1} + x^{j_2} + \cdots + x^{j_\nu}$$

The relationship between the components of the syndrome and the error location is expressed by the following set of equations

$$\begin{aligned} s_1 &= e(\alpha) = \alpha^{j_1} + \alpha^{j_2} + \cdots + \alpha^{j_\nu} \\ s_2 &= e(\alpha^2) = \alpha^{2j_1} + \alpha^{2j_2} + \cdots + \alpha^{2j_\nu} \\ &\vdots \\ s_{2t} &= e(\alpha^{2t}) = \alpha^{2tj_1} + \alpha^{2tj_2} + \cdots + \alpha^{2tj_\nu} \end{aligned}$$

where $\left(\alpha^{j_1},\ \alpha^{j_2},\ \cdots,\ \alpha^{j_\nu}\right)$ are unknown a priori. The method for decoding a BCH code is any method for solving the presented set of equations. Once we have determined $\left(\alpha^{j_1},\ \alpha^{j_2},\ \cdots,\ \alpha^{j_\nu}\right)$, the exponents $(j_1,\ j_2,\ \cdots,\ j_\nu)$ will tell us the error positions from the codeword, according to the method described next.

The second step for decoding corresponds to the computation of an error locator polynomial. This step is accomplished through an algorithm such as the Berlekamp-Massey or the Euclid's algorithm [18]. Let $\beta_u = \alpha^{j_u},\ u \in \{1,\ 2,\ \cdots,\ \nu\}$. Then we can redefine the syndrome as

$$\begin{aligned} s_1 &= \beta_1 + \beta_2 + \cdots + \beta_\nu \\ s_2 &= \beta_1^2 + \beta_2^2 + \cdots + \beta_\nu^2 \\ &\vdots \\ s_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \cdots + \beta_\nu^{2t} \end{aligned}$$

In order to decode a binary BCH code, it is convenient to define the error locator polynomial, which is given by

$$\begin{aligned} \boldsymbol{\lambda}(x) &= (1+\beta_1 x)(1+\beta_2 x)\ \cdots\ (1+\beta_\nu x) = \prod_{i=1}^{\nu}(1+\beta_i x) \\ &= \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_\nu x^\nu \end{aligned}$$

The roots of $\boldsymbol{\lambda}(x)$ are $\{\beta_1^{-1},\ \beta_2^{-1},\ \cdots,\ \beta_\nu^{-1}\}$ which correspond to the inverse of the error location number. The coefficients λ_i are related to the syndrome components by the Newton's identity

$$\begin{aligned} s_1 + \lambda_1 &= 0 \\ s_2 + \lambda_1 s_1 + 2\lambda_2 &= 0 \\ s_3 + \lambda_1 s_2 + \lambda_2 s_1 + 3\lambda_3 &= 0 \\ &\vdots \\ s_\nu + \lambda_1 s_{\nu-1} + \cdots + \lambda_{\nu-1} s_1 + \nu\lambda_\nu &= 0 \\ &\vdots \end{aligned} \tag{1.54}$$

Berlekamp-Massey algorithm is an iterative routine for finding the coefficients from the error locator polynomial that satisfy the Newton's identity from Equations (1.54). For more details about this algorithm, refer to [25].

The third step aims to find the roots of the error locator polynomial. In this thesis we will not describe the algorithm details, but only remark, in the next paragraph, the expected behavior from this routine. More details about this algorithm are found in [23].

The classical algorithm for the root finding step is the Chien Search. Through this algorithm, the roots of $\boldsymbol{\lambda}(x)$ can be obtained by substituting $\{1,\ \alpha,\ \alpha^2,\ \cdots,\ \alpha^{2^m-1}\}$ into $\boldsymbol{\lambda}(x)$. Since $\alpha^n = 1$, then $\alpha^{-1} = \alpha^{n-1}$ and hence, being α^1 a root of $\boldsymbol{\lambda}(x)$, α^{n-1} is an error location number. In this case, the received bit at position $n-1$ is supposed to be flipped. Chien's Search is considered a fast algorithm for determining polynomial roots. This approach has a significantly reduced complexity when compared to a brute-force approach.

1.4.2 Low-Density Parity Check Codes

The LDPC codes belong to the class of linear block codes defined by a sparse parity check matrix $\mathbf{H}$. As seen in Equation (1.31), each row of this matrix is assigned to a constraint of parity checking, while each column is associated to a bit of the codeword. The number of rows of $\mathbf{H}$ is given by the difference among the length of the codeword and the length of the message, i.e., $m = n - k$. A received codeword vector, $\mathbf{r}$, is considered a valid codeword, $\mathbf{c}$, if it satisfies all the parity check constraints, i.e., if the syndrome vector is equal to zero, as follows

$$\mathbf{H}\mathbf{r}^T = 0 \tag{1.55}$$

These codes can be defined from a parity check matrix $\mathbf{H}$ as long as $\mathbf{G}$ can be obtained from permutations and combinations in the elements of $\mathbf{H}$, as seen in Equations (1.34) and (1.35). For a given source message u of length k, the encoded string will be the product in $GF(2)$ between the message and the generator matrix G

$$\mathbf{c} = \mathbf{u}\mathbf{G} \tag{1.56}$$

The parameters of sparsity of $\mathbf{H}$ are known as degrees of distribution of the code. The number of non-zero elements in a row or in a column of $\mathbf{H}$ is the weight i. Usually, we define the fraction of columns with weight i by ν_i and, the fraction of rows with weight i by ζ_i. In general, it follows that

$$m \sum_i \zeta_i i = n \sum_i \nu_i i \tag{1.57}$$

A regular-LDPC code is one in which both, row and column weights, are constant. It means that Equation (1.57) can be reduced to Equation (1.58) since the fraction of 1s in columns and

rows of $\mathbf{H}$, ν and ζ respectively, are constant too

$$m\zeta = n\nu \tag{1.58}$$

The LDPC decoding can be accomplished with either of two decision methods: hard-decision and soft-decision. The input to a hard-decision decoder consists of two levels of the binary bits 0 and 1. In this decoding method, statistical information from the channel that is available in the received signal is lost. However, the lower complexity involving hard-decision decoding is widely used in the scenarios where complexity is strongly limited.

On the other hand, the input to a soft-decision decoder is a multilevel quantised signal. In other words, to fully employ the information in a received waveform and thus, enhance the decision accuracy of the decoder, a quantisation is performed in the signal, improving the system performance. As a result, at the same rate, soft-decision methods use to bring a higher coding gain than hard-decision. However, this requires a high-speed Analog-to-Digital Converter (ADC) to perform sampling and quantisation. Additionally, the soft-decision algorithms are very complex because they must consider the changes in noise probability distribution caused by channel performance deterioration. Due to that, this method uses to greatly increase the processing complexity.

The LDPC codes can be iteratively decoded using *Belief propagation* algorithms that will be discussed in the next chapter. These algorithms are based on soft-values which provide estimates of the confidence of each individual symbol and the parity check matrix, which establishes relationships between the information bits and the parity bits within a codeword. There are several well-known ways of obtaining these soft-decoder output values, starting with the method developed by Gallager [1], in 1963. All the A Posteriori Probability (APP) and Maximum a Posteriori (MAP) algorithms proposed later, such as the sum-product and the min-sum, are based on the channel statistics, i.e., they use probabilities or Log-Likelihood Ratios (LLR) as a decoding metric. In this thesis, we will present more details about the LLR estimation of *a priori probability* from channel in Chapter 3, where two new simplification methods will be described.

A Tanner graph is an effective graphical representation for an LDPC code. This is characterised as a bipartite graph with vertex set $V = V_1 \cup V_2$. Each edge has one endpoint in V_1 and one in V_2. The graph is split into two distinctive sets whereas the edges are connecting nodes of two different types. For an LDPC code with parity check matrix $\mathbf{H}$, this representation has one vertex in V_1 for each row of $\mathbf{H}$ and one vertex in V_2 for each column of $\mathbf{H}$. An edge connects two vertices i and j when the element from $\mathbf{H}$ is $h_{ij} = 1$. The two types of nodes in a Tanner graph are usually called check-nodes (V_1) and variable nodes (V_2).

The decoding complexity of each iteration of a decoding algorithm is proportional to the number of edges in the Tanner graph. The sparseness is a distinguished characteristic that can make the code more tractable to many iterative decoding algorithms, which are able to provide

near optimum performance. This is why sparsity of the parity check matrix is of great importance, and explains the value of LDPC codes.

The effectiveness of an iterative decoder depends on many structural properties of the Tanner graph. For instance, one of these structural properties may be described as a sequence of edges that form a closed path of length $\mathfrak{n}$, usually called $\mathfrak{n}$-cycles. These cycles and their consequences will be discussed with more details in Chapter 2.

1.5 DVB-S2 standard

The DVB-S2 is a digital broadcast standard developed in 2003 for satellite broadband applications such as TV and sound broadcasting, interactivity (Internet access) and other services. Channel coding and modulation are based on the inner LDPC code plus the outer BCH code serially concatenated and four modulation schemes: QPSK, 8-PSK, 16-APSK and 32-APSK [27].

The DVB-S2 transmitter system is organised as a sequence of functional blocks, schematically represented in Figure 1.7. The transmission is based on two levels of framing structures:

- BBFRAME at baseband (BB) level holds a variety of signaling bits, to configure the receiver according to the channel scenario;
- PLFRAME at physical layer (PL) level holds a few highly protected signaling bits to provide synchronisation robustness and signaling.

The Forward Error Correction (FEC) and modulation subsystems in DVB-S2 are the key stages in terms of the excellent performance achieved by this standard in the presence of high levels of noise and interference. This is based on LDPC code with a very powerful error correction capability which is increased by an outer BCH code with low additional overhead. The main purpose of the BCH code is to avoid eventual error floors from LDPC at very low error rates or equivalently, very high SNRs. The FEC approach is described as a concatenated coding although the error correction capability of LDPC dominates the performance by far. The BCH error correction is limited to 8, 10 or 12 bits, depending on the LDPC code rate configuration. There are eleven possible code rates (1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, and 9/10) that can be chosen depending on the modulation mode and the system application.

Error rate requirements for DVB-S2 are very strict. The system may operate with a residual packet error rate (PER) of less than 10^{-7} [3]. For MPEG transport streams, which are characterised by a 188-byte packet, this rate corresponds to approximately less than one erroneous packet per hour for a service bit rate of 5Mb/s. In practice, the distance from the modulation-constrained Shannon limit usually ranges from 0.7 to 1.2dB [27].

The Physical Layer (PL) frame is either composed of 64800 bits in *normal* length configuration or 16200 bits in *short* length configuration. The header contains 90 BPSK symbols which precedes the frame's payload. They hold synchronisation and signaling information which allows

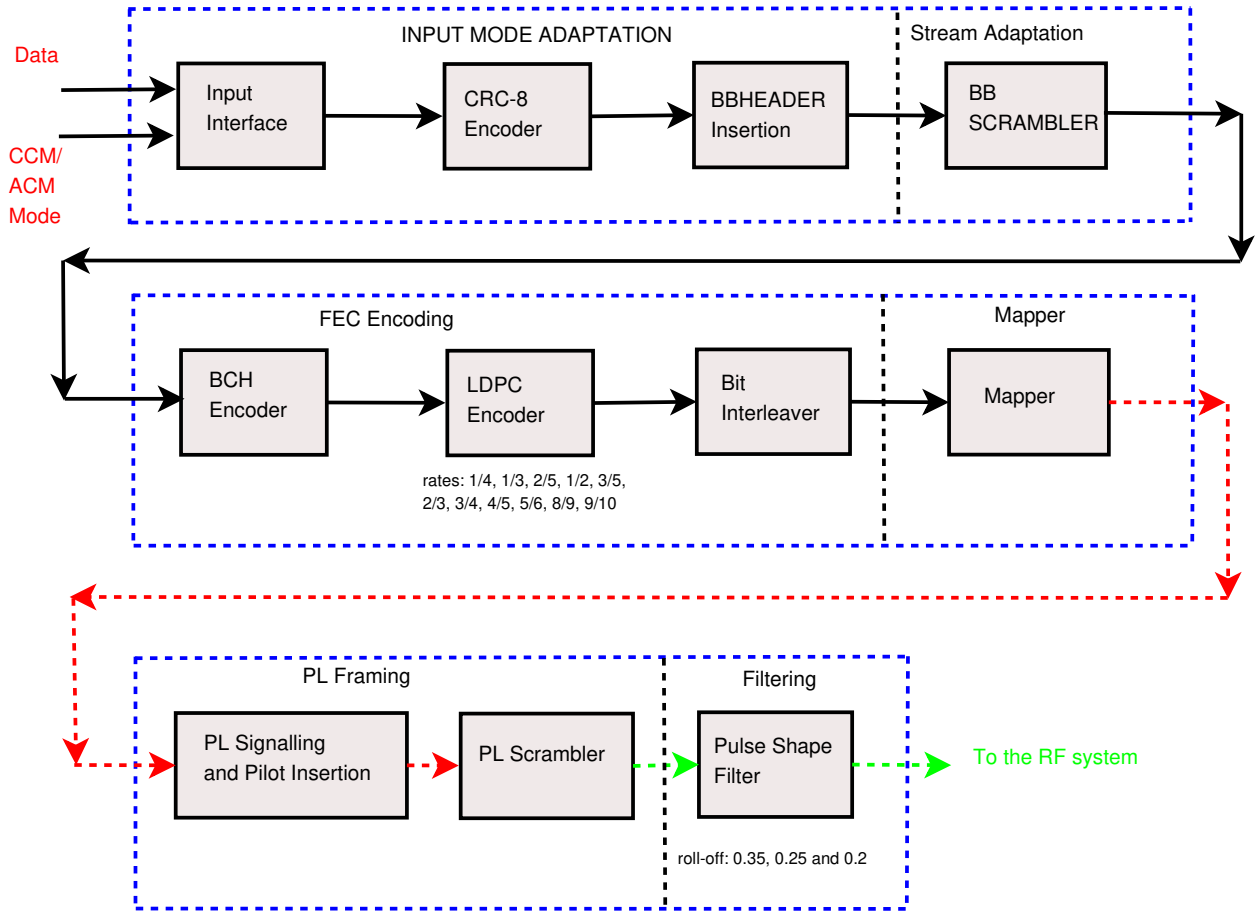

Figure 1.7: Functional block diagram of the DVB-S2 transmitter

the receiver to synchronise (carrier and phase recovery, frame synchronisation) and to detect the modulation and coding parameters.

The first 26 BPSK symbols ($18D2E82_{[\mathrm{HEX}]}$) of the PL header indicate the start of the frame. This fixed sequence of symbols are known as SOF (start of frame). The remaining 64 symbols are employed for signaling the system configuration. Since the PL Header is the first entity to be decoded by the receiver it can not be protected by the FEC scheme as it contains header information that are required for decoding (e.g. adopted modulation scheme and code rate). Nevertheless, the header must be well decodable under the worst-case link conditions [27]. Therefore the main signaling information at this level is reduced to 7 bits (5 bits used to indicate the modulation and coding configuration, 1 bit for frame length (*normal* or *short*) and 1 bit for presence/absence of pilots to facilitate synchronisation). These bits are highly protected by an interleaved first-order Reed-Muller [18] block code with the parameters rate $(64, 7, \mathrm{t} = 32)$.

This thesis employs a full digital simulator developed in Matlab which combines the transmitter structure from Figure 1.7 and the receiver structure from Figure 1.8. The DVB-S2 simulator

collects the error rate at each functional block output as well as the number of iterations performed by the LDPC decoder. The designed transmitter can generate random data or read data from a given file. The simulator is fully compliant with all the available configurations from normal and short frame, all the code rates and modulations. The receiver is able to be configured for three different methods of soft-demapping [28] [29] [30]. The calculations can be performed from Max-Log or any proposed simplification that will be introduced in Chapter 3. There are also four different algorithms available for LDPC decoding (the sum-product, the Min-Sum and two other variants algorithms). These algorithms as well as the maximum number of iterations and the desired signal-to-noise ratio level can be set according to the simulation purposes. All the simulations performed in this thesis employed this tool.

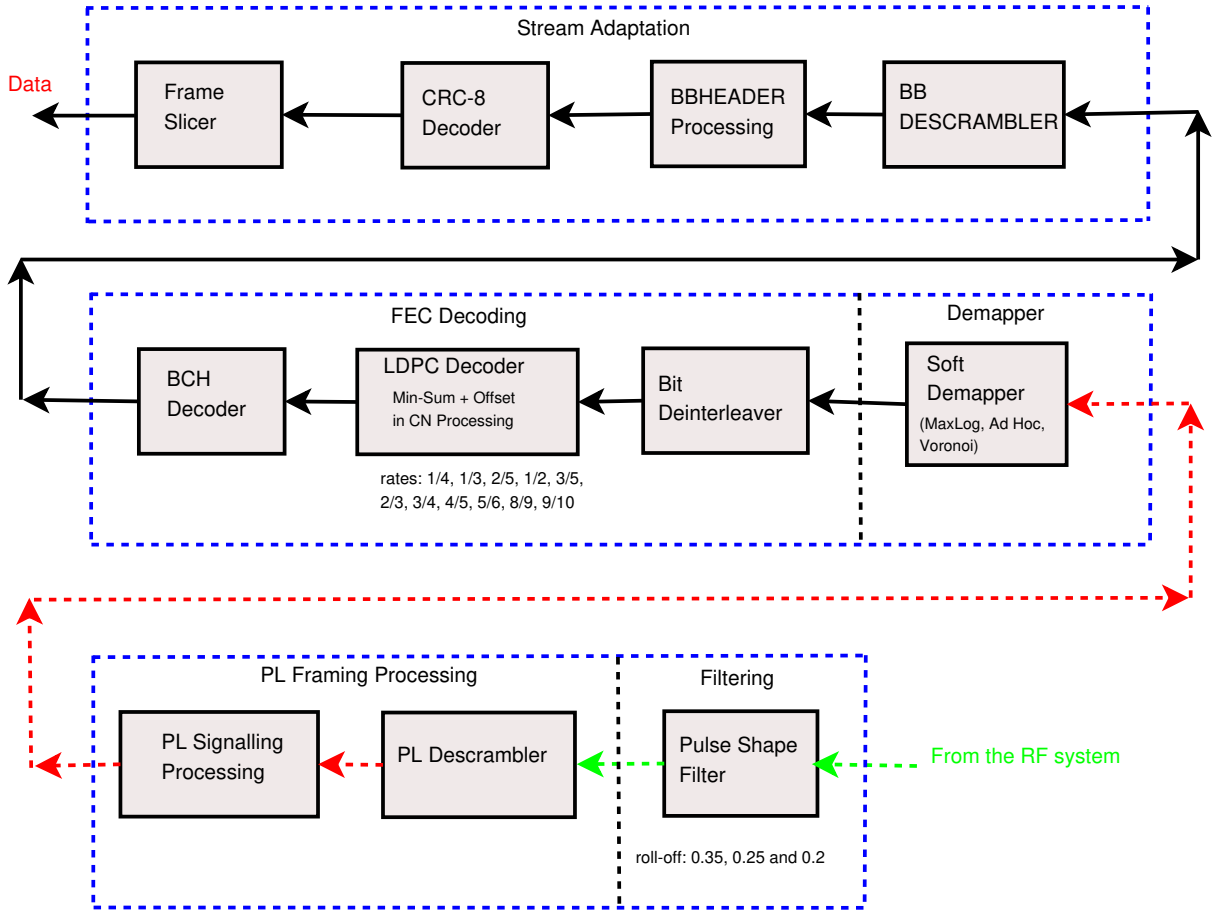

Figure 1.8: General structure of the DVB-S2 receiver.

Chapter 2

LDPC Codes

The LDPC codes are linear codes over GF($2^{\mathfrak{n}}$) ($\mathfrak{n} = 1$ for binary LDPC, $\mathfrak{n} > 1$ for non-binary LDPC codes) whose constraint graph is sparse. They were introduced and analysed by Gallager [31] in a paper that was forgotten for several years and recalled only in the 1990s, after LDPC codes were rediscovered by MacKay [32]. Aside from the sparsity requirement of $\mathbf{H}$, the fact that these codes can make use of soft-information from the channel output and also allow the use of very long codewords, the LDPC codes are not different from any other block code. Block codes are usually shorter and algebraically designed to be as less complex as possible. On the other hand, LDPC decoding is based on an iterative exchange of information (*Message Passing*) among bit nodes and check nodes in order to determine the transmitted information (bit values). Message passing decoding algorithms are based on a messages exchange along the edges of a corresponding Tanner graph. Therefore an LDPC code is usually designed based on the properties of its parity check matrix, $\mathbf{H}$.

In this thesis we will only consider binary LDPC codes. The decoding algorithms for these codes may be classified into hard-decision and soft-decision algorithms. The hard decision algorithms quantise the received signal into two numerical levels stated by a threshold. If the received signal is below this threshold, the decoder output is set to 0. Otherwise, if the signal is above the threshold, the decoder output is set to 1. On the other hand, the soft-decision decoders calculate a graded soft-decision confidence measurement related to the probability of the bit to be one or zero. The Maximum Likelihood (ML) soft-decision decoding algorithms provide a significant performance gain in comparison to the hard-decision decoding methods. In this chapter we will describe both types of decoding. We also focus on the Belief Propagation (essentially the sum-product algorithm, which uses highly nonlinear operations) and the Min-Sum algorithm. We aim to exploit the theoretical concepts by presenting some *toy examples* and small simulations along the chapter.

2.1 Hard-Decision Decoders and Soft-Decision Decoders

The first step of hard-decision algorithms is made over each component of the received signal, $\mathbf{r}$, resulting in a vector

$$\boldsymbol{x}' = \begin{pmatrix} \hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_n \end{pmatrix}$$

For instance, for a BPSK baseband real signal $\{-1,\ +1\}$ we would have

$$x'_j = \text{sign}(r_j)$$

If the vector $\boldsymbol{x}'$ is a valid codeword that belongs to $\mathcal{C}$, then the decoder selects $\hat{\mathbf{x}} = \boldsymbol{x}'$, otherwise the structure of the error correction code is exploited to correct it.

The optimal decision rule, in this case, becomes the minimum Hamming distance. In other words, this sot of algorithm decodes $\mathbf{r}$ as the nearest codeword $\hat{\mathbf{c}}$ to $\boldsymbol{x}'$. We can use this method when the number of codeword is small. However, as the number of codewords increases, it becomes expensive to search and compare all the codewords. The decoder must compare a demodulated codeword $\boldsymbol{x}'$ to $m = 2^k$ possible codewords $\mathbf{c}_\ell \in \mathcal{C}$ deciding for the codeword $\hat{\mathbf{c}}$ which is closer to $\boldsymbol{x}'$ in terms of the Hamming distance. If there is more than one codeword with the same Hamming distance, the decoder takes a random decision. In this sense, an iterative decoding algorithm is much more proper and less computationally expensive.

Instead of using the Hamming distance, the soft-decision algorithms are based on the Euclidean distances. For an AWGN channel, $\mathcal{N}(0, 2\sigma^2)$ whose the variance per dimension is σ^2, the probability density function of $\mathbf{r} \in \mathbb{C}$ conditioned to $\mathbf{s}_\ell \in \mathbb{C}$ is given by

$$f_{R|S}(\mathbf{r}|\mathbf{s}_\ell) = \frac{1}{2\pi\sigma^2} e^{-\frac{\|\mathbf{r}-\mathbf{s}_\ell\|^2}{2\sigma^2}} \tag{2.1}$$

Since the transmitted information is equally likely, the MAP rule and the ML decision rule are both equivalent [8]. Thus the optimal soft-decision estimated symbol is defined by the following criterion

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}_\ell \in \mathbb{C}} \left(f_{R|S}(\mathbf{r}|\mathbf{s}_\ell) \right) \tag{2.2}$$

As the Equation (2.1) is a monotonically decreasing function, we may have the soft-decision estimation of $\hat{\mathbf{s}}$ in terms of the minimisation of a distance, as follows

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}_\ell \in \mathbb{C}} \left(\|\mathbf{r} - \mathbf{s}_\ell\|^2 \right) \tag{2.3}$$

The estimated symbol $\hat{\mathbf{s}}$ and the codeword $\hat{\mathbf{c}}$ are systematically associated to each other according to the bit mapping from the adopted modulation.

The most important message passing algorithm based on hard-decision is known as *bit-flipping* decoding. The use of this algorithm does not require to compare the hard estimated codeword vector $\boldsymbol{x}'$ to the m possible codewords from $\mathcal{C}$. Instead, the messages exchanged

between the Tanner nodes of the code are formatted as binary information. Essentially, the bit-flipping algorithm consists of an interactive decoding algorithm that exchange binary messages passed over bit-nodes and check-nodes and toggles bits in error according to the majority vote of the messages exchanged until it finds a valid codeword. The method is sub-optimal as it discards potentially useful information from the received signal $\mathbf{r}$.

The soft-decision algorithms exchanges messages in the format of probabilities or Log-Likelihood Ratios, that represent the level of confidence involved in the estimation. As seen before, the confidence is not taken into account on hard-decision decoding algorithms. For better illustrating these differences, the bit-flipping algorithm and the most important soft-decision decoding algorithms will be detailed in the next dedicated subsections.

2.1.1 Bit-Flipping

This iterative decoding algorithm is based on exchanging messages between the bit nodes and the parity-check nodes. Initially, the decoder takes the results from the hard channel outputs. Consider a baseband BPSK signal $\{-1, +1\}$ and a received vector

$$\mathbf{r} = \begin{pmatrix} 1.22 & -0.44 & -1.21 & 0.4 & 1.47 \end{pmatrix}$$

in the output of an AWGN channel. The hard decoder detects

$$\mathbf{x}' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Intuitively we may realise this decision was taken with different levels of confidence for each bit since some of the received values are closer to the transmitted levels than others. In other words, the uncertainty over these bits is clearly lower, given their closeness to the threshold level. Nevertheless, this information is not taken into account on hard-decision algorithms. As we will see next, this is the most remarkable difference between the hard and soft-decision decoders.

After the first stage of hard-decision applied over the channel output values, the stage of bit-flipping decoding algorithm begins. As a visual interpretation based on the Tanner graph, we can imagine that from the input $\mathbf{x}'$, the bit-nodes send the received bits for all the parity-check nodes they are connected. Thenceforth, each parity-check node evaluates the local response by calculating the corresponding syndrome s_i. The expected response at the end of the check-node processing is $s_i = \sum_{j=1}^{n} h_{i,j} x'_j = 0 \ \forall \ j \in \{1, \cdots, n\}, \ i \in \{1 \cdots, n-k\}$ as we already stated in section (1.4). If all the parity-check equations are satisfied, i.e., $\mathbf{s} = \mathbf{0}$, the algorithm ends. Otherwise, the check-node sends back a message to each bit-node, telling it if the parity check equation is currently satisfied. Then the bit-nodes use the messages processed by the check-nodes in order to decide the bit value by majority vote, i.e., it tells a check node that it will flip its decision if most of the connected check-nodes tell it to do so. The bit-nodes send the processed messages once again to the connected parity-check nodes and the decoding process restarts either until a maximum number of iterations or until a successful decoding based on the result of the syndrome vector.

The Algorithm (1) described next is a concise statement of the bit-flipping (BF) steps. There are three required inputs: the binary vector $\mathbf{x}'$, corresponding to the output of the hard decoder, the parity check matrix, $\mathbf{H}$, and the maximum number of iterations allowed, $\mathrm{I}_{\mathrm{MAX}}$. One refers to $\mathcal{M}(\mathtt{j})$ as the set of check nodes connected to a bit node $\mathtt{j}$ and to $\mathcal{N}(\mathtt{i})$ as the set of bit nodes connected to a check node $\mathtt{i}$.

The BF is an iterative hard-decision decoder for the Binary Symmetric Channel proposed in [1]. Bit-flipping is sub-optimal in terms of performance, but is still surprisingly good, given the extreme simplicity of the algorithm. As mentioned before, the bit-flipping decoder, also known as the *Gallager B decoder*, works on the Tanner graph of the parity-check matrix by exchanging binary messages over the edges [33]. Since the input and output alphabets are binary, all operations of the decoder are defined over $\mathsf{GF}(2)$.

Algorithm 1 - Bit-Flipping Decoding

Require: x'
Require: H
Require: I_{MAX}

$I_{iter} = 0$

for $j = 1 : n$ **do**
  $M_j = x'_j$
end for

repeat

  for $i = 1 : m$ **do**
    for $j = 1 : n$ **do**
      $E_{i,j} = \sum_{j' \in \mathcal{N}_i} \oplus(M_{j'})$ [1] ▷ Check node processing
    end for
  end for

  for $j = 1 : n$ **do**
    if the majority of the check-node messages $E_{i,j}$ are failing **then** ▷ Bit node processing
      $M_j = M_j \oplus 1$ ▷ $\mathcal{M}_j$ is flipped
    end if
  end for

  for $i = 1 : m$ **do** ▷ Syndrome Calculation
    $s_i = \sum_{j' \in \mathcal{N}_i} \oplus(M_{j'})$
  end for

  if all $(s_i = 0)$ or $(I_{iter} = I_{MAX})$ **then**
    FINISHED
  else
    $I_{iter} = I_{iter} + 1$ ▷ Increment iteration counter
  end if

until FINISHED

The intuitive thinking behind the bit-flipping is that the greater the number of unsatisfied parity check nodes connected to a particular bit, the higher the belief that this bit is in error. However, since such a bit-flipping algorithm uses binary signals from a hard decoder with probabilities and belief information ignored, the decoding algorithm only performs well in high signal-to-noise ratio scenarios.

[1] The operator $\oplus$ is defined as the addition in GF(2). Therefore, the notation $\sum_j \oplus M_j$ refers to the summation of M_j in GF(2)

Example 2.1.1: Bit-flipping

In this example we will use the parity check matrix presented below and a valid codeword $\mathbf{c} = \begin{pmatrix} 0\,1\,1\,1\,1\,0\,0\,0\,1\,0 \end{pmatrix}$. Aside from the requirement that $\mathbf{H}$ be sparse, as LDPC code is not different to any other block code, such as the one represented by the parity-check matrix in (2.4). Indeed, existing block-codes can be employed with any LDPC iterative decoding algorithm [34], including bit-flipping. Typical block-codes are generally decoded with ML decoding algorithm and due to that, they are generally short and specially designed for making this task less complex. LDPC codes are always iteratively decoded and therefore they are designed with focus on the properties of $\mathbf{H}$.

The bit-flipping algorithm is based on the idea that a codeword bit embedded in a large number of unsatisfied parity-check equations is more likely to be incorrect. The sparseness of $\mathbf{H}$ is only useful for spreading the codeword bits into the existing parity check equations so that these equations are unlikely to involve the same set of codeword bits. We will show the effect of overlapping the parity-check equations out of this example.

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{2.4}$$

Lets assume that this codeword is modulated using a baseband BPSK signal $\{-1,\ +1\}$. The passband signal is sent and corrupted by AWGN. It follows that the last bit is in error after the hard-decision demapping taken over the received vector $\mathbf{y}$. Thus, the received codeword is

$$\mathbf{x}' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & \mathbf{1} \end{pmatrix} \tag{2.5}$$

and therefore, the input of the bit-flipping decoder is initialisated as

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & \mathbf{1} \end{pmatrix} \tag{2.6}$$

In the first step of the algorithm, the check-node messages (parity-check equations) are evaluated and the results are placed into the check-node matrix E, in all the positions for which $\mathsf{H}_{i,j} = 1$. At the end of this step we obtain

$$\mathsf{E} = \begin{pmatrix} 0 & . & 0 & . & . & . & . & 0 & 0 & . \\ . & 0 & . & . & . & . & 0 & . & 0 & . \\ . & . & 1 & 1 & . & . & 1 & . & . & 1 \\ . & 0 & . & 0 & . & 0 & . & 0 & . & . \\ . & . & . & . & 1 & 1 & . & . & 1 & 1 \end{pmatrix} \tag{2.7}$$

As we can see in (2.4), the check-node 1, i.e., the first row of the parity-check matrix, is connected to the bit-nodes $\mathcal{N}_1 = \{1,\ 3,\ 8,\ 9\}$. The messages $\mathsf{E}_{1,j}$ are evaluated with respect to the check-node 1 and bit-nodes $j \in \mathcal{N}_1$. In this case, these messages are obtained as

follows

$$\begin{aligned} \mathsf{E}_{1,1} &= \mathsf{E}_{1,3} = \mathsf{E}_{1,8} = \mathsf{E}_{1,9} \\ &= \mathsf{M}_1 \oplus \mathsf{M}_3 \oplus \mathsf{M}_8 \oplus \mathsf{M}_9 \\ &= 0 \end{aligned}$$

The message updates from second, third, fourth and fifth check-node can be evaluated by following the same procedure.

The second step of bit-flipping consists on the update of new bit-node messages. Note that the first column of $\mathbf{E}$, in (2.7), has a single message from check-node 1 (the first parity-check equation). As the message from this check no is 0, i.e., the only parity-check equation connected to the bit-node 1 is satisfied, this bit is kept the same (no flipping). The fifth column of $\mathbf{E}$ has also a single message from check-node 5. As the message from this check-node is 1, the bit-node 5 is flipped. The same occurs in the tenth column of $\mathbf{E}$ in the first iteration. The majority of the check-nodes connected to the bit-node 10 is unsatisfied. At the end of this step of iteration 1, the bit-node messages are updated to

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 0 & 1 & \mathbf{0} \end{pmatrix} \tag{2.8}$$

The third step of this iteration consists on the calculation of the *syndrome* vector. If this is an all zero vector, there is no unsatisfied constraint and therefore, the messages from $\mathbf{M}$ will be returned as a valid codeword. Otherwise, the algorithm would return to the first step of the algorithm in order to update the matrix $\mathbf{E}$. At the end of this iteration, the syndrome is calculated as

$$\begin{aligned} \mathsf{s}_1 &= \mathsf{M}_1 \oplus \mathsf{M}_3 \oplus \mathsf{M}_8 \oplus \mathsf{M}_9 = 0 \\ \mathsf{s}_2 &= \mathsf{M}_2 \oplus \mathsf{M}_7 \oplus \mathsf{M}_9 = 0 \\ \mathsf{s}_3 &= \mathsf{M}_3 \oplus \mathsf{M}_4 \oplus \mathsf{M}_7 \oplus \mathsf{M}_{10} = 0 \\ \mathsf{s}_4 &= \mathsf{M}_2 \oplus \mathsf{M}_4 \oplus \mathsf{M}_6 \oplus \mathsf{M}_8 = 0 \\ \mathsf{s}_5 &= \mathsf{M}_5 \oplus \mathsf{M}_6 \oplus \mathsf{M}_9 \oplus \mathsf{M}_{10} = 1 \end{aligned}$$

As the result is a non-zero syndrome vector, we start a new iteration. The first step of the second iteration follows the same procedure described in the first iteration. The matrix $\mathbf{E}$ is now updated from the most recent bit-node messages, calculated in iteration 1 and shown in (2.8). The calculations result in

$$\mathbf{E} = \begin{pmatrix} 0 & . & 0 & . & . & . & . & 0 & 0 & . \\ . & 0 & . & . & . & . & 0 & . & 0 & . \\ . & . & 0 & 0 & . & . & 0 & . & . & 0 \\ . & 0 & . & 0 & . & 0 & . & 0 & . & . \\ . & . & . & . & 1 & 1 & . & . & 1 & 1 \end{pmatrix} \tag{2.9}$$

From the updated check-node messages in (2.9), we can now update the bit-node messages by following the same procedure described in step 2 of the algorithm. Thus, at the second iteration, we obtain the following

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.10}$$

Updating the syndrome we obtain $\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$. As mentioned before, this result is a stop-condition that ends the algorithm and return $\mathbf{M}$ as the final (valid) codeword.

Many LDPC codes are pseudo-randomly designed, i.e., the first step of construction is based on a random placement of 1s. Some bad configurations, such as the cycles of length 4 called 4-cycles, are either avoided during the design or removed afterwards. It is important to highlight the negative effect of the 4-cycles on the bit-flipping algorithm (not only for hard-decision algorithms but also for the soft-decision ones). This configuration can be identified by checking each pair of columns in $\mathbf{H}$ verifying if they overlap in two different check-node positions. A 4-cycle may also be graphically identified through the corresponding Tanner graph, as shown in Figure 2.1. The cycles lies in a sequence of 2 bit-nodes connected to two coincident check-nodes. The general length of a cycle is given by the number of edges it contains.

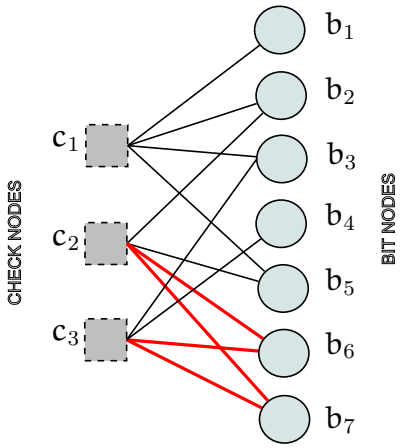

Figure 2.1: The bit nodes b_6 and b_7 are both connected to the check nodes c_3 and c_2. The four red edges represent a 4-cycle

Two different configurations of 4-cycles in the matrix $\mathbf{H}$ are illustrated below. This parity-check matrix has two columns that overlap 1s in two different rows, as emphasised in red and green.

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.11}$$

From the decoding point of view, a 4-cycle can involve two different bits in the same two parity check constraints. In this case, both equations are going to be unsatisfied and thus, it will not be possible to identify which bit is incorrect. Therefore the algorithm will not easily converge since the bits involved in the cycle will be flipped at the same time. When both of the parity-check equations are unsatisfied, it is not possible to determine which bit is erroneous.

For long codes, randomly choosing a parity-check matrix almost always produces a good code. Nevertheless, for practical applications the created codes may not be long enough and due to that, the matrix must be checked a posteriori to ensure it does not contain any 4-cycles. On the contrary, the whole decoding efforts might be to no avail [34].

2.1.2 Sum-Product

The sum-product is a soft-decision LDPC decoding algorithm. Rather than flipping bits, such as the hard-decision decoders, the class of soft-decision algorithms process the information from the bit probabilities coming from the channel statistics for each bit j, P_j^p, also called *a priori* probabilities. The aim of this algorithm is to compute the *a posteriori* probability for each bit, which consists on the probability that the j-th codeword bit is 0 or 1, conditional on the received channel output and also on the event $\mathbf{s} = \mathbf{H}\mathbf{c}^T = \mathbf{0}$, i.e., that all the parity-check constraints are satisfied if c_j assumes the adopted binary value. The sum-product chooses the bit value $c_j = 0$ or $c_j = 1$ associated to the maximum value of the a posteriori probability, i.e., $P(c_j = \{0, 1\} \mid r_j,\ \mathbf{H}\mathbf{c}^T = \mathbf{0})$.

As mentioned before, the decoding complexity for the true optimum decoding is exponential in k, since it requires an exhaustive search over all $m = 2^k$ possible codewords. Instead of doing this, the sum-product iteratively seeks for a codeword whose bits $c_j \in \{0,\ 1\}$ maximise each probability, $P(c_j = \{0, 1\} \mid r_j,\ \mathbf{H}\mathbf{c}^T = \mathbf{0})$. This is an equivalent measure of confidence on the established value of the bit j, under the condition that all the parity check constraints are satisfied with this decision.

The sum-product algorithm is an instance of message passing algorithm. For each edge $(i,\ j) \in \mathcal{C}$ that connects a check-node i to a bit-node j, the sum-product operates through two distinct sorts of message which operate in to two possible directions along the edges. As well as in the bit-flipping algorithm, these messages are exchanged through the Tanner graphs toward to the check-node or to the bit-nodes. In sum-product, the exchanged messages represent probabilities, i.e., the amount of reliability for a certain bit of the codeword. In what we call sum-product horizontal step, the probabilities from a bit-node are passed toward to the check-nodes it is connected. These messages are combined so that we obtain the probability that each check-node is satisfied, given the current state of the considered bit-node. In what we call vertical step, the messages are passed from a check-node toward to the bit-nodes connected to it. For each bit, the associated check-nodes probabilities are used to update the bit-node probability message. In the absence of cycles, such message passing algorithm compute exact probabilities. However, when the structure of cycles are present in the code, the sum-product algorithm is able to computes only approximated solutions [35].

In this thesis we will derive the sum-product exchanged messages in terms of log-likelihood ratios (LLR). This derivation will emphasise some of the likelihood arithmetic and highlight the calculations of extrinsic probabilities, a priori probabilities and a posteriori probabilities. We note that the LLR is only applicable to binary codes. For a binary variable x_i' at the output of any

type of soft-decoder, it is easy to find $P(x'_i = 1)$ given $P(x'_i = 0)$, since $P(x'_i = 1) = 1 - P(x'_i = 0)$ and so we only need to store one probability value for x'_i. Log-likelihood ratios are used to represent this sort of metric by one single value. For any modulation of order M, the LLR of these probabilities, $L(x'_i)$, is given by the ratio

$$L(x'_i) = \ln\left(\frac{P(x'_i = 0)}{P(x'_i = 1)}\right), \quad i \in \{0,\ 1,\ \cdots,\ \log_2(M) - 1\} \tag{2.12}$$

where $L(x'_i)$ is positive if $P(x'_i = 0) > P(x'_i = 1)$ and negative if $P(x'_i = 0) < P(x'_i = 1)$. Thus the sign of $L(x'_i)$ has direct consequence over the binary hard-decision of x'_i. In other words, if $L(x'_i) < 0$, x'_i will be set to one. Similarly, if $L(x'_i) > 0$, x'_i will be set to zero. On the other hand, the modulo $|L(x'_i)|$ is related to the reliability of the binary decision over x'_i. The greater the magnitude of $L(x'_i)$, the more confidence on the hard-decision involving this particular bit. Furthermore, this sort of metric has a significant weight on the reduction of decoder complexity since this replaces the multiplicative operations by additions.

The sum-product algorithm is an special case of the forward/backward algorithm, sometimes referred to as the BCJR or APP algorithm [36] in coding theory. This involves combination of behavioural (the graph) and probabilistic modeling. From the probabilistic model, we have the vectors P^o_j, which consists on the a posteriori probability and P^p_j, which consists on the a priori probability. This algorithm involves recursions on the calculation of two matrices of the same dimension of $\mathbf{H}$: one to compute the extrinsic messages as a function of the last update of $P^{ext}_{i,j'}$ as well as the value of P^p_j, and one to compute the intrinsic messages as a function of the last update of $P^{int}_{i,j'}$ as well as the value of P^p_j. Both recursions do not interact, so they can be computed in parallel [36]. The extrinsic probability is the probability that bit j makes the parity check i to be satisfied **if bit** j **is** 1, i.e.,

$$P^{ext}_{i,j} = \frac{1}{2} - \frac{1}{2}\prod_{j' \in \mathcal{N}(i), j' \neq j} 1 - 2P^{int}_{i,j'} \tag{2.13}$$

where $P^{int}_{i,j'}$ is the intrinsic probability, i.e., the current estimation, available for the check-node i, of the probability that bit j' is a one. Appendix A shows the mathematical proof of the Equation (2.13). The corresponding LLR of $P^{ext}_{i,j}$, sent from the check-node i to the bit-node j is given by

$$E_{i,j} = \mathrm{LLR}(P^{ext}_{i,j}) = \ln\left(\frac{1 - P^{ext}_{i,j}}{P^{ext}_{i,j}}\right) \tag{2.14}$$

Keeping focus on the manipulation of Equation (2.14), for a generic variable a, we can use the relationship

$$\tanh\left(\frac{1}{2}\ln\left(\frac{1-a}{a}\right)\right) = 1 - 2a \tag{2.15}$$

to write this equation as

$$E_{i,j} = \ln\left(\frac{1 + \prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right)}{1 - \prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right)}\right) \tag{2.16}$$

On the other hand, the term $M_{i,j'}$ corresponds to the LLR of $P^{int}_{i,j'}$, i.e.,

$$M_{i,j'} = \ln\left(\frac{1 - P^{int}_{i,j'}}{P^{int}_{i,j'}}\right) \tag{2.17}$$

The term $L_j = \mathrm{LLR}(P^o_j)$ is given by the sum of the LLR of its *a priori probability*, $\mathrm{LLR}(P^p_j)$, and the LLRs of the extrinsic probabilities associated to the set of connected parity check-nodes. Thus, for each iteration, the soft-output of the sum-product is

$$L_j = \mathrm{LLR}(P^p_j) + \sum_{i \in \mathcal{M}(j)} E_{i,j} \tag{2.18}$$

From the backward stage, the message sent from a bit-node to all its connected parity check-nodes, $M_{i,j}$, can not consider the total LLR, L_j for each bit-node. Instead of that, it must exclude the portion of LLRs coming from the parity check node which is being updated. This will ensure not sending back an information that was already computed. The LLR of intrinsic probability is then calculated at each iteration as

$$M_{i,j} = \mathrm{LLR}(P^p_j) + \sum_{i \in \mathcal{M}(j) i' \neq i} E_{i',j} \tag{2.19}$$

From now on, we focus on the derivation of the exact a priori LLR expression for a BPSK constellation onto Binary Symmetric Channel (BSC) and in AWGN. This is a mere introduction on this topic since we dedicate Chapter 3 for proposing, explaining and providing results of lower-complexity to approximate LLR expressions for higher order constellations.

As mentioned before, one input of the sum-product algorithm is the a priori probability LLR, which is calculated from the channel output, according to the channel model and the employed constellation. Thus, for a BSC, the corresponding LLR is defined as

$$\mathrm{LLR}(P^p_j) = \ln\left(\frac{P\left(c_j = 0 | x'_j\right)}{P\left(c_j = 1 | x'_j\right)}\right) = \ln\left(\frac{P\left(x'_j | c_j = 0\right) P\left(c_j = 0\right) P\left(x'_j\right)}{P\left(x'_j | c_j = 1\right) P\left(c_j = 1\right) P\left(x'_j\right)}\right) \tag{2.20}$$

Considering that the binary source is identically distributed, i.e, $P(c_j = 0) = P(c_j = 1)$, this LLR is reduced to

$$\mathrm{LLR}(P^p_j) = \ln\left(\frac{P(x'_j | c_j = 0)}{P(x'_j | c_j = 1)}\right) \tag{2.21}$$

where c_j is the bit originally transmitted by the source (before being corrupted). For real channel outputs, x'_j is replaced by the received real symbol $\mathbf{r}$ and c_j is replaced by the original transmitted symbol $\mathbf{s}_k$, as we will see later.

A BSC model is graphically illustrated in Figure 2.2, also recognised as an equivalent model to an AWGN channel with quantised output. We denote p as the probability of crossover. The event of crossover occurs when the decoder receives a flipped version of the bit originally

sent by the source. Therefore, the probability of no flipping is given by $1 - p$. According to Equation (2.21), the LLR of a priori probability for this binary channel model is given by

$$\mathrm{LLR}(P_j^p) = \begin{cases} \ln\left(\frac{p}{1-p}\right) & \text{if } r_j = 1 \\ \ln\left(\frac{1-p}{p}\right) & \text{if } r_j = 0 \end{cases} \tag{2.22}$$

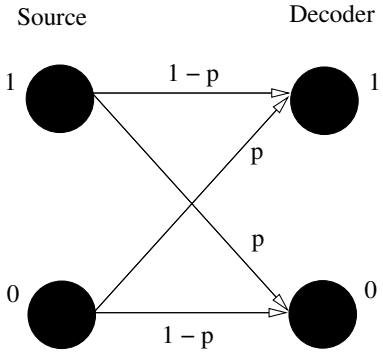

Figure 2.2: Binary Symmetric Channel model

Now considering the AWGN model and a BPSK baseband signal, the received vector r is modelled as $r = s + n$. The signal $s \in \mathbb{R}$ is the BPSK modulated symbol, which is defined as $s = -1$ for $c_j = 0$ and $s = +1$ for $c_j = 1$. The signal, n, is the channel noise, which corresponds to a real random Gaussian variable $n \sim \mathcal{N}(0, \sigma^2)$.

From the general definition of the a priori probability in Equation (2.21), we can derive an equivalent expression of this metric assuming the AWGN and a BPSK modulated real signal s. The probability density function of the Gaussian random received variable $r \in \mathbb{R}$ conditioned to the symbol $s_i,\ i \in \{0, 1\}$ is

$$P(r \mid s_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r-s_i)^2}{2\sigma^2}} \tag{2.23}$$

The final a priori LLR for any bit j from this constellation is therefore given by

$$\mathrm{LLR}(P_j^p) = \ln\left(\frac{e^{\frac{-(r+1)^2}{2\sigma^2}}}{e^{\frac{-(r-1)^2}{2\sigma^2}}}\right) = \frac{-(r+1)^2 + (r-1)^2}{2\sigma^2} \tag{2.24}$$

$$\mathrm{LLR}(P_j^p) = -\frac{2r}{\sigma^2} \tag{2.25}$$

The sum-product decoding is presented in Algorithm (2). There are three required inputs: the log-likelihood ratio of the a priori probabilities, $\mathrm{LLR}(P_j^p)$, the parity check matrix $\mathbf{H}$ and the maximum number of iterations allowed for the decoding process, I_{MAX}.

Example 2.1.2: Sum-Product

This example presents an illustrative description of the the sum-product algorithm. We use the same parity check matrix from Example (2.1.1). Also the same assumptions from that example with respect to $\mathbf{H}$ can be replicated in this.

This time the codeword $\mathbf{c} = \begin{pmatrix} 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0 \end{pmatrix}$ is sent from a QPSK mapper, whose modulated symbols $\mathbf{s} \in \{-\frac{\sqrt{2}}{2} - \mathrm{i}\frac{\sqrt{2}}{2},\ -\frac{\sqrt{2}}{2} + \mathrm{i}\frac{\sqrt{2}}{2},\ \frac{\sqrt{2}}{2} - \mathrm{i}\frac{\sqrt{2}}{2},\ \frac{\sqrt{2}}{2} + \mathrm{i}\frac{\sqrt{2}}{2}\}$ are associated to the binary set of strings $\mathsf{b}_1\mathsf{b}_0 = \begin{pmatrix} 00\ 01\ 10\ 11 \end{pmatrix}$, in this order. For simplicity, lets assume the QPSK modulation as a BPSK constellation per dimension so that Equation (2.25) can be applied independently over the real and the imaginary part of the received symbol $\mathbf{r}$. Thus, the a priori LLR message can be estimated as

$$\mathrm{LLR}(\mathsf{P}_1^{\mathsf{p}}) = \frac{2\,\mathrm{Re}\{\mathbf{r}\}}{\sigma^2} \tag{2.26}$$

$$\mathrm{LLR}(\mathsf{P}_0^{\mathsf{p}}) = \frac{2\,\mathrm{Im}\{\mathbf{r}\}}{\sigma^2} \tag{2.27}$$

Chapter 3 will be dedicated to present in details the other methods of estimation of these LLRs at higher order modulations.

In this example, consider that the baseband signal is corrupted by AWGN with $\frac{\mathsf{E_b}}{\mathsf{N_0}}$ configured to 0dB. Assume that the transmitter pulse shape filter is a normalised RRC filter, i.e., $\frac{1}{\mathsf{L}}\sum_{\mathsf{i}=1}^{\mathsf{L}} \mathsf{g}_\mathsf{i}^2 = 1$. From Equation (1.21) we obtain

$$\mathsf{E_b} = \frac{\mathsf{E}[|\mathsf{s_i}|^2]\frac{1}{\mathsf{L}}\sum_{\mathsf{i}=1}^{\mathsf{L}} \mathsf{g}_\mathsf{i}^2}{\mathsf{br}} = 1 \tag{2.28}$$

$$\sigma^2 = \frac{\mathsf{N_0}}{2} = \frac{\mathsf{E_b}}{2\frac{\mathsf{E_b}}{\mathsf{N_0}}} = \frac{1}{2} \tag{2.29}$$

White Gaussian noise can be generated using *randn* function in Matlab. This function generates a random sequence λ that follows a Gaussian distribution $\lambda \sim \mathcal{N}(0, 1)$. For adjusting the variance of λ to σ^2, we multiply the random vector by the corresponding standard deviation. Thus the complex received noisy signal with $\frac{\mathsf{E_b}}{\mathsf{N_0}} = 0\mathrm{dB}$ is given by

$$\mathbf{r} = \mathsf{s_i} + \sqrt{\sigma^2}\lambda + \mathrm{i}\sqrt{\sigma^2}\lambda \tag{2.30}$$

After adding noise, consider that we have the following vector at the channel output

$$\mathbf{r} = \begin{pmatrix} -0.8407 + 1.1990\mathrm{i} & 0.9945 - 0.2480\mathrm{i} & 0.9824 - 1.2560\mathrm{i} & -0.8789 + 1.5415\mathrm{i} & -1.1730 - 0.5069\mathrm{i} \end{pmatrix}$$

The standard sum-product algorithm begins its first iteration so that the obtained matrix of bit to check-node messages, $\mathbf{M}$, initialised with the values of $\mathrm{LLR}(\mathsf{P}_\mathsf{j}^{\mathsf{p}})$, is

$$\mathbf{M} = \begin{pmatrix} -3.3629 & . & 3.2883 & . & . & . & . & 7.1244 & -6.1135 & . \\ . & 5.7682 & . & . & . & . & -2.8092 & . & -5.7357 & . \\ . & . & 7.0597 & -5.0702 & . & . & -7.5653 & . & . & 1.3139 \\ . & 8.0432 & . & 0.7286 & . & -3.1906 & . & 8.9393 & . & . \\ . & . & . & . & 3.9295 & -5.9923 & . & . & -10.8556 & -1.1403 \end{pmatrix}$$

At the same iteration, the check to bit-node messages, E, are evaluated from Equation (2.16). The resulting matrix is

$$\mathsf{E} = \begin{pmatrix} -3.5071 & . & 3.0816 & . & . & . & . & 2.7732 & -2.8929 & . \\ . & 3.2471 & . & . & . & . & -4.0496 & . & -3.2707 & . \\ . & . & -0.6898 & 1.7206 & . & . & 0.7064 & . & . & 0.8871 \\ . & 0.9721 & . & -4.0782 & . & -0.9683 & . & 0.9583 & . & . \\ . & . & . & . & -1.9165 & 1.8333 & . & . & 1.8492 & 3.3414 \end{pmatrix}$$

The final a posteriori probability LLR results is

$$\mathsf{L} = \begin{pmatrix} -6.8700 & 9.0154 & 6.3699 & -3.3496 & 2.0129 & -4.1590 & -6.8588 & 9.8976 & -9.0063 & 2.2010 \end{pmatrix}$$

and the binary decision at the end of this iteration, which is based on the sign of these LLRs, results in the following estimated codeword

$$\hat{\mathbf{c}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

For block codes, there are three general ways in which the decoding algorithm may terminate [37]:

- Decoder successful: The decoder has found a valid codeword $\hat{\mathbf{c}}$ equals to $\mathbf{c}$.
- Decoder error: The decoder has found a valid codeword $\hat{\mathbf{c}}$ that differs from $\mathbf{c}$.
- Decoder failure: The decoder was unable to find a valid codeword using the resources specified.

For the error and the failure cases, the decoder has been unable to find the correct sent codeword $\mathbf{c}$. The main difference between them is that decoder failures are detectable whereas decoder errors are not.

In this case, $\hat{\mathbf{c}}$ results in a null syndrome vector, i.e., at the end of this single iteration, $\hat{\mathbf{c}}$ converges to a valid codeword. But unfortunately this valid codeword does correspond to the exactly transmitted codeword $\mathbf{c} \in \mathcal{C}$.

If we had no successful decoding at the end of this iteration, the bit to check-node messages, $\mathbf{M}$, would be updated from Equation (2.19) and the routine would return to the calculation of a new matrix E. However, due to the decoder error, the sum-product algorithm will terminate at this current state of $\mathbf{M}$ and E.

Algorithm 2 - Sum-Product Decoding

Require: H
Require: I_{MAX}
Require: $\mathrm{LLR}(\mathbf{P}^p)$

$I_{iter} = 0$

for $j = 1 : n$ **do**
  for $i = 1 : m$ **do**
    $M_{i,j} = \mathrm{LLR}(P_j^p)$
  end for
end for

repeat

  for $i = 1 : m$ **do**
    for $j \in \mathcal{N}_i$ **do**
      $E_{i,j} = \ln\left(\frac{1+\prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right)}{1-\prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right)}\right)$ ▷ Calculate extrinsic messages
    end for
  end for

  for $j = 1 : n$ **do**
    $L_j = \sum_{i \in \mathcal{M}_j} E_{i,j} + \mathrm{LLR}(P_j^p)$ ▷ Calculate A Posteriori Probability LLR
    $\hat{c}_j = \begin{cases} 1 & , \ L_j \leqslant 0 \\ 0 & , \ L_j > 0 \end{cases}$ ▷ Decode the codeword based on the A Posteriori Probability LLR
  end for

  for $i = 1 : m$ **do**
    $s_i = \sum_{j' \in \mathcal{N}_i} \oplus(\hat{c}_{j'})$ ▷ Calculate Syndrome vector
  end for

  if all $s_i = 0$ or $I_{iter} = I_{MAX}$ **then**
    FINISHED
  else
    for $j = 1 : n$ **do**
      for $i \in \mathcal{M}_j$ **do**
        $M_{i,j} = \sum_{i' \in \mathcal{M}_j, i' \neq i} E_{i',j} + \mathrm{LLR}(P_j^p)$ ▷ Calculate the Intrinsic Probability LLR
      end for
    end for
    $I_{iter} = I_{iter} + 1$ ▷ Increment iteration counter
  end if

until FINISHED

The calculation of extrinsic probability LLR message $E_{i,j}$, which measures the probability of the parity check constraint i to be satisfied given that the bit j is 1, can be modified in order to achieve lower complexity [11]. By taking the identity

$$2\tanh^{-1}(s) = \ln\left(\frac{1+a}{1-a}\right) \tag{2.31}$$

and assuming

$$a = \prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right) \tag{2.32}$$

we can write Equation (2.16) in a simpler manner as

$$E_{i,j} = 2\tanh^{-1} \prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{M_{i,j'}}{2}\right) \tag{2.33}$$

Moreover, the LLR message, $M_{i,j'}$, which regards to the intrinsic probability can also be reshaped in terms of two factors: the signal factor $\rho_{i,j'}$, which is given by

$$\rho_{i,j'} = \mathrm{sign}(M_{i,j'}) \tag{2.34}$$

and the magnitude factor $\tau_{i,j'}$, that is

$$\tau_{i,j'} = |M_{i,j'}| \tag{2.35}$$

Thus we can write

$$M_{i,j'} = \rho_{i,j'}\tau_{i,j'} \tag{2.36}$$

and Equation (2.33) becomes

$$E_{i,j} = \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'}\right) 2\tanh^{-1}\left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \tanh\left(\frac{\tau_{i,j'}}{2}\right)\right) \tag{2.37}$$

Additionally, the manipulation of Equation (2.37) yields

$$E_{i,j} = \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'}\right) 2\tanh^{-1}\left[\ln^{-1}\left(\sum_{j' \in \mathcal{N}(i), j' \neq j} \ln\left[\tanh\left(\frac{\tau_{i,j'}}{2}\right)\right]\right)\right] \tag{2.38}$$

As a benefit, the product from Equation (2.37) is replaced by a sum. Finally, the last step of this simplification relies on the following identity [34]

$$\phi(x) = -\ln\left(\tanh\left(\frac{x}{2}\right)\right) = \ln\frac{e^x+1}{e^x-1} \tag{2.39}$$

which has the property $\phi^{-1} = \phi$ since $\phi(\phi(x)) = x$.

From this result, Equation (2.16) becomes

$$E_{i,j} = \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'} \right) \phi \left(\sum_{j' \in \mathcal{N}(i), j' \neq j} \phi(\tau_{i,j'}) \right) \tag{2.40}$$

These small, yet effective simplifications presented so far can reduce the check-node computation complexity compared to the same costly step from the optimal sum-product algorithm. Furthermore, the simplified sum-product also uses to achieve great performance.

2.2 Sub-optimal Soft-decision Decoding Algorithms

The extrinsic LLR message calculation is usually the hardest step to implement an LDPC decoder. This step can be more efficiently implemented by using one of the sub-optimal algorithms which significantly reduce the hardware complexity at the cost of acceptable performance degradation. Oftenly, these algorithms correspond to enhanced versions of the so-called Min-Sum [5] algorithm. This section is dedicated to the Min-Sum, the Normalised Min-Sum [6] and the Offset Min-Sum algorithm [6], which are able to perform at least as well as the original sum-product algorithm when finite message precision is considered. We begin with a description of the classic Min-Sum algorithm.

2.2.1 Min-Sum and its variants algorithms

The substantial simplification employed in the min-sum algorithm arises from the remark that the product in Equation (2.16) is essentially dominated by the minimum term of $|M_{i,j}|$. In other words, the term corresponding to the smallest $\tau_{i,j'}$ well approximates the summation from Equation (2.40). The Min-Sum algorithm is nothing more than the standard log domain Sum-Product algorithm (SPA) with the extrinsic probability LLR calculation update step replaced by

$$E_{i,j} = \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'} \right) \min_{j' \in \mathcal{N}(i), j' \neq j} \tau_{i,j'} \tag{2.41}$$

For a certain pair $E^{I}_{i,j}$ and $E^{II}_{i,j}$ which are computed from Equations (2.16) and (2.41), respectively, it can be proved that $E^{I}_{i,j}$ and $E^{II}_{i,j}$ have the same sign and $E^{II}_{i,j} > E^{I}_{i,j}$ [38]. As a consequence of this, we note that this approximation introduces a sort of overestimation of the reliability magnitude. The approximation causes a small degradation on the decoding performance. In the Normalised Min-Sum algorithm, the output of the check-node is simply multiplied by a factor of normalisation, $0 < \alpha < 1$, as shown in the next equation. This small change compensates the

overestimation introduced by the original min-sum approximation.

$$E_{i,j} = \alpha \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'} \right) \min_{j' \in \mathcal{N}(i), j' \neq j} (\tau_{i,j'}) \tag{2.42}$$

A second alternative simplification can also reduce the overestimation of the outgoing extrinsic message by simply adding an offset constant. The Offset Min-Sum algorithm is implemented through the replacement of Equation (2.41) by

$$E_{i,j} = \left(\prod_{j' \in \mathcal{N}(i), j' \neq j} \rho_{i,j'} \right) \max \left(\min_{j' \in \mathcal{N}(i), j' \neq j} (\tau_{i,j'}) - \beta,\ 0 \right) \tag{2.43}$$

Figure 2.3 depicts the comparative performance among the Sum-Product algorithm (SPA), the Min-Sum (MS) and its variant algorithms: the Normalised Min-Sum and the Offset Min-Sum. These simulations were evaluated from DVB-S2 Matlab model, described in Chapter 1 of this thesis, configured as short FECFRAME transmission ($N = 16200$) under QPSK scheme, LDPC code rate $r = 3/4$ and maximum number of iterations limited to 50. Offset constant and normalisation constant were fixed to $\beta = 0.14$ and $\alpha = 0.57$ for the Offset MS and Normalised MS, respectively. The bit-error rate is computed at the output of LDPC decoder and the received signal at the channel output is corrupted by additive white Gaussian noise. As shown in this figure, the SPA achieves better performance than the other algorithms. Offset MS offers the second best BER performance, but as we know that it holds a lower complexity than SPA, one can say this algorithm provides the best trade-off between performance and complexity.

Defining $\eta = br$ as the spectral efficiency or information rate of the system, for reliable transmissions in our system configuration, we must have

$$\frac{E_b}{N_0}_{\text{SHANNON}} \geqslant 10 \log_{10} \left(\frac{2^\eta - 1}{\eta} \right) = 0.86\text{dB} \tag{2.44}$$

where $\eta = br$ is the product between the number of bits per symbol (2 bits under QPSK scheme) and r is the code rate. Reliable communication in the information-theoretic context means that error probability tends to zero as the codeword lengths get large. A practical system is said to be reliable if it operates at some desired non-zero, but very small error probability level. DVB-S2 standard denotes this practical state of small error probability as Quasi Error Free (QEF) condition, achieved by computer simulation limited to 50 LDPC decoding iterations, perfect carrier and synchronisation recovery, no phase noise and AWGN channel. At our current system configuration, $(E_s/N_0)_{\text{QEF}} \geqslant 4.03\text{dB}$ is suggested on the expected standard error performance. Thus we have,

$$\frac{E_b}{N_0}_{\text{QEF}} = \frac{E_s}{N_0}_{\text{QEF}} - 10 \log_{10} (\eta) \geqslant 4.03\text{dB} - 10 \log_{10} \left(2 \times \frac{3}{4} \right) = 2.2691\text{dB} \tag{2.45}$$

For short FECFRAMEs an additional degradation of 0.2dB to 0.3dB has to be taken into account [3]. Therefore, adopting $(E_s/N_0)_{\text{QEF}} \geqslant 4.03 + 0.25\text{dB}$ for the short FECFRAME transmission, we obtain

$$\frac{E_b}{N_0}_{\text{QEF}^*} \geqslant 2.5191\text{dB} \tag{2.46}$$

which is perfectly in accordance to our results in Figure 2.3.

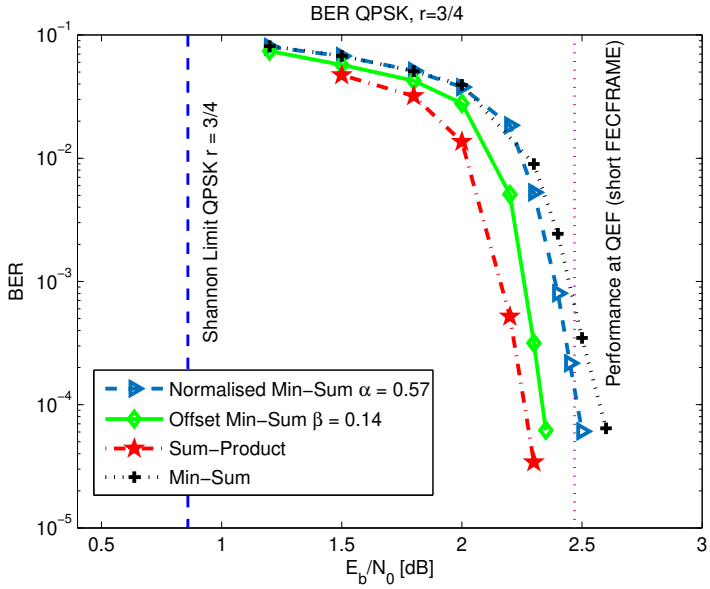

Figure 2.3: Bit-error rate performance of LDPC decoding algorithms for short FECFRAME transmission and code rate $\mathrm{r} = 3/4$ under QPSK modulation.

Chapter 3

Simplified Soft-Demappers for Higher-Order Modulation Schemes

Chapter 2 presented the soft-decision decoders as a class of algorithms that outperform the family of hard-decision decoders. As remarked before, for soft-decision decoding algorithms, the reliability metrics are calculated based on the channel outputs and also on the channel estimated parameters (e.g., for AWGN, the SNR). These decoders employ the reliability measures to gain knowledge over the transmitted codewords. Nevertheless, the performance power from soft-decision comes at the expense of higher complexity. As opposed to hard-decision, Log-Likelihood Ratios at the channel output have been shown to be very efficient metrics for soft-decoding of many powerful codes such as the turbo codes [39] and Low-Density Parity check codes [1]. The LLRs offer practical advantages in many steps along decoding process, as we will see next.

Regarding the LLR derivation for AWGN, we will see that, as the modulation order increases, the calculations becomes more complex. This increase in terms of complexity may cause decoding delays, extra power dissipation at the receiver as well as increasing the required hardware area. But for high speed wireless transmissions, the decoder might not be able to support these effects. The LLR approximations for DVB-S2 standard will be introduced in this Chapter as an efficient implementation of this sort of decoder. First we will present an *Ad Hoc* solution, based on intuitive constellation splitting for which we derive LLR expressions expressed in terms of constant values. Then we present a second solution based on the analytical derivation of constellation boundaries using the *Voronoi* [40] criterion in order to find an optimal polyhedral decomposition of sectors.

3.1 Introduction to the *Ad Hoc* Simplification

The calculation of Log-Likelihood Ratios at the channel output may impose great demand of memory and hardware area, especially for high-order modulations. As a part of this research, one introduces a new possible approach for the LLR approximation under AWGN. This is based on the splitting of the original constellation into smaller sectors where our assumptions are valid.

Each new sector has a less complex configuration in which it is possible to take into account only two symbols from the constellation besides the received one. The proposed solution is applied under 8-PSK and 16-APSK constellations adopted in the DVB-S2 standard.

As already mentioned before, LDPC codes provide excellent bit error rates in AWGN channels [1]. Nowadays design techniques for LDPC generation enable the construction of codes which approach the Shannon's capacity [41] to within hundredths of a decibel. However, the channel codes such as LDPC require a soft-demapper [8] for the calculation of a soft-metric for each bit from the constellation. The LLRs offer practical advantages such as numerical stability [42] (low dynamic range) as well as the simplification of many decoding algorithms. They are defined as the logarithmic ratio between the probability of the bit to be 0 ($b_j = 0$) and the probability of the bit to be 1 ($b_j = 1$) conditioned to a given received complex symbol, $\mathbf{r}$, i.e,

$$\begin{aligned} \mathrm{LLR}(b_j) &\triangleq \ln\left(\frac{P(b_j = 0 \mid \mathbf{r})}{P(b_j = 1 \mid \mathbf{r})}\right) && (3.1)\\ &= \ln\left(\frac{P(\mathbf{r} \mid b_j = 0)P(b_j = 0)/P(\mathbf{r})}{P(\mathbf{r} \mid b_j = 1)P(b_j = 1)/P(\mathbf{r})}\right) && (3.2)\\ &= \ln\left(\frac{P(\mathbf{r} \mid b_j = 0)}{P(\mathbf{r} \mid b_j = 1)}\right) && (3.3) \end{aligned}$$

Note that Equation (3.3) is obtained as a consequence of the Bayes' rule application and the assumption that we have an equiprobable binary source.

The event $b_j = 0$ is a consequence of the sum of disjoint events from the transmission of any symbol, $\mathbf{s}_i$, from the constellation whose $b_j = 0$. Due to this, through the marginalisation rule, we may obtain the channel LLR in terms of the received and transmitted symbols associated to $b_j = 0$. Thus, the exact calculation of the channel LLR for an M-ary complex modulation under AWGN is defined as

$$\mathrm{LLR}(b_j) = \ln\left(\frac{\sum_{i\in\{\mathcal{A}_{b_j=0}\}} V_i}{\sum_{i\in\{\mathcal{A}_{b_j=1}\}} V_i}\right) \tag{3.4}$$

where $\mathcal{A}$ is the alphabet of symbols, $\mathcal{A}_{b_j=x}$ is the set of symbols which present the j-th bit equal to $x \in \{0,\ 1\}$ and V_i is the Gaussian probability density function of the received complex symbol given that s_i was transmitted, which in this case yields

$$V_i = \frac{1}{2\pi\sigma^2} e^{\frac{-|\mathbf{r}-\mathbf{s}_i|^2}{2\sigma^2}} \quad i = 0, 1 \cdots M-1 \tag{3.5}$$

In Equation (3.5), σ^2 is the variance per dimension, $\mathbf{r}$ is the received symbol and $\mathbf{s}_i$ is the i-th symbol from the constellation.

Equations (3.4) and (3.5) require a high computational cost and hardware complexity exemplified by the large number of multiplications, exponential and logarithm operations in addition to the memory requirements, mainly observed in high order modulations. In this sense, the Max-Log approach [28] brings a significant reduction in the amount of operations through the

elimination of the exponentials and logarithm computations. According to this method, the LLR calculations rely on the Jacobian Logarithm [28] identity, which is approximated by

$$\ln(e^{x_1} + e^{x_2}) \approx \max(x_1, x_2) \tag{3.6}$$

Appendix B provides the Jacobian identity as well as its proof.

Next subsections briefly present a new alternative [29] approximation to the computation of the LLRs. The presented simplification is tailor-made to the constellations defined in the DVB-S2 [3]. Our main concern consists in avoiding the unnecessary operations involving multiplications, additions and the comparisons. Subsection 3.1.1 analyses the Max-Log approach for the 8-PSK scheme and then introduces the proposed *Ad Hoc* approximation, the involved assumptions and the final LLR expressions. In subsection 3.1.2 an extension of this approximation is presented under 16-APSK scheme. The numerical results, simulations of performance and gains in terms of complexity reduction will be presented in Chapter 4.

3.1.1 Channel Output LLR via *Ad Hoc* Approximation under 8-PSK

From the Max-Log approximation it is possible to achieve a substantial reduction in the amount of operations required by the soft-decision decoders. To see this, we begin by detailing the Max-Log approximation under 8-PSK constellation of DVB-S2, shown in Figure 3.1. Note that, for instance, symbols $\mathbf{s}_0$, $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$ all correspond to the most significant bit $\mathtt{b}_2 = 0$, while symbols $\mathbf{s}_4$, $\mathbf{s}_5$, $\mathbf{s}_6$ and $\mathbf{s}_7$ correspond to $\mathtt{b}_2 = 1$. As a consequence of this remark and applying Equation (3.6) in (3.4), the Max-Log approximation yields

$$\mathrm{LLR}(\mathtt{b}_2) \approx \max(\mathtt{D}_0, \mathtt{D}_1, \mathtt{D}_2, \mathtt{D}_3) - \max(\mathtt{D}_4, \mathtt{D}_5, \mathtt{D}_6, \mathtt{D}_7) \tag{3.7}$$

Following the same reasoning, the LLRs corresponding to the remaining bits are derived as

$$\mathrm{LLR}(\mathtt{b}_1) \approx \max(\mathtt{D}_0, \mathtt{D}_1, \mathtt{D}_4, \mathtt{D}_5) - \max(\mathtt{D}_2, \mathtt{D}_3, \mathtt{D}_6, \mathtt{D}_7) \tag{3.8}$$

$$\mathrm{LLR}(\mathtt{b}_0) \approx \max(\mathtt{D}_0, \mathtt{D}_2, \mathtt{D}_4, \mathtt{D}_6) - \max(\mathtt{D}_1, \mathtt{D}_3, \mathtt{D}_5, \mathtt{D}_7) \tag{3.9}$$

where

$$\mathtt{D}_\mathtt{i} = -\frac{|\mathbf{r} - \mathbf{s}_\mathtt{i}|^2}{2\sigma^2} \quad \mathtt{i} = 0, 1 \cdots 7 \tag{3.10}$$

$\mathtt{b}_2$ is the Most Significant Bit (MSB) and $\mathtt{b}_0$ is the Least Significant Bit (LSB).

The proposed simplification stems from the observation that each LLR is calculated from four Euclidean distances and six comparisons (three for each max function) when, in fact, both terms from the max function are always obtained from the symbol $\mathbf{s}_\mathtt{i}$ that is the closest to the received symbol $\mathbf{r}$. Proceeding from this point, one may approximate the LLR by designing decision areas for each bit, neglecting the probability that the noise corrupts the symbol such that it reaches another sector outside the original one. Thus, each sector will contain only two symbols from which the LLRs of a corresponding bit will be approximated. The sector to be employed in this approach is that in which $\mathbf{r}$ resides. In other words, the simplification introduced by

Max-Log approach uses all the constellation symbols for the LLR calculations. Nonetheless, as we are aiming at reduce the number of operations and their complexity, the proposed method splits the constellation into smaller regions of decision such that the number of symbols to be considered in each sector is reduced to only two of them.

In order to obtain appropriate decision areas, a rotation $\phi = -\pi/8$ is applied over the original defined 8-PSK constellation. This transformation will lead to convenient sector bounds, i.e., one sector per quadrant. As we can see in Figure 3.11, each sector contains a symbol in which $\mathtt{b}_2 = 1$ and a symbol in which $\mathtt{b}_2 = 0$. We propose to approximate the LLR in each sector through the two corresponding symbols

$$\mathrm{LLR}(\mathtt{b}_2) \approx \begin{cases} \mathrm{D}_0 - \mathrm{D}_4; & \mathrm{I} > 0,\ \mathrm{Q} > 0 \\ \mathrm{D}_1 - \mathrm{D}_5; & \mathrm{I} > 0,\ \mathrm{Q} < 0 \\ \mathrm{D}_2 - \mathrm{D}_6; & \mathrm{I} < 0,\ \mathrm{Q} > 0 \\ \mathrm{D}_3 - \mathrm{D}_7; & \mathrm{I} < 0,\ \mathrm{Q} < 0 \end{cases} \tag{3.11}$$

where I (in-phase) is the real value of the rotated received symbol, $\mathbf{r}$, and Q (quadrature) is the imaginary value of this symbol.

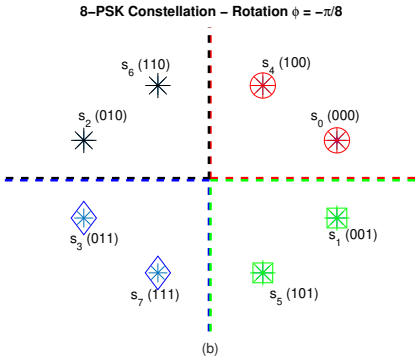

Figure 3.1: Decision areas for bit $\mathtt{b}_2$. Note that each symbol is represented as an ordered triplet of bits $(\mathtt{b}_2\mathtt{b}_1\mathtt{b}_0)$

The sectors splitting for the bit $\mathtt{b}_1$ is even simpler due to the geometrical symmetry of this constellation. The symbols in which $\mathtt{b}_1 = 0$ and those in which $\mathtt{b}_1 = 1$ are symmetrically located with respect to the vertical axis. Figure 3.2 shows the symbol partitions for which this approximation is valid. The LLR for bit $\mathtt{b}_1$ is approximatef by Equation (3.12).

$$\mathrm{LLR}(\mathtt{b}_1) \approx \begin{cases} \mathrm{D}_4 - \mathrm{D}_6; & \mathrm{Q} > 0,\ |\mathrm{Q}| > |\mathrm{I}| \\ \mathrm{D}_0 - \mathrm{D}_2; & \mathrm{Q} > 0,\ |\mathrm{I}| > |\mathrm{Q}| \\ \mathrm{D}_1 - \mathrm{D}_3; & \mathrm{Q} < 0,\ |\mathrm{I}| > |\mathrm{Q}| \\ \mathrm{D}_5 - \mathrm{D}_7; & \mathrm{Q} < 0,\ |\mathrm{Q}| > |\mathrm{I}| \end{cases} \tag{3.12}$$

Similarly, the symbols in which $\mathtt{b}_0 = 0$ and those in which $\mathtt{b}_0 = 1$ are symmetrically located with respect to the horizontal axis. Figure 3.3 depicts the symbol sectors for which this approx-

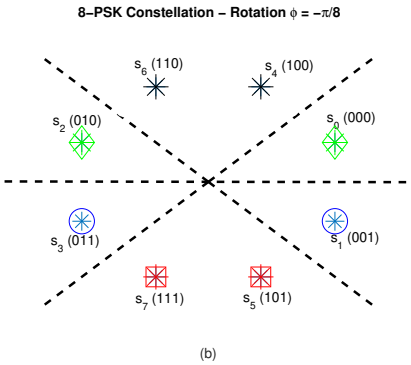

Figure 3.2: Decision areas for bit $\mathfrak{b}_1$.

imation yields valid results. The LLR approximation for this bit is given by Equation (3.13).

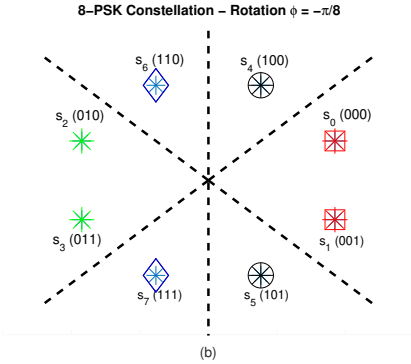

Figure 3.3: Decision areas for bit $\mathfrak{b}_0$ (Right - LSB)

$$\mathrm{LLR}(\mathfrak{b}_0) \approx \begin{cases} D_2 - D_3; & I < 0,\ |I| > |Q| \\ D_6 - D_7; & I < 0,\ |Q| > |I| \\ D_4 - D_5; & I > 0,\ |Q| > |I| \\ D_0 - D_1; & I > 0,\ |I| > |Q| \end{cases} \tag{3.13}$$

By writting the LLRs in each sector one can derive a general expression as a function of the real and imaginary parts of the complex received symbol, the variance per dimension and two constants μ_i and μ_q. These constants can be calculated offline and then plugged to the equation according to where $\mathbf{r}$ resides. The final obtained equation is

$$\mathrm{LLR}(\mathfrak{b}_j) \approx \frac{1}{\sigma^2}\left(I\mu_i + Q\mu_q\right) \tag{3.14}$$

where $\mu_i = I_{\mathcal{A}_{\mathfrak{b}_j=0}} - I_{\mathcal{A}_{\mathfrak{b}_j=1}}$ and $\mu_q = Q_{\mathcal{A}_{\mathfrak{b}_j=0}} - Q_{\mathcal{A}_{\mathfrak{b}_j=1}}$ are the constant values we are able to determine in advance, given the constellation symbols associated to each sector. The symbols $\mathbf{s}_{\mathcal{A}_{\mathfrak{b}_j=0}} = I_{\mathcal{A}_{\mathfrak{b}_j=0}} + iQ_{\mathcal{A}_{\mathfrak{b}_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{\mathfrak{b}_j=1}} = I_{S_{\mathcal{A}_{\mathfrak{b}_j=1}}} + iQ_{S_{\mathcal{A}_{\mathfrak{b}_j=1}}}$ are, respectively, the symbol within

the sector for which $b_j = 0$ and the symbol within the sector for which $b_j = 1$.

In the context of hardware implementations, the constants μ_i and μ_q can be easily calculated and stored in memories. Table 3.1 shows the obtained values, μ_i and μ_q, under 8-PSK constellation defined in DVB-S2.

Table 3.1: Constants for LLR estimation under 8-PSK constellation.

LLR	μ_i	μ_q	Bound
b_2	0.5412	-0.5412	$I > 0, Q > 0$
	0.5412	0.5412	$I > 0, Q < 0$
	-0.5412	-0.5412	$I < 0, Q > 0$
	-0.5412	0.5412	$I < 0, Q < 0$
b_1	0.7654	0	$Q > 0, \|Q\| > \|I\|$
	1.8478	0	$Q > 0, \|I\| > \|Q\|$
	1.8478	0	$Q < 0, \|I\| > \|Q\|$
	0.7654	0	$Q < 0, \|Q\| > \|I\|$
b_0	0	0.7654	$I < 0, \|I\| > \|Q\|$
	0	1.8478	$I < 0, \|Q\| > \|I\|$
	0	1.8478	$I > 0, \|Q\| > \|I\|$
	0	0.7654	$I > 0, \|I\| > \|Q\|$

3.1.2 Channel Output LLR via *Ad Hoc* Approximation under 16-APSK

The 16-APSK scheme counts on two additional aspects beyond those we presented on last subsection under a PSK constellation: the natural symmetry of the constellation symbols without the need of rotation, as opposed to the 8-PSK constellation, and the possibility of mapping any received symbol to the first quadrant. Due to this latter aspect, we are able to reduce the number of sectors to be considered, i.e., we can further reduce the efforts on the classification of $\mathbf{r}$ in the sector where it resides.

The main distinction between both schemes is that, in the PSK modulations, all the constellation symbols have the same energy or the same radius. Hence, for any sector where two symbols reside, $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$, the LLRs are approximated by

$$\text{LLR} \approx -\frac{(I - I_{\mathcal{A}_{b_j=0}})^2 + (Q - Q_{\mathcal{A}_{b_j=0}})^2 + (I - I_{\mathcal{A}_{b_j=1}})^2 + (Q - Q_{\mathcal{A}_{b_j=1}})^2}{2\sigma^2} \tag{3.15}$$

For 8-PSK, the terms $-(I^2_{\mathcal{A}_{b_j=0}} + Q^2_{\mathcal{A}_{b_j=0}}) + (I^2_{\mathcal{A}_{b_j=1}} + Q^2_{\mathcal{A}_{b_j=1}})$, and $-(I^2+Q^2)+(I^2+Q^2)$ in Equation (3.15) cancel each other such that all the quadratic terms in the LLR equation are removed. But the same is not true for the APSK constellations since the symbols $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$ might not have necessarily the same level of energy.

To overcome the amplitude variation and the resulting loss of performance due to ignore the quadratic terms, we suggest to create *Equivalent Received Symbols* that will be referred to as ERS from now on. They are defined by a pair of two real gains which either multiply the real and imaginary part of the received symbol or simply multiply the radius of one of the constellation rings, when the ERS is fixed. In both cases these gains represents a phase shift on the symbol which is being converted to an ERS. These gains are not necessarily equal to each other and they can be heuristically determined so that we can properly adjust the soft-decoder performance. In summary, we can have two different types of ERSs: those which are variant and therefore dependent on the received symbol through the pair of gains G_I and G_Q, i.e.,

$$\mathbf{r}_{\text{ERS}} = G_I I + iG_Q Q$$

and those that are simply given by the product between the gains and a fixed radius from an specific ring R_i, i.e.,

$$\mathbf{r}_{\text{ERS}} = G_I R_i + iG_Q R_i$$

where i corresponds to the ring index, $i \in \{0,\ 1,\ 2\}$. These indices are associated to the middle radius, the inner radius and outer radius from the constellation, in this order. The middle radius is not defined by the standard and does not contain any original constellation symbol. This simply represents an intermediate distance among the inner and the outer radius, both defined by DVB-S2.

An illustrative scheme of the adopted sectors for the LLR estimation of bit b_3 is shown in Figure 3.4. As early mentioned, in this 16-APSK constellation the approximation can be established by only taking the first quadrant, given the symmetry among the others. Each sector will have approximated LLRs based on two original constellation symbols or on the family of ERS symbols we create. In the same figure we refer to $\mathbf{s}_\xi$ as the fixed created ERS and $\mathbf{s}_\chi$ as the variant ERS associated to the received symbol. The approximations of $\text{LLR}(b_3)$ in each sector are derived as follows:

Region 1: Bounds: $|\mathbf{r}| < R_0,\ \pi/6 \leqslant \theta \leqslant \pi/2$

$$\begin{aligned}\text{LLR}(b_3) &\approx -\frac{|\mathbf{r_2} - \mathbf{s}_0|^2}{2\sigma^2} + \frac{|\mathbf{r_1} - \mathbf{s}_{12}|^2}{2\sigma^2} \\ &= \frac{(R_1^2 - R_2^2) + I_2 I_{s_0} + Q_2 Q_{s_0}}{\sigma^2} - \frac{I_1 I_{s_{12}} + Q_1 Q_{s_{12}}}{\sigma^2}\end{aligned}$$

As we can see in the calculation above, the approximate LLR is obtained by multiplying the real and imaginary parts of the received symbol by two constant values. As well as in the 8-PSK scheme, in hardware implementation context, all these constants can be stored in memories,

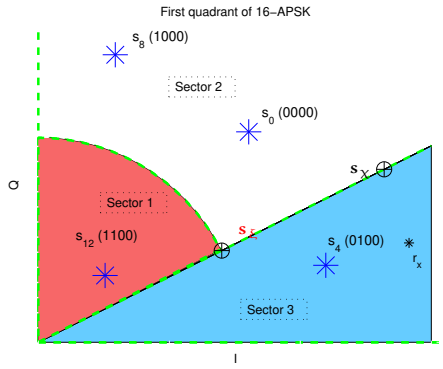

Figure 3.4: Decision areas for bit $\mathfrak{b}_3$.

making the process of calculation much less complex.

Region 2: Bounds: $|\mathbf{r}| \geqslant \mathsf{R}_0,\ \pi/6 \leqslant \theta \leqslant \pi/2$

$$\begin{aligned} \mathrm{LLR}(\mathfrak{b}_3) &\approx -\frac{|\mathbf{r}_2 - \mathbf{s}_0|^2}{2\sigma^2} + \frac{|\mathbf{r}_2 - \mathbf{s}_8|^2}{2\sigma^2} \\ &= \frac{\mathsf{I}_2\,(\mathsf{I}_{s_0} - \mathsf{I}_{s_8})}{\sigma^2} + \frac{\mathsf{Q}_2\,(\mathsf{Q}_{s_0} - \mathsf{Q}_{s_8})}{\sigma^2} \end{aligned}$$

Similarly to the sector 1, the approximation for the sector 2 may be obtained by simply storing a set of constants.

Region 3: Bounds: $0 < \theta \leqslant \pi/6$

This sector present a significant higher occurrence of bit errors than the other sectors due to sector boundary issues. Inside this sector, the closest symbol in which the bit $\mathfrak{b}_3$ has the value 1 may not have always the same radius neither the same phase than all the possible received symbols within this sector. The same occurs for the associated symbols in which bit $\mathfrak{b}_3$ is 0. As an attempt to overcome this limitation, the insertion of a fixed ERS, $\mathbf{s}_\xi = \mathsf{R}_0\cos(\pi/6) + \mathfrak{i}\mathsf{R}_0\sin(\pi/6)$ and a variant ERS $\mathbf{s}_\chi = |\mathbf{r}|\cos(\pi/6) + \mathfrak{i}|\mathbf{r}|\sin(\pi/6)$ is proposed. As a consequence of this, the approximate expression of $\mathrm{LLR}(\mathfrak{b}_3)$ in this sector is calculated as

$$\begin{aligned} \mathrm{LLR}(\mathfrak{b}_3) &\approx -\frac{|\mathbf{r}_0 - \mathbf{s}_\xi|^2}{2\sigma^2} + \frac{|\mathbf{r} - \mathbf{s}_\chi|^2}{2\sigma^2} \\ &= \frac{(|\mathbf{r}|^2 - \mathsf{R}_0^2) + \mathsf{I}_0\mathsf{I}_{s_\xi} + \mathsf{Q}_0\mathsf{Q}_{s_\xi}}{\sigma^2} - \frac{\mathsf{I}\mathsf{I}_{s_\chi} + \mathsf{Q}\mathsf{Q}_{s_\chi}}{\sigma^2} \end{aligned}$$

Similar remarks can be made over the corresponding sectors of the bit $\mathfrak{b}_2$. However, the ERS gains of $\mathbf{s}_\xi$ and $\mathbf{s}_\chi$ must be updated to $\mathsf{G}_\mathsf{I} = \cos(\pi/3)$ and $\mathsf{G}_\mathsf{Q} = \sin(\pi/3)$. The sector bounds of each sector for the approximation of $\mathrm{LLR}(\mathfrak{b}_2)$ are depicted in Figure 3.5.

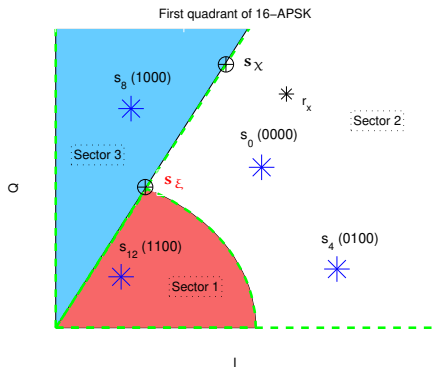

Figure 3.5: Decision areas for bit $\mathtt{b}_2$.

Region 1: Bounds: $|\mathbf{r}| < \mathrm{R}_0,\ 0 \leqslant \theta \leqslant \pi/3$

$$\begin{aligned} \mathrm{LLR}(\mathtt{b}_2) &\approx -\frac{|\mathbf{r_2} - \mathbf{s}_0|^2}{2\sigma^2} + \frac{|\mathbf{r_1} - \mathbf{s}_{12}|^2}{2\sigma^2} \\ &= \frac{(\mathrm{R}_1^2 - \mathrm{R}_2^2) + \mathrm{I}_2\mathrm{I}_{s_0} + \mathrm{Q}_2\mathrm{Q}_{s_0}}{\sigma^2} - \frac{\mathrm{I}_1\mathrm{I}_{s_{12}} + \mathrm{Q}_1\mathrm{Q}_{s_{12}}}{\sigma^2} \end{aligned}$$

Region 2: Bounds: $|\mathbf{r}| \geqslant \mathrm{R}_0,\ 0 \leqslant \theta \leqslant \pi/3$

$$\begin{aligned} \mathrm{LLR}(\mathtt{b}_2) &\approx -\frac{|\mathbf{r}_2 - \mathbf{s}_0|^2}{2\sigma^2} + \frac{|\mathbf{r}_2 - \mathbf{s}_4|^2}{2\sigma^2} \\ &= \frac{\mathrm{I}_2\left(\mathrm{I}_{s_0} - \mathrm{I}_{s_4}\right)}{\sigma^2} + \frac{\mathrm{Q}_2\left(\mathrm{Q}_{s_0} - \mathrm{Q}_{s_4}\right)}{\sigma^2} \end{aligned}$$

Region 3: Bounds: $\pi/3 < \theta \leqslant \pi/2$

$$\begin{aligned} \mathrm{LLR}(\mathtt{b}_2) &\approx -\frac{|\mathbf{r_O} - \mathbf{s}_\xi|^2}{2\sigma^2} + \frac{|\mathbf{r} - \mathbf{s}_\chi|^2}{2\sigma^2} \\ &= \frac{(|\mathbf{r}|^2 - \mathrm{R}_0^2) + \mathrm{I_O}\mathrm{I}_{s_\xi} + \mathrm{Q_O}\mathrm{Q}_{s_\xi}}{\sigma^2} - \frac{\mathrm{II}_{s_\chi} + \mathrm{QQ}_{s_\chi}}{\sigma^2} \end{aligned}$$

The LLRs for the two least significant bits of 16-APSK are obtained in a similar way to the bit $\mathtt{b}_1$ and bit $\mathtt{b}_0$ from 8-PSK constellation. Fortunately for bit $\mathtt{b}_1$ of 16-APSK, the symbols $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_j=1}}$ are symmetric with respect to the vertical axis, as illustrated in Figure 3.6. Hence, the LLR for bit $\mathtt{b}_1$ is given by

$$\begin{aligned} \mathrm{LLR}(\mathtt{b}_1) &\approx -\frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{\mathtt{b}_j=0}}|^2}{2\sigma^2} + \frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{\mathtt{b}_j=1}}|^2}{2\sigma^2} \\ &= \frac{2\mathrm{II}_{\mathrm{S}_{\mathcal{A}_{\mathtt{b}_j=0}}}}{2\sigma^2} \end{aligned}$$

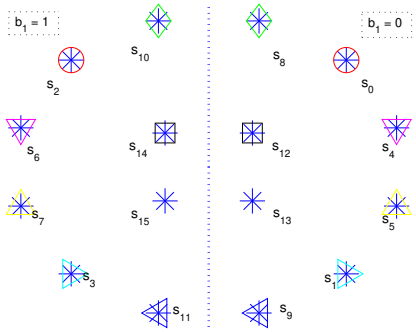

Figure 3.6: Geometrical configuration of the symbols in 16-APSK constellation as a reference to the bit $\mathfrak{b}_1$.

In order to avoid many sectors and as a consequence to reduce the efforts on the classification of the received symbol, we propose to approximate the LLR by a general expression assumed to be valid for any point of the complex plane and with no dependence on any constellation symbol. In other words,

$$\mathrm{LLR}(\mathfrak{b}_1) \approx \frac{\mathrm{I}}{2\sigma^2} \tag{3.16}$$

The same idea can be applied to the bit $\mathfrak{b}_0$ (LSB). But in this case, the symmetry occurs with respect to the horizontal axis, as we can note in Figure 3.7. Hence, the $\mathrm{LLR}(\mathfrak{b}_0)$ can be similarly approximated as

$$\mathrm{LLR}(\mathfrak{b}_0) \approx \frac{\mathrm{Q}}{2\sigma^2} \tag{3.17}$$

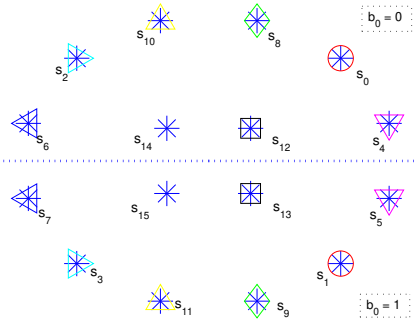

Figure 3.7: Geometrical configuration of the symbols in 16-APSK constellation as a reference to the bit $\mathfrak{b}_0$.

The results obtained from the *Ad Hoc* approximation method will be presented and discussed in Chapter 4. In the next section we will present a new method of channel output LLR estimation based on the Voronoi polyhedral decomposition.

3.2 Introduction to the Approximation based on the Voronoi Decomposition

In this section we exploit an alternative proposal for the step of constellation splitting. Unlike the *Ad Hoc* splitting proposal from Section 3.1, this method consists of adopting optimal and deterministic bound limits which arise from the Voronoi [43, 40] criterion for polyhedral decomposition. We aim to introduce the general structure for sector classification as well as its formal mathematical definition in the context of the Log-Likelihood Ratio simplified calculation. The resulting method can be applied to any constellation. Figure 3.8 illustrates, in advance, the results of this approach for the step of symbol's partitioning in the particular case of a rotated version of 32-APSK constellation from DVB-S2 standard.

The splitting depicted in Figure 3.8 is obtained from the minimisation of the distance between each constellation symbol or *seed* and a corresponding set of points that reside in a given region of the complex plane, $\mathcal{S}$ [8]. The seeds are specified by the coordinates of the original constellation symbols and each region of the complex plane correspond to a constellation sector that we aim to determine. As we can note in Figure 3.8, we are able to only take into account the seeds from one or two quadrants of the plane due to the opportune geometrical configuration of symbols in this constellation. This symmetry allows us to save unnecessary calculations in the sense that it reduces the final number of sectors to be considered. For each constellation symbol, $\mathbf{s}$, there exists a corresponding area containing all the symbols closer to this symbol than to any other one. Each one of these areas corresponds to a Voronoi cell.

The main idea behind the LLR simplification is similar to the employed in *Ad Hoc* method, i.e., we wish to estimate the reliability magnitudes through two constellation symbols only. However, in this new proposal we devote more attention to the partitioning algorithm. As a consequence of our simplification approach, the sectors might be determined through an optimal splitting criterion, which in this case must establish that the distance from each point which resides in the sector to the constellation symbol associated to this sector be necessarily smaller than its distance to any other constellation symbol. In other words, the received symbol $\mathbf{r}$ must be closer to a given seed, $\mathbf{s}$, than to any other seed, $\mathbf{s}_{\mathfrak{i}}$. The correspondent linear inequality system for an alphabet length M is given by

$$|\mathbf{r}-\mathbf{s}|_2^2 \leqslant |\mathbf{r}-\mathbf{s}_{\mathfrak{i}}|_2^2 \quad \mathfrak{i} \in \{0, 1, \cdots, \mathsf{M}-1\} \tag{3.18}$$

or

$$\mathbf{r}(\mathbf{s}_{\mathfrak{i}}-\mathbf{s})^{\mathsf{T}} \leqslant \frac{|\mathbf{s}_{\mathfrak{i}}|^2-|\mathbf{s}|^2}{2} \tag{3.19}$$

We have used Equation (3.18) to figure out a linear approach for classifying the received symbols into the correct sector since the bounds may not always be easily specified through angles and radius such as exploited in *Ad Hoc* proposal. Keeping in mind the criterion and the approach of our LLR estimation method based on two distinct constellation symbols, the solution for the sector boundaries emerges from the intersection between two linear inequality

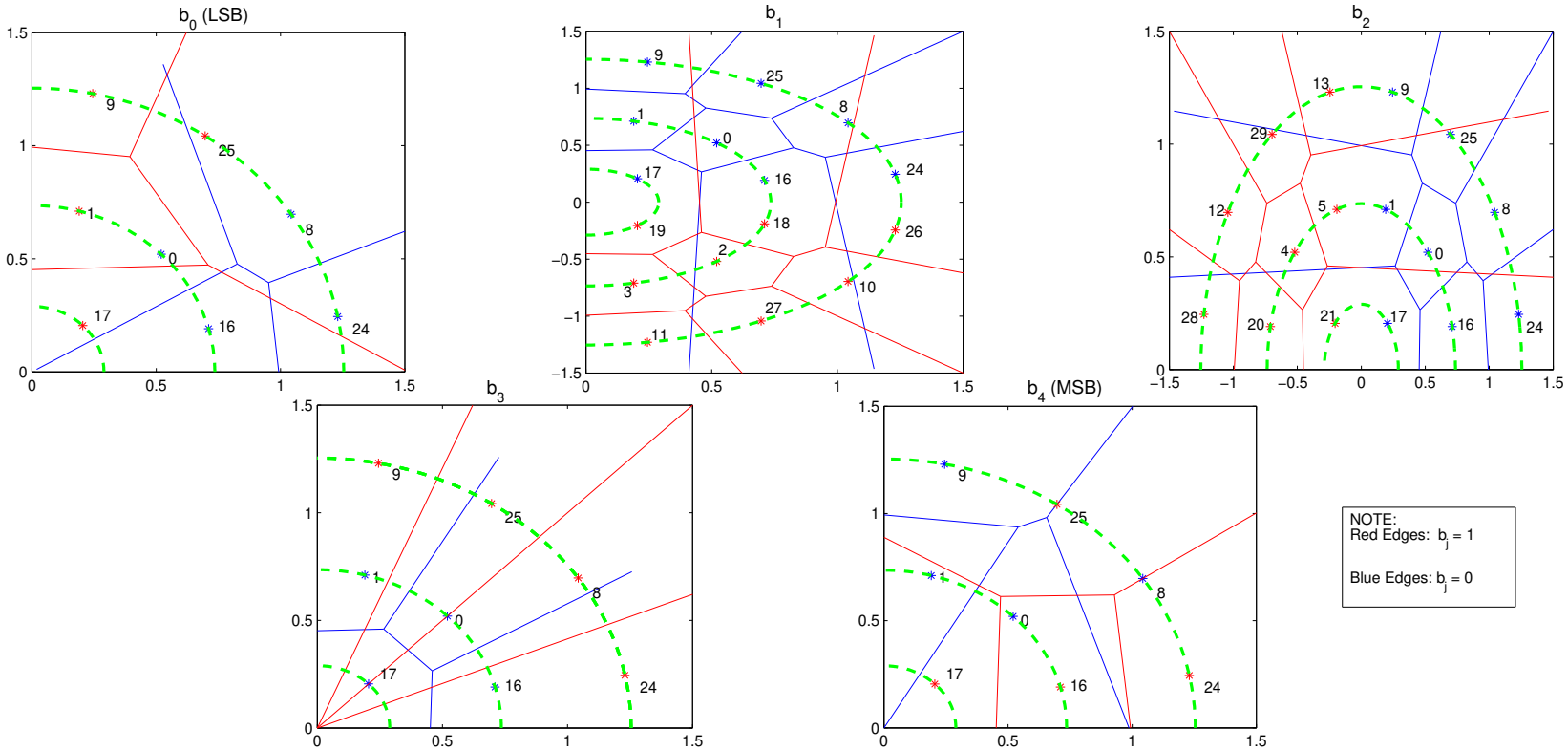

Figure 3.8: Results from the original Voronoi decomposition applied for 32-APSK constellation.

systems that are given by Equation (3.19). The first system must be defined for the set of the constellation symbols $\mathcal{A}_{b_j=0}$, where the bit of interest, b_i, is zero. The second system is defined in the set of symbols $\mathcal{A}_{b_j=1}$, where the bit b_i is one. For each received symbol we obtain a pair of associated seeds, $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$, being the first one associated with a symbol where $b_i = 0$ and the second one associated with a symbol where $b_i = 1$. As remarked earlier, the corresponding LLR approximation is obtained by taking into account these two symbols.

We can obtain a general inequality expression which allows us to define both associated seeds for any received symbols $\mathbf{r}$. Our aim is to find these two symbols for which all the respective inequalities are satisfied. From the established optimal criterion and the derivation from Equation (3.19), the classification is simply reduced to the accomplishment of the following linear inequality system, defined for each set of symbols

$$\mathbf{rA} - \mathbf{B} \leqslant \mathbf{0} \tag{3.20}$$

where $\mathbf{r} = (\mathrm{I} \quad \mathrm{Q})$ is the vector of the received symbol. The matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained for each seed in the corresponding set of symbols. For the system which represents $b_i = 0$, we have

$$\mathbf{A_j} = \begin{pmatrix} \mathrm{I}^0_{\mathcal{A}_{b_i=0}} - \mathrm{I}^j_{\mathcal{A}_{b_i=0}} & \cdots & \mathrm{I}^j_{\mathcal{A}_{b_i=0}} - \mathrm{I}^j_{\mathcal{A}_{b_i=0}} & \cdots & \mathrm{I}^n_{\mathcal{A}_{b_i=0}} - \mathrm{I}^j_{\mathcal{A}_{b_i=0}} \\ \mathrm{Q}^0_{\mathcal{A}_{b_i=0}} - \mathrm{Q}^j_{\mathcal{A}_{b_i=0}} & \cdots & \mathrm{Q}^j_{\mathcal{A}_{b_i=0}} - \mathrm{Q}^j_{\mathcal{A}_{b_i=0}} & \cdots & \mathrm{Q}^n_{\mathcal{A}_{b_i=0}} - \mathrm{Q}^j_{\mathcal{A}_{b_i=0}} \end{pmatrix}$$

$$\mathbf{B_j} = \begin{pmatrix} \frac{|\mathbf{s}^0_{\mathcal{A}_{b_i=0}}|^2 - |\mathbf{s}^j_{\mathcal{A}_{b_i=0}}|^2}{2} & \cdots & \frac{|\mathbf{s}^j_{\mathcal{A}_{b_i=0}}|^2 - |\mathbf{s}^j_{\mathcal{A}_{b_i=0}}|^2}{2} & \cdots & \frac{|\mathbf{s}^n_{\mathcal{A}_{b_i=0}}|^2 - |\mathbf{s}^j_{\mathcal{A}_{b_i=0}}|^2}{2} \end{pmatrix}$$

where j is the superscript index which denotes the constellation symbol in the set $\mathcal{A}_{b_i=0}$ (constellation symbols for which $b_i = 0$) and n is the index of the last symbol from the set $\mathcal{A}_{b_i=0}$. Note that $\mathbf{A_j}$ and $\mathbf{B_j}$ delimit the edges for the sector containing the constellation symbol $\mathbf{s}_j = (\mathrm{I}^j_{\mathcal{A}_{b_i=0}} \quad \mathrm{Q}^j_{\mathcal{A}_{b_i=0}})$. Similarly, the system which represent $b_i = 1$ is obtained by

$$\mathbf{A_k} = \begin{pmatrix} \mathrm{I}^0_{\mathcal{A}_{b_i=1}} - \mathrm{I}^k_{\mathcal{A}_{b_i=1}} \cdots \mathrm{I}^k_{\mathcal{A}_{b_i=1}} - \mathrm{I}^k_{\mathcal{A}_{b_i=1}} \cdots \mathrm{I}^m_{\mathcal{A}_{b_i=1}} - \mathrm{I}^k_{\mathcal{A}_{b_i=1}} \\ \mathrm{Q}^0_{\mathcal{A}_{b_i=1}} - \mathrm{Q}^k_{\mathcal{A}_{b_i=1}} \cdots \mathrm{Q}^k_{\mathcal{A}_{b_i=1}} - \mathrm{Q}^k_{\mathcal{A}_{b_i=1}} \cdots \mathrm{Q}^m_{\mathcal{A}_{b_i=1}} - \mathrm{Q}^k_{\mathcal{A}_{b_i=1}} \end{pmatrix}$$

$$\mathbf{B_k} = \begin{pmatrix} \frac{|\mathbf{s}^0_{\mathcal{A}_{b_i=1}}|^2 - |\mathbf{s}^k_{\mathcal{A}_{b_i=1}}|^2}{2} \cdots \frac{|\mathbf{s}^k_{\mathcal{A}_{b_i=1}}|^2 - |\mathbf{s}^k_{\mathcal{A}_{b_i=1}}|^2}{2} \cdots \frac{|\mathbf{s}^m_{\mathcal{A}_{b_i=1}}|^2 - |\mathbf{s}^k_{\mathcal{A}_{b_i=1}}|^2}{2} \end{pmatrix}$$

where k is the superscript index for the constellation symbols in the set $\mathcal{A}_{b_i=1}$ (which includes the constellation symbols for which $b_i = 1$) and m is the index of last symbol from the set $\mathcal{A}_{b_i=1}$. Note that $\mathbf{A_k}$ and $\mathbf{B_k}$ delimit the edges for the sector containing the constellation symbol $\mathbf{s}_k = (\mathrm{I}^k_{\mathcal{A}_{b_i=1}} \quad \mathrm{Q}^k_{\mathcal{A}_{b_i=1}})$.

The employed linear systems might be implemented in hardware by storing M matrices, $\mathbf{A}$, and M vectors, $\mathbf{B}$, without the need of computing either the angles or the quadratic norm of the received symbols such as in the *Ad Hoc* proposal. Therefore, obtaining the pairs of constellations symbols $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$ that will be used on the LLR approximation for each received symbols is reduced to verify if all the inequalities from Equation (3.20) are satisfied. As a consequence, we avoid the use of the most obvious approach, i.e, the calculation of the minimum Euclidean distances from the received symbol to each one of the constellation symbols. Besides, once a constraint of any linear inequality system assumes a positive value, the checking process can be terminated in advance and we can discard the evaluated symbol as a possible seed associated to the received symbol. In other words, the number of operations involved in the process of obtaining each pair of seeds can be further reduced, since one single unsatisfied inequality is sufficient to terminate the evaluation of the corresponding constellation symbol and to discard it as one of the seeds associated to $\mathbf{r}$.

The original edges that arise from the Voronoi decomposition might be a little bit modified in order to make even less complex the process of verifying the pairs of seeds associated to each received symbol. Next subsection will present these small approximations by taking 16-APSK scheme as a toy example. Afterwards we will also introduce an optimised method for eliminating unnecessary edges and thus reduce the number of constraints in the inequality systems for higher constellations by taking the 32-APSK scheme as an illustrative case.

3.2.1 LLR calculation for 16-APSK constellation using the Voronoi decomposition

The 16-APSK constellation from DVB-S2 is composed of two concentric rings with 4 and 12 PSK uniformly spaced symbols in the inner radius R_1 and the outer radius R_2, respectively. The ratio between the outer radius and the inner radius is $\gamma = R_2/R_1$. For the next calculations, we will assume $\gamma = 3.15$. This ratio is compliant with the DVB-S2 standard for code rate $r = 2/3$. Figure 3.9 shows the result of the Voronoi decomposition and the suggested simplifications as a reference to the LLR calculation of bit b_3. Note that, given the constellation symmetry, we can estimate these LLRs by taking into account only the first quadrant of the complex plane, i.e., we can consider the absolute values of real and imaginary part of the received symbols with the purpose of reducing the number of sectors and the operations required to the classification. According to the formulations introduced in section 3.2, we obtained two Voronoi edges which split the 4 symbols that are located in the first quadrant. The blue edge – above which the seed is $\mathbf{s}_0$ and below which the seed is $\mathbf{s}_4$ – splits the symbols associated to $b_3 = 0$ whereas the red edge – above which the seed is $\mathbf{s}_8$ and below which the seed is $\mathbf{s}_{12}$ – splits the symbols associated to $b_3 = 1$.

As opposed to the empirical bounds we proposed in the *Ad Hoc* method, now we introduce a more formal criterion from which it is easier to achieve the sector boundaries for any constellation scheme. With this criterion we can be sure that our approximation has the same precision as the Max-Log approximation. As mentioned earlier, the splitting of bit b_3 must be firstly taken

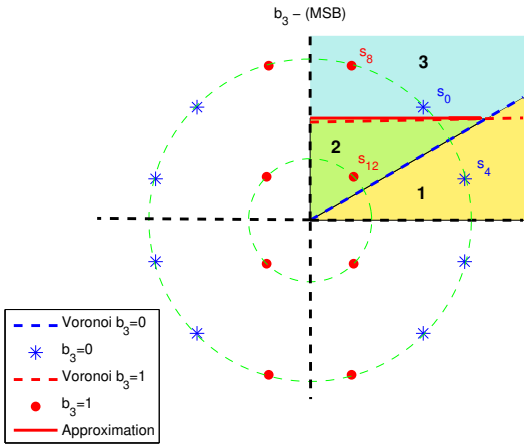

Figure 3.9: Decision areas for bit $\mathtt{b}_3$.

over the set of symbols for which this bit is 0 and then over the set of symbols for which $\mathtt{b}_3$ is 1. But fortunately, the bounds of the sectors associated to the bit $\mathtt{b}_3$ obtained through the original Voronoi decomposition criterion may be even more simplified as we include small adjustments to the Voronoi edges. These approximation can be included in order to make even easier both classification processes in which the received symbols must pass through. Nevertheless, as will be seen in Chapter 4, these introduced approximations come at the expense of a small performance degradation. In this example, one single edge had a small and arbitrated adjustment in its slope. Generally, these small adjustments can be wisely (but non-systematically) determined such that the symbols classifications become less complex.

Note that, in Figure 3.9, the sector 3 actually corresponds to the intersection between two Voronoi sectors. In this case, one of the intersected Voronoi sectors is associated to the seed $\mathbf{s}_0$ and the other one is associated to the seed $\mathbf{s}_8$, i.e., as a form of simplifying the stage of classification of $\mathbf{r}$, we may define new sectors which represent an intersection (or an approximated intersection) of both liner classifiers: the one for $\mathtt{b}_3 = 0$ and the other for $\mathtt{b}_3 = 1$. Equivalently, sector 2 is associated to the seeds $\mathbf{s}_0$ and $\mathbf{s}_{12}$. See that, for obtaining perfect intersections, the sector 1 should be, indeed, separated into two different sub-sectors. If we had aimed at defining the perfect intersection between the two remaining Voronoi sectors, we should have actually had one sub-sector associated to the intersection of the sectors with seeds $\mathbf{s}_4$ and $\mathbf{s}_{12}$ and another sub-sector associated to the intersection of the sectors with seeds $\mathbf{s}_4$ and $\mathbf{s}_8$. Instead of that, and as an additional form of approximation, we joined these sub-sectors into a single region. As a result, determining the pair of seeds associated to $\mathbf{r}$ is now reduced to the following simplified steps: (1) one single comparison between Q and a constant value which represents the limit of sectors 2 and 3 and; (2) one single comparison between the arc tangent of Q/I with a constant angle which represents the limit between sector 1 and sectors 2 and 3. Note that the edge simplification and the approximated merges between the original Voronoi sectors (in sector 3) will introduce some imprecision to the final LLR estimation. Due to this, we can not expect to obtain the same performance as the Max-Log anymore.

Let $\theta = \arctan\left(\frac{Q}{I}\right)$ be the angle between the imaginary and real part. Assuming the absolute values of I and Q, the LLRs for the bit b_3 can be approximated, in each sector, by the following equations:

Region 1: Bounds: $0 \leqslant \theta \leqslant \pi/6$

$$\begin{aligned} \mathrm{LLR}(b_3) &\approx -\frac{|\mathbf{r}-s_4|^2}{2\sigma^2} + \frac{|\mathbf{r}-s_{12}|^2}{2\sigma^2} \\ &= \frac{I\left[2R_2\cos(\pi/12) - 2R_1\cos(\pi/4)\right]}{2\sigma^2} + \\ &+ \frac{Q\left[2R_2\sin(\pi/12) - 2R_1\sin(\pi/4)\right]}{2\sigma^2} \\ &+ \frac{R_1^2 - R_2^2}{2\sigma^2} \end{aligned}$$

Region 2: Bounds: $\pi/6 < \theta \leqslant \pi/2, \quad Q \leqslant R_2\sin(\pi/6)$

$$\begin{aligned} \mathrm{LLR}(b_3) &\approx -\frac{|\mathbf{r}-s_0|^2}{2\sigma^2} + \frac{|\mathbf{r}-s_{12}|^2}{2\sigma^2} \\ &= \frac{I\left[2R_2\cos(\pi/4) - 2R_1\cos(\pi/4)\right]}{2\sigma^2} + \\ &+ \frac{Q\left[2R_2\sin(\pi/4) - 2R_1\sin(\pi/4)\right]}{2\sigma^2} \\ &+ \frac{R_1^2 - R_2^2}{2\sigma^2} \end{aligned}$$

Region 3: Bounds: $\pi/6 < \theta \leqslant \pi/2, \quad Q > R_2\sin(\pi/6)$

$$\begin{aligned} \mathrm{LLR}(b_3) &\approx -\frac{|\mathbf{r}-s_0|^2}{2\sigma^2} + \frac{|\mathbf{r}-s_8|^2}{2\sigma^2} \\ &= \frac{I\left[2R_2\cos(\pi/12) - 2R_2\cos(5\pi/12)\right]}{2\sigma^2} + \\ &+ \frac{Q\left[2R_2\sin(\pi/12) - 2R_2\sin(5\pi/12)\right]}{2\sigma^2} \end{aligned}$$

The general LLR expression for bits b_3 and b_2 from 16-APSK is given by

$$\mathrm{LLR}(b_j) \approx \frac{1}{\sigma^2}\left[I(I_{\mathcal{A}_{b_j=0}} - I_{\mathcal{A}_{b_j=1}}) + Q(Q_{\mathcal{A}_{b_j=0}} - Q_{\mathcal{A}_{b_j=1}})\right] + \frac{R^2_{\mathcal{A}_{b_j=1}} - R^2_{\mathcal{A}_{b_j=0}}}{2\sigma^2} \tag{3.21}$$

where $R^2_{\mathcal{A}_{b_j=0}} = I^2_{\mathcal{A}_{b_j=0}} + Q^2_{\mathcal{A}_{b_j=0}}$ and $R^2_{\mathcal{A}_{b_j=1}} = I^2_{\mathcal{A}_{b_j=1}} + Q^2_{\mathcal{A}_{b_j=1}}$ refer to the symbol energy (or the squared radius from the constellation symbol) for which the bit b_j is equal to 0 and the symbol energy for which the bit b_j is equal to 1, respectively. Note that in the context of hardware implementation, the constants $\mu_i = (I_{\mathcal{A}_{b_j=0}} - I_{\mathcal{A}_{b_j=1}})$, $\mu_q = (Q_{\mathcal{A}_{b_j=0}} - Q_{\mathcal{A}_{b_j=1}})$ and $c = (R^2_{\mathcal{A}_{b_j=1}} - R^2_{\mathcal{A}_{b_j=0}})/2$ can be calculated offline and later stored in memories.

Equivalently, the Voronoi decomposition for b_2 is taken over the set of symbols for which $b_2 = 0$ as well as over the set of symbols for which $b_2 = 1$. The sectors originally obtained

through this decomposition can also be wisely simplified by taking approximated Voronoi edges and by also merging the original Voronoi sectors into approximated sectors associated to two different seeds (one for $b_2 = 0$ and another for $b_2 = 1$), as explained before. This can be made for the same reasons and with the same purposes presented to b_3. Figure 3.10 illustrates the results of the original Voronoi decomposition as well as the included simplifications. From the absolute values of I and Q, the LLRs of bit b_2 can be estimated, in each sector, accordingly to the following equations:

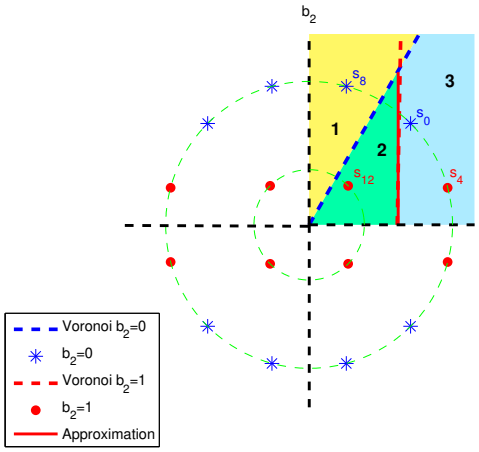

Figure 3.10: Decision areas for bit b_2.

Region 1: Bounds: $\pi/3 \leqslant \theta \leqslant \pi/2$

$$\begin{aligned} \mathrm{LLR}(b_2) &\approx -\frac{|\mathbf{r}-s_8|^2}{2\sigma^2} + \frac{|\mathbf{r}-s_{12}|^2}{2\sigma^2} \\ &= \frac{I\left[2R_2\cos(5\pi/12) - 2R_1\cos(\pi/4)\right]}{2\sigma^2} + \\ &+ \frac{Q\left[2R_2\sin(5\pi/12) - 2R_1\sin(\pi/4)\right]}{2\sigma^2} \\ &+ \frac{R_1^2 - R_2^2}{2\sigma^2} \end{aligned}$$

Region 2: Bounds: $0 \leqslant \theta < \pi/3, \quad I \leqslant R_2\cos(\pi/3)$

$$\begin{aligned} \mathrm{LLR}(b_2) &\approx -\frac{|\mathbf{r}-s_0|^2}{2\sigma^2} + \frac{|\mathbf{r}-s_{12}|^2}{2\sigma^2} \\ &= \frac{I\left[2R_2\cos(\pi/4) - 2R_1\cos(\pi/4)\right]}{2\sigma^2} + \\ &+ \frac{Q\left[2R_2\sin(\pi/4) - 2R_1\sin(\pi/4)\right]}{2\sigma^2} \\ &+ \frac{R_1^2 - R_2^2}{2\sigma^2} \end{aligned}$$

Region 3: Bounds: $0 \leqslant \theta < \pi/3, \quad I > R_2 \cos(\pi/3)$

$$\begin{aligned} LLR(b_2) &\approx -\frac{|\mathbf{r} - s_0|^2}{2\sigma^2} + \frac{|\mathbf{r} - s_4|^2}{2\sigma^2} \\ &= \frac{I\,[2R_2\cos(\pi/4) - 2R_2\cos(\pi/12)]}{2\sigma^2} + \\ &+ \frac{Q\,[2R_2\sin(\pi/4) - 2R_2\sin(\pi/12)]}{2\sigma^2} \end{aligned}$$

Table 3.2 presents the obtained values of μ_i, μ_q and c for bits b_3 and b_2, as we configure $\gamma = 3.15$.

Table 3.2: Constants for LLR estimation under 16-APSK constellation.

Bit	μ_i	μ_q	c	Bound
b_3	0.8421	0.0390	−0.5800	$\theta \leqslant \pi/6$
	0.5482	0.5482	−0.5800	$\theta > \pi/6,\ Q \leqslant R_2 \sin(\pi/6)$
	0.5092	−0.2940	0	$\theta > \pi/6,\ Q > R_2 \sin(\pi/6)$
b_2	0.0390	0.8421	−0.5800	$\theta \geqslant \pi/3$
	0.5482	0.5482	−0.5800	$\theta < \pi/3,\ I \leqslant R_2 \cos(\pi/3)$
	−0.2940	−0.5092	0	$\theta < \pi/3,\ I > R_2 \cos(\pi/6)$

The LLRs for the two least significant bits from 16-APSK can be coincidentally obtained in a similar way to the *Ad Hoc* approximation that was applied over the bits b_1 and b_0 from the 8-PSK constellation. Note that the approximated LLR expressions assume the closest complementary symbols $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$ as those that are symmetric with respect to the vertical or the horizontal axis of the complex plane. For the bit b_1, given the symmetry of these symbols with respect to the vertical axis of the plane, we can assume that $|I_{\mathcal{A}_{b_1=0}}| = |I_{\mathcal{A}_{b_1=1}}|$. In other words, we are considering that the symbols $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and the closest symbol $\mathbf{s}_{\mathcal{A}_{b_j=1}}$ are always in opposite sides of the vertical axis, even knowing that this association is not true for every received symbol. From this assumption, we obtain the following expressions

$$LLR(b_1) \approx -\frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{b_1=0}}|^2}{2\sigma^2} + \frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{b_1=1}}|^2}{2\sigma^2} \tag{3.22}$$

$$= \frac{2II_{\mathcal{A}_{b_1=1}}}{\sigma^2} \tag{3.23}$$

$$= \frac{2II_{\mathcal{A}_{b_0=1}}}{\sigma^2} \tag{3.24}$$

Note that this extrapolation is a perfect sense approximation for some particular cases but not for all. For instance, the symbols which reside in the sector associated to the the symbol $\mathbf{s}_8$ will be always correctly associated by the second classifier system to the complementary symbol $\mathbf{s}_{10}$ with respect to the bit b_1. The converse is true and therefore, the symbols which

reside in the sector associated to the symbol $\mathbf{s}_{10}$ will also be always correctly associated to the complementary symbol $\mathbf{s}_8$ of this bit. In fact, given that we transmit one of these symbols, the closest complementary constellation symbol is, indeed, in the opposite side of the real axis (with the same modulo of the real component). The same situation occurs for the sectors associated to the symbols $\mathbf{s}_{14}$ and $\mathbf{s}_{12}$, $\mathbf{s}_{15}$ and $\mathbf{s}_{13}$, $\mathbf{s}_{11}$ and $\mathbf{s}_9$. On the other hand, note that this assumption is not valid for the sectors associated to the symbols $\mathbf{s}_4$ and $\mathbf{s}_6$, for instance. In this case, the closest complementary symbol with respect to the bit $\mathfrak{b}_1$ might be not exactly symmetric with respect to the vertical axis, i.e., $|\mathrm{I}_{\mathcal{A}_{\mathfrak{b}_1=0}}| \neq |\mathrm{I}_{\mathcal{A}_{\mathfrak{b}_1=1}}|$. As we can see in the Figure 3.11, the symbols which reside in the sector associated to the symbols $\mathbf{s}_4$ should be correctly associated to the closest complementary symbol $\mathbf{s}_{14}$ instead of $\mathbf{s}_6$. But in order to avoid complex steps for handling this sort of impairment, we choose to simply extrapolate our first assumption for the entire constellation, assuming that these pairs of symbols will also be always associated to the symbols in opposite side of the horizontal axis (even not being the closest complementary symbol). Therefore, the LLR estimation would be given by Equation (3.24), without need of any further classification steps or additional constellation sectors. We remark that the elimination of the constant values which multiply the real part from the received symbol do not have huge impact over the final performance of this soft-decoder. Hence, we can take following approximation

$$\mathrm{LLR}(\mathfrak{b}_1) \approx \frac{\mathrm{I}}{\sigma^2} \tag{3.25}$$

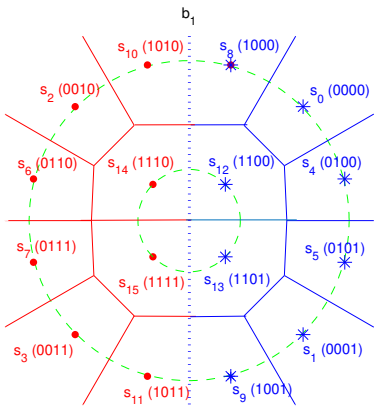

Figure 3.11: Reference of decomposition for the bit $\mathfrak{b}_1$ in 16-APSK constellation.

The same can be asserted in regard to the bit $\mathfrak{b}_0$ (LSB). But in this case, the symbols are symmetric with respect to the horizontal axis, as depicted in Figure 3.12. Hence, the approximated LLR is given by

$$\begin{aligned} \mathrm{LLR}(b_0) &\approx -\frac{|\mathbf{r}-\mathbf{s}_{\mathcal{A}_{b_0=0}}|^2}{2\sigma^2} + \frac{|\mathbf{r}-\mathbf{s}_{\mathcal{A}_{b_0=1}}|^2}{2\sigma^2} \\ &= \frac{2QQ_{\mathcal{A}_{b_0=0}}}{\sigma^2} \\ &= \frac{2QQ_{\mathcal{A}_{b_0=1}}}{\sigma^2} \\ &\approx \frac{Q}{\sigma^2} \end{aligned}$$

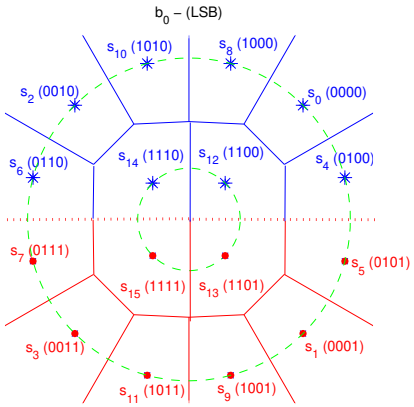

Figure 3.12: Reference of decomposition for the bit b_0 in 16-APSK constellation.

The results from this example will be further discussed in Chapter 4. In next subsection we will present an enhanced method for simplifying the original Voronoi sectors without relying on ad hoc merges employed in this example or on the non-systematic approximations and the intuitive edge adjustments. We aim to show a new formulation for optimising the approximations made over the Voronoi edges in the sense of minimising the performance effects caused by the simplifications.

3.2.2 LLR calculation for 32-APSK constellation using the Voronoi decomposition

In this subsection we boost the decomposition uniquely based on the Voronoi criterion by adding two supplementary steps of optimisation which aim to reduce the complexity of both linear classifier systems. For a clearer understanding of the proposed optimisations, we will focus on a single illustrative constellation 32-APSK. Note that in principle, the Voronoi decomposition and the additional optimisations could be applied to any APSK or PSK constellation. The first decomposition, described in section 3.2 as much as the second steps, which include the offline optimisations, are both shown to be also very adaptable to other schemes.

The 32-APSK constellation is composed of three concentric rings with 4, 12 and 16 PSK uniformly spaced symbols in the inner ring (radius R_1), in the intermediate ring (radius R_2)

and in the outer ring (radius R_3). Each one of these parameters are defined by DVB-S2 standard in terms of two ratios, $\gamma_1 = R_2/R_1$ and $\gamma_2 = R_3/R_1$. These ratios can vary according to the code rate. In our practical example, we are going to fix $r = 3/4$ such that $\gamma_1 = 2.85$ and $\gamma_2 = 5.27$. Note that the example treated in this subsection is actually developed for a modified version of the original constellation defined in DVB-S2. We keep the same bit mapping from this standard applying a counterclockwise rotation of $\phi = \pi/16$ to the symbols placed in outer ring. This rotation is proposed in order to obtain symmetries with respect to the coordinate axis. These symmetries are very important for achieving a significant reduction in the number of sectors to be considered along both stages of classification. In this case, not making this change would negatively affect the number of quadrants from the complex plane to be considered in the Voronoi decomposition. Instead of using one or two quadrants, we would have to create sectors for the entire constellation, i.e., in the four quadrants. Moreover, the second part document that includes the new DVB-S2 extensions, the DVB-S2X [44], already includes this important change as the default configuration, that is fortunately in accordance to what we propose in this example.

In section 3.2.1, we adopted approximated edges for 16-APSK scheme in order to simplify the optimal bounds obtained via the standard Voronoi criterion. As will be shown in the chapter of results from 16-APSK, this strategy does not cause much negative impact in terms of performance at the LDPC decoder output. On the other hand, it brought a significant reduction in terms of the amount of operations even not being either an optimal criterion of simplification or generalisable. In this section we introduce an analytical strategy for obtaining suitable and well-defined Voronoi a posteriori simplifications based on an energy function J. This function will be defined as a reference parameter and a measure of quality which will guide us to the new optimal simplified sectors. Note that this criterion of simplification will be employed after the standard Voronoi decomposition whose results are shown in the Figure 3.8.

The steps of optimisation are built from the specific geometrical features that we obtain previously, as a result of the Voronoi decomposition, and the handling of the energy function J, that includes the sum of bivariate normal probability density functions. The means of each bivariate function is placed at the exact coordinates of the constellation symbols of each sector. The basic goal of each optimisation step is to collapse a given selected edge, from the original decomposition, into a single vertex placed somewhere along the original edge. The optimum vertex is the one for which we obtain the maximum value of J. The search of this vertex will also iteratively modify the slopes of the edges previously connected to the collapsed edge. For each evaluated vertex v_i, these slopes will be adjusted such that the connected edges can terminate at the exact position of v_i, in which the eliminated edge is being collapsed.

Note that each edge from a Voronoi sector represents an inequality to be verified for the classification of each received symbol. The removal of an edge leads to one less inequality to that particular classifier system and consequently, further complexity reduction. The fact that the vertex search be limited to the points within the edges that are being collapsed turns out

to be reasonable since our optimisations are being included over the results of the Voronoi decomposition, which is already optimal according to our first (main) splitting criterion. At this point, it becomes acceptable to state that an optimal vertex obtained from the second criterion (the optimisation steps) must also be valid within the set of values fixed by the first optimal criterion (the Voronoi decomposition).

The energy function J is defined by the sum of bivariate normal probability density functions, whose means are given by the coordinates of each constellation symbol associated to a Voronoi sector. Note that for J, each bivariate PDF is unvalued in the outside of the sector for which it is associated. The function J is the sum of the surface integrals evaluated in each bivariate PDF, of each Voronoi sector with index $P \in \{1, \cdots N\}, N < M$, i.e.,

$$J = \sum_{P} \int\int_{A} f(x, \mathbf{s}_P, \psi) dS \tag{3.26}$$

where x and $\mathbf{s}_P$ are 1-by-2 vectors and ψ is a 2-by-2 symmetric positive definite covariance matrix. The vector x corresponds to the In-Phase and Quadrature coordinates from any point within the plane and $\mathbf{s}_P$ is the vector of coordinates of the constellation symbol (seed) from the sector P. The region A includes the set of points from the complex plane which belong to P. The diagonal elements of ψ contain the variances in each dimension of the plane (real and complex), whereas the off-diagonal elements contain the cross correlation between each dimension. Assuming the noise in each dimension is uncorrelated, the off-diagonal elements of ψ are null. Thus the function $f(x, \mathbf{s}_P, \psi)$ is given by

$$f(x, \mathbf{s}_P, \psi) = \frac{1}{\sqrt{|\psi|(2\pi)^2}} e^{-\frac{1}{2}(x-\mathbf{s}_P)^T \psi^{-1} (x-\mathbf{s}_P)} \tag{3.27}$$

and ψ is

$$\psi = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \tag{3.28}$$

The choice of the edges to be collapsed can be based on their relative length. In this case, we are assuming that the edges of shorter relative lengths can be eliminated at first since they will represent less impact in terms of the energy function, J. From now on we will take as example the edge collapsing for the set of sectors from the Voronoi decomposition for the bit b_2 of 32-APSK. First, we will focus on the set of sectors (and symbols) for which $b_2 = 0$, i.e, the Voronoi bounds defined from the constellation symbols ($\mathbf{s}_9$, $\mathbf{s}_{25}$, $\mathbf{s}_8$, $\mathbf{s}_{24}$, $\mathbf{s}_{16}$, $\mathbf{s}_{17}$, $\mathbf{s}_0$, $\mathbf{s}_{20}$). Figure 3.13 is a reference of indices for edges, vertices and sectors employed along the next optimisation steps. The first step of our optimisation aims to collapse the edge number 8 from the original Voronoi bounds to a given optimal vertex within this edge. In Figure 3.8, note the symmetry with respect to the vertical axis between the sectors associated to $b_2 = 0$ and those associated to $b_2 = 1$. See that b_1 also shares a similar geometrical configuration to the sectors of b_2. The sectors associated to $b_1 = 0$ and those associated to $b_1 = 1$ have symmetry with respect to the horizontal axis. These relations of symmetry can save us a lot of work since we can take advantage of the optimal vertices calculated for many distinct set of the Voronoi sectors at once.

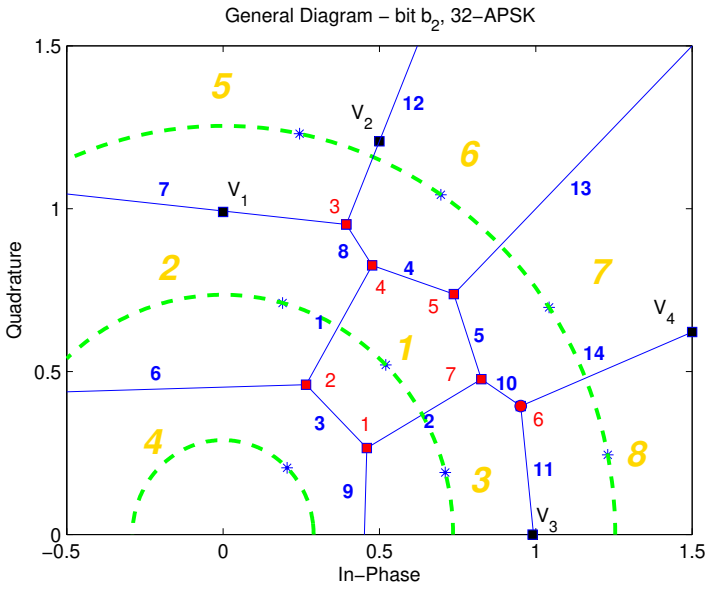

Figure 3.13: General diagram containing the edges (blue) and vertices(red) indices for the splitting of the symbols for which bit $b_2 = 0$ in the 32-APSK constellation.

The optimal vertex from the first step of the optimisation was chosen by comparing the values of J for the 90 samples (coordinates) of the edge number 8. The closer J_i from the i-th coordinate sample is from the maximum possible energy value, J_{MAX}, which is obtained by calculating the energy function over the original Voronoi sectors, the better and more opportune the vertex v_i under evaluation. In other words, we choose to eliminate the least significant portions of the Voronoi sectors by maximising the result of the sum of integrals of the normal probability density functions calculated over each surface, according to Equation (3.26). Figure 3.14 shows the energy function behaviour along the samples evaluated over this edge.

In the second step of the optimisation, we focus on the search of an optimal vertex along 90 samples of the edge number 10, the second shorter edge from the original Voronoi edges. The number of samples for the first and second steps of optimisation was arbitrarily set. The contour lines resulting from both steps can be seen in the Figure 3.15, where both edges are already collapsed into their respective optimal vertex.

The vertices V_1, V_2, V_3 and V_4 were kept fixed, as indicated in Figure 3.13. Fixing vertices is required whenever the edges connected to the collapsed edge are not connected to any other Voronoi edge beyond. In this case, these vertices were arbitrated in advance, as an auxiliary point around which we could adjust the the new slopes of the respective edges. The coordinates of these fixed vertices are presented in the Table 3.3.

The results from both optimisation steps are shown in Table 3.4, which includes the optimal vertices and the values of their respective energy J.

Table 3.3: Fixed vertices and their respective coordinates in the first and second steps of optimisation.

Edges	Coordinates	Fixed Vertex
1	[0.2413 0.4179]	2
4	[0.7255 0.7255]	5
7	[0.0000 0.9847]	V_1
12	[0.5000 1.2070]	V_2
2	[0.4179 0.2413]	1
5	[0.7255 0.7255]	5
11	[0.9847 0.0000]	V_3
14	[1.5000 0.6213]	V_4

Table 3.4: Results obtained after collapsing both edges.

Optimisation Step	J (Energy Function)	Optimal Vertices	Edge
–	5.9081	–	–
1	5.8915	[0.4282 0.8847]	8
2	5.8733	[0.8960 0.4198]	10

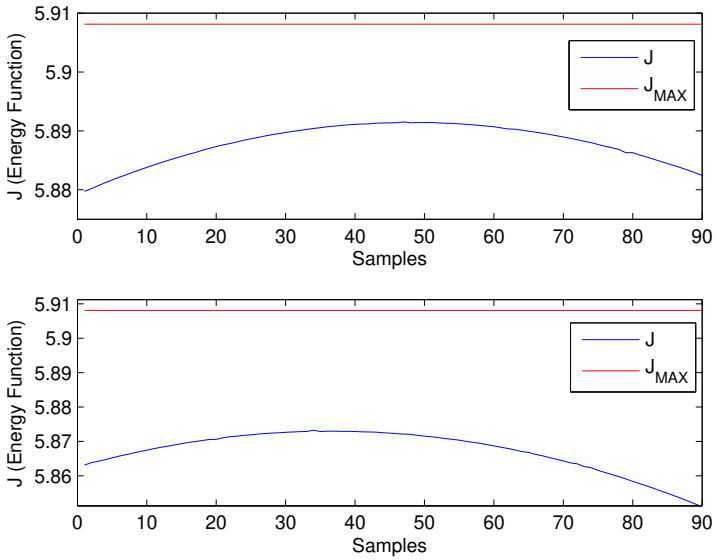

Figure 3.14: From the top to the bottom: Energy level along the 90 samples in the first edge collapsing; Energy level along the 90 samples in the second edge collapsing. Both graphs are limited a priori (red line) by the maximum energy level which is achieved through the calculation of J in the original Voronoi configuration.

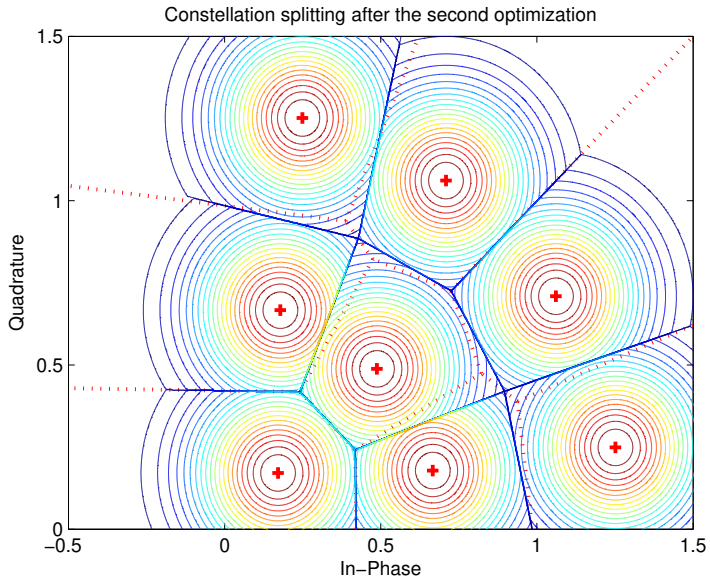

Figure 3.15: Contour Graph of the normal probability density functions for which $\mathtt{b}_2 = 0$ after collapsing the edges 8 and 10, in the second step of optimisation. The edges obtained from the Voronoi decomposition are represented in red and the new edges obtained after optimisation are represented in blue. Note that after collapsing edge number 8, the edges number 1, 4, 7, 12 were also modified. Similarly, after collapsing edge number 10, the edges number 2, 5, 11, 14 were modified.

From the point of view of the symbol's classification, introduced in section 3.2, the matrices of classification $\mathbf{A}_j$, $\mathbf{A}_k$ and the vectors $\mathbf{B}_j$, $\mathbf{B}_k$ from Equation (3.20) can be obtained from a final optimised matrix $\mathbf{A}$ which includes the adapted and remaining Voronoi edges. Matrix $\mathbf{A}$ has dimension L-by-2 and contains the factors that multiply the real part of the symbols in the first column and the imaginary part of these symbols in the second column. The ratio between the second and first column of this matrix represents the slope of each edge. For each sector P of the classifier system associated to $b_2 = 0$, the elements $A^P_{i,j}$ and B^P_i from the matrices $\mathbf{A}_j$, $\mathbf{A}_k$ and the vectors $\mathbf{B}_j$, $\mathbf{B}_k$ can be extracted from $\mathbf{A}$ by the following equation

$$A^P_{i,j} = \left(A_{i,j} \odot \begin{pmatrix} F(i,P) & F(i,P) \end{pmatrix}\right)^T \tag{3.29}$$

$$B^P_i = \left(B_i \odot F(i,P)\right)^T \tag{3.30}$$

where $\odot$ is the Hadamard (element-wise) matrix multiplication and $\mathbf{F}$ is the binary matrix whose columns are responsible for selecting the line coefficients (or the edges) from the matrix $\mathbf{A}$. Each column of the matrix $\mathbf{F}$ contains non-zero unitary elements at the row postions associated to the edges of the sector defined by this column. The final optimised matrices $\mathbf{A}$, $\mathbf{F}$ and the vector $\mathbf{B}$, from which we derive each matrix and vector of the classification systems, are found in Appendix C of this thesis.

In summary, for verifying if the received symbol $\mathbf{r}$ indeed resides in the sector P, from the linear inequality system associated to $b_2 = 0$ we need to ensure that

$$\mathbf{A}^P\mathbf{r} - \mathbf{B}^P \leqslant \mathbf{0} \tag{3.31}$$

If this inequality is satisfied, then the symbol $\mathbf{s}_P$ will be one of the two symbols included to this LLR approximation.

Due to the symmetries, the matrices of classification for the sectors from the classification system associated to $b_2 = 1$ can be obtained as we simply reverse the sign of the first column of $\mathbf{A}$, i.e., $A'_{i,1} = -A_{i,1}$. By doing this, Equation (3.29) becomes also valid for the sectors of $b_2 = 1$. Note that the matrices of classification for the sectors from the classifier system associated to $b_1 = 0$ are the same as from the system of $b_2 = 0$. Also, the sectors of the associated to $b_1 = 1$ can obtained by reversing the sign of the second column of $\mathbf{A}$, i.e., $A''_{i,2} = -A_{i,2}$.

The general LLR expression for b_2 and b_1 is given by Equation (3.21). The tables of constants μ_i, μ_q and c for b_2 and b_1 at the configuration $\gamma_1 = 2.85$ and $\gamma_2 = 5.27$ (code rate $r = 3/4$) were attached to Appendix D. The results from this applied example will be discussed in the next Chapter.

3.3 On the LLR Statistics

In this section we define a statistical model to the channel LLRs obtained from the soft-demappers based on the adapted Voronoi method proposed in this thesis. We derive the first

and second moments from a Gaussian Mixture Model that describes the statistical distributions that arises from this method.

First, consider a real valued BPSK constellation $\{-1,\ +1\}$, which maps a single bit $\mathtt{b}_0$ to $\{0,\ 1\}$, respectively. The LLR for this scheme of modulation is given in terms of the received real symbol $\mathbf{r} = \mathbf{s} + \mathbf{n}$ and the constellation symbols $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}} = +1$ and $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}} = -1$. Therefore, the LLR expression for this constellation is given by

$$\begin{aligned} \mathrm{LLR}(\mathtt{b}_0) &= -\frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}}|^2}{2\sigma^2} + \frac{|\mathbf{r} - \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}}|^2}{2\sigma^2} && (3.32)\\ &= -\frac{2\mathbf{r}}{\sigma^2} && (3.33) \end{aligned}$$

where σ^2 is the noise variance per dimension. We are assuming that $\mathbf{s}$ and $\mathbf{n}$ are independent random variables, $\mathbf{n} \sim \mathcal{N}(0, 2\sigma^2)$, $\mathsf{E}[.]$ is the expected value operator, $\mathsf{E}[\mathbf{s}] = 0$ and $\mathsf{E}[\mathbf{s}^2] = 1$.

We may approximate this LLR distribution with a Gaussian Mixture Model (GMM). For BPSK constellations, this can be made from a bimodal normal distribution. We are interested in calculating the expected values, i.e., the means of each mode from the mixture, conditional to the transmitted symbol. If the symbol $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}}$ is transmitted, we know that $\mathsf{E}\left[\mathbf{r} \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}}\right] = -1$. Thus, the expected value of the channel LLR distribution, given that $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}}$ was transmitted, can be derived from Equation (3.33) as

$$\begin{aligned} \mathsf{E}\left[\mathrm{LLR}(\mathtt{b}_0 = 0 \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}})\right] &= -\frac{2}{\sigma^2}\mathsf{E}\left[\mathbf{r} \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=0}}\right] && (3.34)\\ &= \frac{2}{\sigma^2} && (3.35) \end{aligned}$$

Similarly, if the symbol $\mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}}$ is transmitted, we know that $\mathsf{E}\left[\mathbf{r} \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}}\right] = +1$. Thus, the expected value of the LLR distribution is

$$\begin{aligned} \mathsf{E}\left[\mathrm{LLR}(\mathtt{b}_0 = 1 \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}})\right] &= -\frac{2}{\sigma^2}\mathsf{E}\left[\mathbf{r} \mid \mathbf{s}_{\mathcal{A}_{\mathtt{b}_0=1}}\right] && (3.36)\\ &= -\frac{2}{\sigma^2} && (3.37) \end{aligned}$$

The LLR variance is also derived from Equation (3.33) as follows

$$\begin{aligned} \mathrm{Var}\Big(\mathrm{LLR}\Big) &= \mathsf{E}\left[\left(-\frac{2\mathbf{r}}{\sigma^2}\right)^2\right] - \mathsf{E}^2\left[\left(-\frac{2\mathbf{r}}{\sigma^2}\right)\right] && (3.38)\\ &= \frac{4}{\sigma^4}\mathsf{E}\left[\left(\mathbf{s}^2 + 2\mathbf{s}\mathbf{n} + \mathbf{n}^2\right)^2\right] - \frac{4}{\sigma^4}\mathsf{E}^2\left[(\mathbf{s} + \mathbf{n})\right] && (3.39)\\ &= \frac{4}{\sigma^2} && (3.40) \end{aligned}$$

Equation (3.40) yields the following important relationship between the variance and the expected value of the output channel LLR:

$$\mathrm{Var}\Big(\mathrm{LLR}\Big) = 2\mathsf{E}\left[\mathrm{LLR}(\mathtt{b}_0) \mid \mathbf{s}\right] \qquad (3.41)$$

For M-ary PSK or APSK constellations, according to our proposed LLR simplification that is built from two constellation symbols, we can extend Equation (3.32) as an LLR approximation based on the complex symbols $\mathbf{s}_{\mathcal{A}_{b_j=0}}$ and $\mathbf{s}_{\mathcal{A}_{b_j=1}}$. Therefore, for a general constellation of order M, as derived in section 3.2, we obtain the following expression

$$\mathrm{LLR}(b_j) \approx \frac{1}{\sigma^2}\left(\mathrm{I}(\mathrm{I}_{\mathcal{A}_{b_j=0}} - \mathrm{I}_{\mathcal{A}_{b_j=1}}) + \mathrm{Q}(\mathrm{Q}_{\mathcal{A}_{b_j=0}} - \mathrm{Q}_{\mathcal{A}_{b_j=1}})\right) \tag{3.42}$$

The corresponding expected values from the LLR Gaussian Mixture Model we propose in this section can be calculated conditional to each symbol that is associated to a corresponding constellation sector. For a sector $P = i$, whose associated constellation symbol is $\mathbf{s}_{\mathcal{A}_{b_j=0}}$, we have

$$\mathrm{E}\left[\mathrm{LLR}(b_j \mid P = i)\right] = \frac{|\mathbf{s}_{\mathcal{A}_{b_j=0}} - \mathbf{s}_{\mathcal{A}_{b_j=1}}|}{2\sigma^2} \tag{3.43}$$

where $\mathbf{s}_{\mathcal{A}_{b_j=1}}$ is a symbol that resides in a neighbor sector $P = u$. This is also the closest symbol for which $b_j = 1$. Obviously, we can note that the expected LLR value for bit b_j conditional to the transmitted symbol from the sector $P = u$ has the same modulo than $\mathrm{E}[\mathrm{LLR}(b_j \mid P = i)]$ but different sign, given that the neighbor sector is, now, $P = i$, i.e.,

$$\mathrm{E}\left[\mathrm{LLR}(b_j \mid P = u)\right] = -\frac{|\mathbf{s}_{\mathcal{A}_{b_j=0}} - \mathbf{s}_{\mathcal{A}_{b_j=1}}|}{2\sigma^2} \tag{3.44}$$

Each Gaussian mean value from the Mixture can be calculated by simply analysing the available combinations among the proposed sectors. The LLR values that result from a received symbol which resides in one of the sectors whose associated seed is in the set of symbols for which $b_j = 0$ are always positive. Although, the LLR values from a received symbol that resides in one of the sectors whose associated seed is in the set of symbols for which $b_j = 1$ are always negative.

The LLR variance from each Gaussian function resulting from M-ary constellations is also given by twice its associated mean, as stated in Equation (3.41). Figure 3.16 shows the Probability Density Functions (*PDFs*) of the LLRs calculated from the adapted Voronoi and the Max-Log approximation for 16-APSK. The *PDF* that result from our approximation can be modelled through the statistics which are derived from the sector neighborhood.

The mean values of each Gaussian from the *PDF* of $\mathrm{LLR}(b_0)$ can be calculated as the imaginary part of the constellation symbols from the first quadrant, weighted by the inverse of σ^2. Similarly, the mean values of each Gaussian from the *PDF* of $\mathrm{LLR}(b_1)$ can be calculated as the real part of the same constellation symbols weighted by the inverse of σ^2. On the other hand, the mean values from the LLRs of the bits b_3 and b_2 can be calculated as described above. From the analysis of each possible sector combination and according to their corresponding seed, we obtain the following expected LLR values:

$$\mathrm{E}\Big[\mathrm{LLR}(\mathrm{b}_2)\Big] = \begin{pmatrix} \frac{|\mathbf{s}_0-\mathbf{s}_4|^2}{2\sigma^2} \\ -\frac{|\mathbf{s}_0-\mathbf{s}_4|^2}{2\sigma^2} \\ \frac{|\mathbf{s}_8-\mathbf{s}_{12}|^2}{2\sigma^2} \\ -\frac{|\mathbf{s}_0-\mathbf{s}_{12}|^2}{2\sigma^2} \end{pmatrix} \qquad \mathrm{E}\Big[\mathrm{LLR}(\mathrm{b}_3)\Big] = \begin{pmatrix} -\frac{|\mathbf{s}_0-\mathbf{s}_{12}|^2}{2\sigma^2} \\ -\frac{|\mathbf{s}_0-\mathbf{s}_8|^2}{2\sigma^2} \\ \frac{|\mathbf{s}_4-\mathbf{s}_{12}|^2}{2\sigma^2} \\ \frac{|\mathbf{s}_0-\mathbf{s}_8|^2}{2\sigma^2} \end{pmatrix}$$

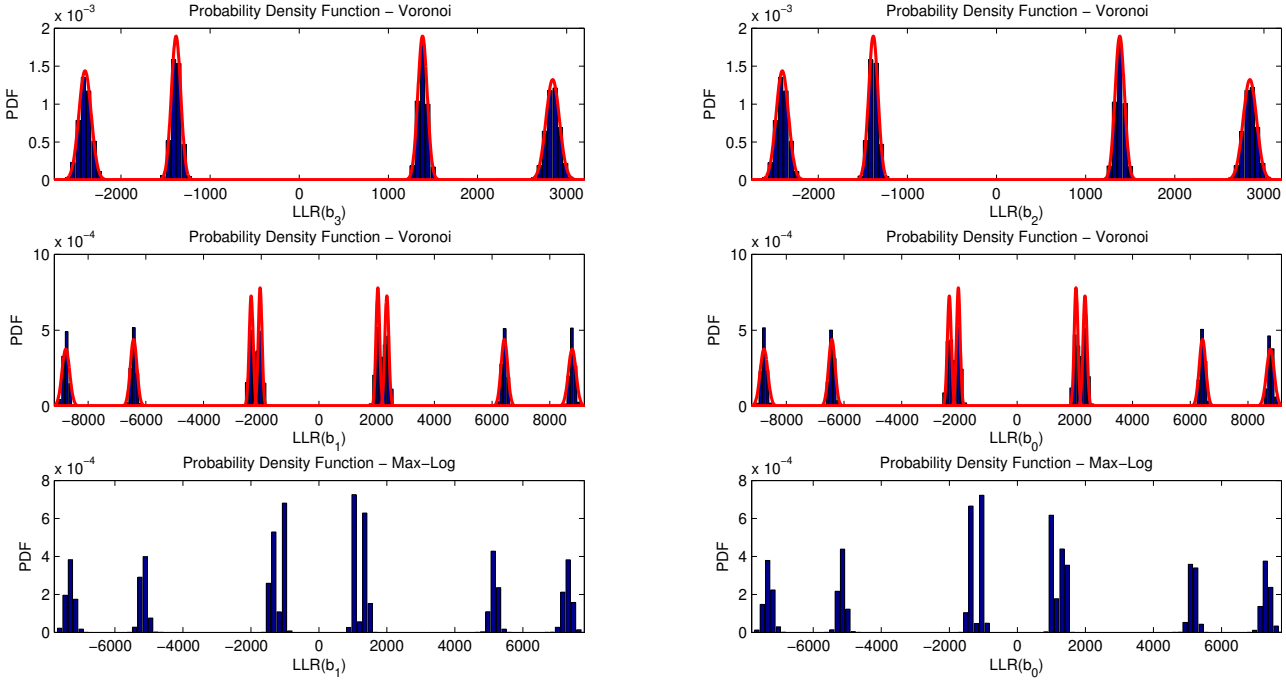

Figure 3.16: Probability Density Functions from the approximated LLRs based on the adapted Voronoi decomposition of 16-APSK constellation. The red curves represent the GMM from the modes of each Multimodal Gaussian. The blue bars represent the real obtained distribution of the LLR values. Note that the resulting *PDF* profiles from the Max-Log approximation and from the proposed method are very close to each other.

Chapter 4

Results

This work has the main objective of creating reliable and suitable soft-decoders with low complexity for LLR estimations under AWGN channel. In Chapter 3, we presented the key steps of two different solutions for simplified LLR estimation. We took the constellations from DVB-S2 standard as a practical application example. In this chapter we will focus on the simulations and results obtained from the implemented DVB-S2 model. These aspects will be presented and interpreted from the point of view of performance and complexity reduction. Therefore, the performance of both proposed simplifications as well as their complexity will be compared to the Max-Log approximation, the Shannon limit and the expected QEF condition. Results of the next Monte Carlo simulations were obtained for a fixed maximum number $I_{MAX} = 50$ of LDPC decoding iterations.

4.1 Results from the *Ad Hoc* approximation for 8-PSK and 16-APSK

The performance of the *Ad Hoc* soft-demapper was evaluated in two separate parts. First, we verified the performance from the point of view of the resulting hard-decisions, i.e., by taking into account only the sign of the LLRs. Second, we evaluated the LLRs in terms of their magnitude (reliability), i.e., the obtained LLRs were connected to the block of LDPC decoder. Figure 4.1 shows the bit error rate from the hard-decision only. As we can note in this figure, for 8-PSK the estimation of sign performed very close to the Max-Log. However, the sign estimation for the *Ad Hoc* method in 16-APSK constellation is slightly worse than the Max-Log.

The performance at the output of the LDPC decoder is shown in Figure 4.2. The offset term of the Offset Min-Sum [6] decoding algorithm is set to $\beta = 0.14$. Bit error rate was estimated for both constellations, in short FECFRAME configuration from DVB-S2 (16200 bits), code rate $r = 3/5$ for 8-PSK and $r = 2/3$ for 16-APSK.

These simulations confirm that the proposed LLR calculation has a low degradation of performance compared to the well-known Max-Log simplification. In addition, the method provides

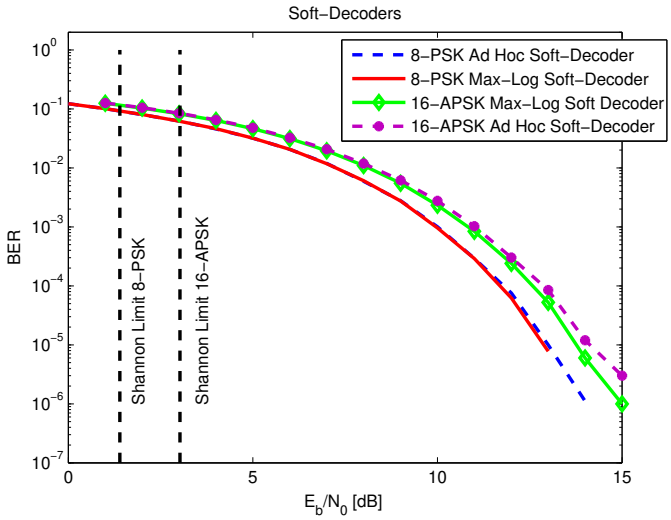

Figure 4.1: Bit error rate performance of the hard-decision.

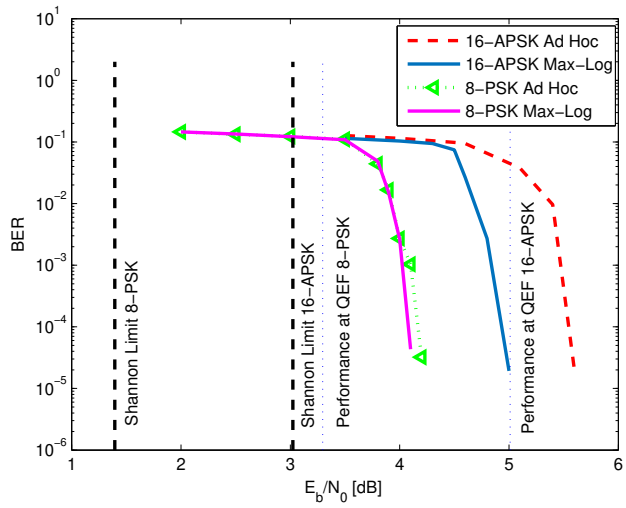

Figure 4.2: Bit-error rate at the output of LDPC decoder under 8-PSK and 16-APSK.

a significant reduction in hardware and software complexity since the LLRs are essentially obtained by the multiplication between the real and imaginary part of the received symbol by a set of constants that can be previously calculated and stored.

The computational complexity is shown in Tables 4.1 and 4.2. Note that, according to our criterions, the proposed soft-demapper has a lower complexity than the Max-Log since the number of operations of each type is much smaller.

The results reveal a good performance for applications under 8-PSK. On the other hand, a

Table 4.1: Computational complexity for computing the soft-information on the $\mathfrak{i}$-th bit in 8-PSK constellation.

	Proposed 8-PSK	**Max-Log 8-PSK**
Additions	$1 \longrightarrow M/2-3$	$25 \longrightarrow (3M+1)$
Multiplications	$2 \longrightarrow M/2-2$	$16 \longrightarrow (2M)$
Comparisons	$4 \longrightarrow (M/2)$	$6 \longrightarrow (M-2)$

Table 4.2: Computational complexity for computing the soft-information on the $\mathfrak{i}$-th bit in 16-APSK constellation.

	Proposed 16-APSK	**Max-Log 16-APSK**
Additions	$7 \longrightarrow M/2-1$	$49 \longrightarrow (3M+1)$
Multiplications	$16 \longrightarrow M$	$32 \longrightarrow (2M)$
Comparisons	$8 \longrightarrow (M/2)$	$14 \longrightarrow (M-2)$

degradation nearly 0.6dB with respect to the Max-Log approach is observed in *Ad Hoc* proposal under 16-APSK constellation. These analyses confirm that the proposed soft-demapper presents a satisfactory performance as well as low complexity.

The loss of performance in 16-APSK can be certainly justified by the adopted sector bounds due to these parameters do not include any optimised criterion. The *Ad Hoc* sectors include some points in the complex plane whose the corresponding LLRs should not be approximated by the constellation symbols associated to the sector we proposed given that these points are indeed, closer to other symbols from the constellation. Apparently, for APSK schemes, this wrong association proved to be a more critical aspect for the approximation.

The isolated results from the soft-demapper are much less affected than the results from LDPC decoder output. One possible explanation for this arises from the lack of scaling factors for adjusting the calculated LLRs levels, making them unsuitable or numerically unstable for the employed LDPC decoding algorithm. Figure 4.3 shows the scenarios where the lack of these scaling factors become unbearable. For received symbols whose real part is smaller than $\mathfrak{R}_0$, the proposed LLR values and Max-Log values are still quite close to each other. Above the limit $\mathfrak{R}_0$, the proposed approximation and Max-Log LLR values quickly move away from each other. Apart from the need of finding these factors, we conclude that the proposed splitting, even being reasonable and valid for the tested configurations, is not easily generalisable for higher order modulations since it is not tied to any formal and well-defined mathematical criterion. More details will be discussed in chapter 5.

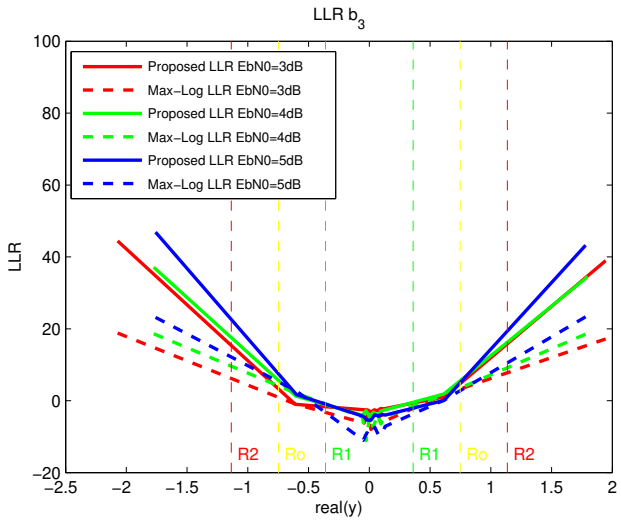

Figure 4.3: The LLR levels of one single bit calculated through Max-Log and *Ad Hoc* method under 16-APSK constellation.

4.2 Results from the approximation based on the adapted Voronoi criterion for 8-PSK and 16-APSK

In this section the performance evaluation of the soft-demapper based on adapted Voronoi method is built from a single analysis at the output of LDPC decoder. Simulations were configured to short FECFRAME transmission (16200 bits), code rate $\mathtt{r} = 3/5$ for 8-PSK and $\mathtt{r} = 2/3$ for 16-APSK. We performed the LDPC decoding using the Offset-Min-Sum [6] algorithm (offset term is kept on $\beta = 0.14$). The obtained results are shown in the Figure 4.4.

Note that, as well as in earlier simulations, the adopted code length is far from infinity. Due to the short code length, it is actually expected that the algorithm performs about 3dB away from Shannon limit. The normal FECFRAME configuration would outperform this result to less than 1dB away from the limit at cost of much higher computational cost and much more time of simulation.

The details about the Voronoi decomposition for the constellation 8-PSK were not described in any previous sections of this work given the simplicity of this application. Our implementation relies on the application of the original Voronoi criterion without any further simplification. Coincidentally, the final result of the constellation splitting, the calculated constants and the performance, were very close to what had been proposed and obtained in the *Ad Hoc* method. Furthermore, for 16-APSK scheme we can observe much less degradation of performance in Voronoi method than in *Ad Hoc* method. Remind that we obtained a degradation nearly 0.6dB at the output of LDPC decoder based on the *Ad Hoc*. On the other hand, the method based on the adapted Voronoi decomposition presents a degradation close to 0.1dB.

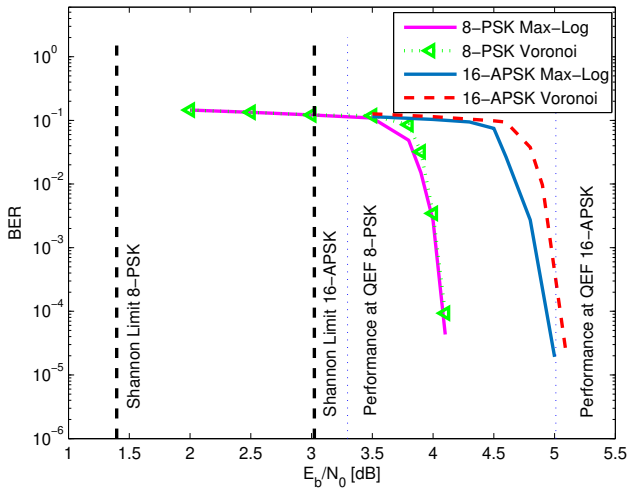

Figure 4.4: BER Performance at the output of LDPC decoder under 8-PSK and 16-APSK.

The results from Tables 4.3 and 4.4 confirm that the soft-demapper based on the adapted Voronoi criterion has a lower complexity than the Max-Log approximation since the number of operations is significantly reduced.

Table 4.3: Computational complexity for computing the soft-information on the $\mathfrak{i}$-th bit in 8-PSK constellation.

	Proposed 8-PSK	**Max-Log 8-PSK**
Additions	1	$25 \longrightarrow (3\mathrm{M}+1)$
Multiplications	2	$16 \longrightarrow (2\mathrm{M})$
Comparisons	4	$6 \longrightarrow (\mathrm{M}-2)$

Table 4.4: Computational complexity for computing the soft-information on the $\mathfrak{i}$-th bit in 16-APSK constellation.

	Proposed 16-APSK	**Max-Log 16-APSK**
Additions	6	$49 \longrightarrow (3\mathrm{M}+1)$
Multiplications	6	$32 \longrightarrow (2\mathrm{M})$
Comparisons	5	$14 \longrightarrow (\mathrm{M}-2)$

4.3 Results from the approximation based on the adapted Voronoi criterion for 32-APSK

This section aimed at analysing the bit error rate performance at the output of LDPC decoder by comparing two different approaches of LLR approximation. The simulations were configured to use Offset-Min-Sum decoding algorithm (offset term is kept on $\beta = 0.14$ under short FECFRAME transmission, 32-APSK constellation and code rate $\mathtt{r} = 3/4$. The bit error rate on AWGN channel is shown in Figure 4.5. Note that for 32-APSK constellation, the performance is almost the same as from the Max-Log approximation. Remind that for 16-APSK, we could observe a difference of about 0.1dB among Max-Log and adapted Voronoi bit error rate curve. This small variation of performance between 16-APSK and 32-APSK can be justified, since the adaptations suggested to the Voronoi edges of 32-APSK were built from an optimised criterion of adjustment, as opposed to what was made to the original Voronoi edges of 16-APSK constellation.

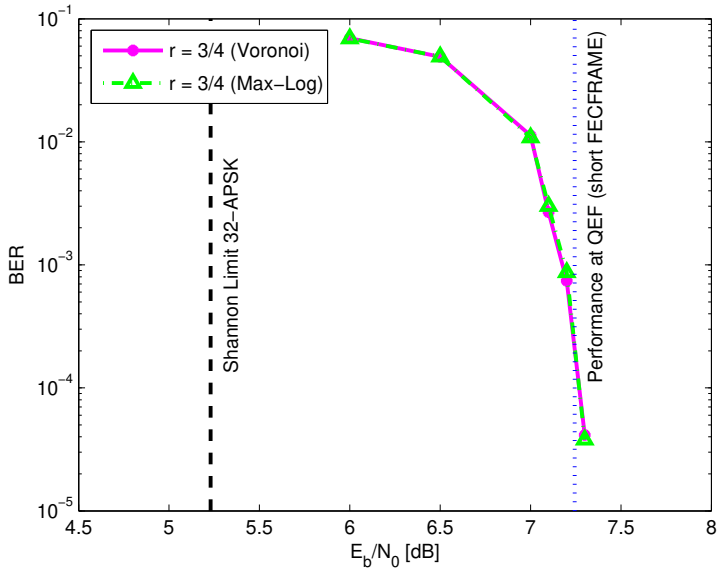

Figure 4.5: Bit error rate performance at the output of LDPC decoder for the LLR approximation based on adapted Voronoi decomposition and based on Max-Log approach for code rate $\mathtt{r} = 3/4$, 32-APSK constellation.

We note that both methods present a quite similar nature regarding their capacities of LLR approximation. We also found out no relevant difference among the statistical distributions built from each type of approximation. Instead of counting the number of each required operation, we chose to analyse the gains in terms of complexity reduction for both algorithms by comparing the elapsed time of simulation. This new metric is required due to the significant growth on the final number of operations to be computed in both methods. We emphasise that the estimations based on the counting of operations becomes increasingly difficult as the order of the modula-

tion increases. At this point we believe to be more convenient changing the paradigm of our complexity metric in order to simply avoid unnecessary efforts. Table 4.5 summarises the results from the both complexity evaluations obtained from Matlab. Specifically with this purpose, adapted Voronoi and Max-Log algorithms were implemented without including any compiled Matlab functions to the respective scripts. Particularly with this measure, we can expect to approximate the metric of hardware complexity by restricting the way the codes are built. As a result, we can obtain a very simple software metric that is also supposed to well represent an equivalent metric in terms of the final hardware design.

Table 4.5: Elapsed CPU time comparison for the LLR calculation involving one single bit in one single signal-to-noise ratio

Method	**Elapsed Time[s]**
Max-Log	6.0533
Voronoi + Optimisation	0.6263

Chapter 5

Conclusion

The ultimate goal of this thesis is to show and compare the performances of two proposed low-complexity methods of channel LLR approximation under AWGN. Our proposals are based on the splitting of the symbols, from either PSK or APSK constellations, into smaller sectors of decision. Due to this approach, the LLRs can always be approximated by using only two constellation symbols. Both methods are mainly aimed at simplifying hardware implementations of DVB-S2 receivers with low performance degradation and low dependence on the constellation order.

In Chapter 1, we discussed the relationships and reinterpretations on selected topics of the Digital Communication Theory, the Galois Theory and the Linear Block Codes properties. First we discussed the AWGN discrete-time model and presented some particular derivations which lead to the matched filter structure. Both subjects were very important for the implementation of the DVB-S2 model, employed as a test platform for the proposed simplification methods. We also discussed the Galois theory focusing on the relevant aspects to the implementation of the BCH and the LDPC encoder and decoder. Finally, we presented the DVB-S2 standard main features and the block structure adopted to the simulator.

In Chapter 2, we studied the LDPC codes focusing on the most important decoding algorithms. First we presented the BP sum-product algorithm and the min-sum. Then we introduced the min-sum variant algorithms which improve the trade-off between performance and decoder complexity. At the end of this chapter, we fix the offset min-sum as a very suitable algorithm for DVB-S2 in terms of performance and complexity.

In Chapter 3, we addressed the simplified soft-demappers tailor-made for DVB-S2 constellations. We described the *Ad-Hoc* solution for the constellation splitting and applied the method under 8-PSK and 16-APSK constellations. We also showed that the main issues involving this proposal resides on the infeasibility for creating generalisations at higher-order modulation as well as in the loss of performance due to the non-optimised sector boundaries. By taking these conclusions into account, we proposed an optimised splitting method based on the adaptation of the Voronoi decomposition. In addition, we showed the derivation of the simplified linear classifi-

cation systems, the general expressions and the obtained constants for the LLR approximation. We applied this approach for 8-PSK, 16-APSK and a modified version of DVB-S2 32-APSK constellations.

Finally, in Chapter 4, we presented the results from both approaches and remarked the benefits in terms of reduction on the number of operations and the good performance of bit error rate when the simplified LLRs were embedded on the LDPC decoder.

The Hardware Description Language (HDL) code for the adapted Voronoi solution as well as the *Ad-Hoc* solution is yet to be implemented. Our initial results obtained via software implementation has indicated a significant reduction in the number of operations and revealed good perspectives regarding to the final decoder performance.

It would be interesting to investigate the results of the proposed algorithms from a real hardware implementation. The analyses made until here are still incomplete to definitely establish these approaches as a possible best choice of simplification in terms of the hardware design. For a better consolidation of these simplifications, it would be important to start working with the hardware implementation such that the final logic area of these soft-demappers could be compared.

References

[1] R. G. Gallager, "Low-Density Parity Check Codes," Ph.D. dissertation, MIT, 1963.

[2] D. MacKay and R. Neal, "Good error-correcting codes on very sparse matrices," *Proc. IMA Conf. Cryptography*, vol. 1025, pp. 100–111, 1995.

[3] E. ETSI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News gathering and other broadband satellite applications (DVB-S2)," Tech. rep., ETSI, Tech. Rep., 2013.

[4] T. J. Richardson and R. L. Urbanke, "The capacity of Low-Density Parity-Check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.

[5] M. P. Fossorier, M. Mihaljević, and H. Imai, "Reduced complexity iterative decoding of Low-Density Parity Check codes based on belief propagation," *Communications, IEEE Transactions on*, vol. 47, no. 5, pp. 673–680, 1999.

[6] J. Chen and M. P. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *Communications Letters, IEEE*, vol. 6, no. 5, pp. 208–210, 2002.

[7] J. G. Proakis, M. Salehi, N. Zhou, and X. Li, *Communication systems engineering*. Prentice-hall Englewood Cliffs, 1994, vol. 1.

[8] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital communication*. Springer Science & Business Media, 2003.

[9] R. A. Carrasco and M. Johnston, *Non-binary error control coding for wireless communication and data storage*. The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: John Wiley & Sons, 2008.

[10] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[11] S. J. Johnson, "Reported thresholds and BER performance for LDPC and LDPC-Like codes," Department of Electrical and Computer Engineering, University of Newcastle, Australia, Tech. Rep., December 2012.

[12] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[13] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[14] G. D. Forney Jr and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *Information Theory, IEEE Transactions on*, vol. 44, no. 6, pp. 2384–2415, 1998.

[15] H. Malepati, *Digital media processing: DSP algorithms using C*. Newnes, 2010.

[16] S. Ling and C. Xing, *Coding theory: a first course*. Cambridge University Press, 2004.

[17] G. Birkhoff and S. Mac Lane, *A survey of modern algebra*. Universities Press, 1965.

[18] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

[19] J. B. Anderson, *Digital transmission engineering*. John Wiley & Sons, 2006, vol. 12.

[20] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (paris)*, vol. 2, no. 147-156, p. 116, 1959.

[21] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[22] W. W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri codes," *Information Theory, IRE Transactions on*, vol. 6, no. 4, pp. 459–470, 1960.

[23] R. T. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes." *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, 1964.

[24] G. Forney, "On decoding BCH codes," *IEEE Transactions on Information Theory*, vol. 11, no. 4, pp. 549–557, 1965.

[25] E. R. Berlekamp, "On decoding binary Bose-Chadhuri-Hocquenghem codes," *Information Theory, IEEE Transactions on*, vol. 11, no. 4, pp. 577–579, 1965.

[26] J. L. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," *Information Theory, IEEE Transactions on*, vol. 11, no. 4, pp. 580–585, 1965.

[27] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broadband services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210–227, 2006.

[28] P. S. M. P. G. F. Martina M., Masera G., "On Practical Implementation and Generalization of max* Operator for Turbo and LDPC Decoders." *IEEE Trans. on Instrumentation and Measurement*, vol. 61, no. 4, pp. 888–895, 2012.

[29] S. Ryoo, S. Kim, and S. P. Lee, "Efficient soft demapping method for high order modulation schemes," *CIC 2003*, 2003.

[30] J. W. Park, C. D. Ryu, M. H. Sunwoo, P. S. Kim, and D.-I. Chang, "Simplified soft-decision demapping algorithm for DVB-S2," in *SoC Design Conference (ISOCC), 2009 International*. IEEE, 2009, pp. 444–447.

[31] R. G. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[32] M. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *Communications IEEE Lett.*, vol. 2, no. 6, pp. 4165–167, 1998.

[33] S. Sankaranarayanan, S. K. Chilappagari, R. Radhakrishnan, and B. Vasic, "Failures of the Gallager B decoder: Analysis and applications," in *Proc. Information Theory and Applications Workshop UCSD*, vol. 17, 2006.

[34] S. J. Johnson, "Introducing low-density parity-check codes," *University of Newcastle, Australia*, 2006.

[35] M. J. Wainwright, T. Jaakkola, and A. S. Willsky, "Tree-based reparameterization for approximate inference on loopy graphs," in *Advances in neural information processing systems*, 2001, pp. 1001–1008.

[36] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, 2001.

[37] "Early-decision decoding of LDPC] codes, author=Blad, Anton, year=2009, publisher=Linköping University Electrnoic Press."

[38] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low density parity check nodes," *Communications, IEEE Transactions on*, vol. 50, pp. 406–414, 2002.

[39] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes." *Communications, IEEE Transactions on*, vol. 44, no. 10, pp. 1261–1271, 1996.

[40] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[41] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell System Technical Journal*, vol. 38, no. 3, pp. 611–656, 1959.

[42] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[43] M. G. M. Rodriguez and V. V. de Serio, "Voronoi cells via linear inequality systems," *Linear Algebra and its Application*, vol. 436, no. 7, pp. 2169–2186, 2012.

[44] E. ETSI, "ETSI EN 302307-2, Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part II: S2-Extensions (S2-X)," Tech. rep., ETSI, Tech. Rep., 2014.

Appendix A

The messages exchanged between check and bit nodes in the course of the decoding algorithm is defined as a conditional probability that the received bit is a 1 or a 0, given the received entry, $P_j^{int} = p(c_j = 1 \mid y)$, that is conditional probability of c_j is 1, given the received symbol y. Obviously, $p(c_j = 0 \mid y) = 1 - P_j^{int}$.

Consider the message $q_{i,j}^L$ as the probability sent by the bit node c_j to the check node f_i at iteration L. In particular, $q_{i,j}^0(1) = P_j^{int}$ and $q_{i,j}^0(0) = 1 - P_j^{int}$. Similarly, define $r_{i,j}^L = P_{i,j}^{ext}$ as the message sent by check node f_i to bit node c_j at iteration L. The message $r_{i,j}^L(0)$ is also associated to the probability that there is an even number of 1s in all the bit nodes but c_j connected to the check node f_i.

First of all, lets consider the probability that there is an even number of 1s in two bit nodes connected to a parity check node. Let q_1 being the probability that the bit node c_1 is 1 and q_2 being the probability that the bit node c_2 is 1 too. It follows that

$$\begin{aligned} p(c_1 \oplus c_2 = 0) &= q_1 q_2 + (1 - q_1)(1 - q_2) \\ &= 1 - q_1 - q_2 + 2q_1 q_2 \\ &= \frac{1}{2}\left[1 + (1 - 2q_1)(1 - 2q_2)\right] \\ &\triangleq q \end{aligned}$$

Thus, $(1 - q)$ is the probability that there is an odd number of 1s in the bit nodes c_1 and c_2. Now, lets extend our assumption by considering that there is an even number of 1s in the set of 3 bit nodes connected to one single check node. In this case, we have

$$\begin{aligned} p(c_1 \oplus c_2 \oplus c_3 = 0) &= q(1 - q_3) + (1 - q)q_3 \\ &= q - qq_3 + q_3 - qq_3 = q + q_3 - 2qq_3 \\ &= \frac{1}{2}[2(q + q_3) - 4qq_3] = \frac{1}{2}\left[1 + (1 - 2(1 - q))(1 - 2q_3)\right] \\ &= \frac{1}{2}\left[1 + (1 - 2q_1)(1 - 2q_2)(1 - 2q_3)\right] \end{aligned}$$

By induction, the probability that the i-th check node is satisfied is given by

$$p(c_1 \oplus c_2 \oplus ... \oplus c_n = 0) = \frac{1}{2} + \frac{1}{2}\prod_{i=1}^{n}(1 - 2q_i)$$

In other words,

$$P_{i,j}^{ext} = \frac{1}{2} + \frac{1}{2} \prod_{j' \in \mathcal{N}(i) \neq j} (1 - 2P_{i,j'}^{itr})$$

Therefore, the message that i-th check node sends to the j-th bit node at iteration L is

$$E_{i,j} = \mathrm{LLR}\left(\frac{P_{i,j}^{ext}}{1 - P_{i,j}^{ext}}\right) = \ln\left(\frac{\frac{1}{2} + \frac{1}{2}\prod_{j' \in \mathcal{N}(i) \neq j} (1 - 2P_{i,j'}^{itr\ (L-1)})}{\frac{1}{2} - \frac{1}{2}\prod_{j' \in \mathcal{N}(i) \neq j} (1 - 2P_{i,j'}^{itr\ (L-1)})}\right)$$

Appendix B

The *Jacobian logarithm identity* is given by

$$\ln(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + \ln(1 + e^{-|x_1 - x_2|}) \tag{B.1}$$

From

$$\begin{aligned} \ln(e^{x_1} + e^{x_2}) &= \ln\left(\frac{e^{\max(x_1,x_2)}}{e^{\max(x_1,x_2)}}(e^{x_1} + e^{x_2})\right) \\ &= \ln\left(\frac{e^{\max(x_1,x_2)}}{e^{\max(x_1,x_2)}}e^{x_1} + \frac{e^{\max(x_1,x_2)}}{e^{\max(x_1,x_2)}}e^{x_2}\right) \\ &= \max(x_1, x_2) + \ln\left(e^{x_1 - \max(x_1,x_2)} + e^{x_2 - \max(x_1,x_2)}\right) \end{aligned}$$

- if $\max(x_1, x_2) = x_1$

$$\ln(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + \ln(1 + e^{x_2 - x_1})$$

- if $\max(x_1, x_2) = x_2$

$$\ln(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + \ln(1 + e^{x_1 - x_2})$$

that yields

$$\ln(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + \ln(1 + e^{|x_1 - x_2|}) \tag{B.2}$$

The so-called *Max-Log* approximation uses only the first term of the Equation (B.2).

Appendix C

The final matrices $\mathbf{A}$, $\mathbf{F}$ and the vector $\mathbf{B}$ obtained from the adaptation steps applied over the original Voronoi classifier are presented below. From these matrices we are able to derive each optimised matrix of classification, as described in the section 3.2.2.

$$\mathbf{A} = \begin{pmatrix} -2.4972 & 1.0000 \\ -0.3734 & 1.0000 \\ -0.3167 & -0.3167 \\ 0.5357 & 1.0000 \\ 1.7930 & 1.0000 \\ -0.0074 & -0.4953 \\ 0.2336 & 1.0000 \\ -0.4953 & -0.0074 \\ 4.7314 & 1.0000 \\ -4.4889 & 1.0000 \\ 0.3521 & -0.3521 \\ -0.3336 & 1.0000 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} -0.1846 \\ 0.0853 \\ -0.2088 \\ 1.1141 \\ 2.0263 \\ -0.2088 \\ 0.9847 \\ -0.2088 \\ 4.6592 \\ -1.0374 \\ 0.0000 \\ 0.1209 \end{pmatrix}$$

$$\mathbf{F} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Appendix D

The next tables organise the constants μ_i, μ_q and c from the bits b_2 and b_1 for the configuration $\gamma_1 = R_2/R_1 = 2.85$ and $\gamma_2 = R_3/R_1 = 5.27$, code rate $r = 3/4$ and modulation 32-APSK.

Table D.1: Constant values for μ_i used in the approximation of LLR(b_2). Each column is associated to a sector obtained from the splitting to $b_2 = 0$ and each row is associated to a sector obtained from the splitting to $b_2 = 1$.

$b_2 = 0$ / $b_2 = 1$	1	2	3	4	5	6	7	8
1	–	0.6665	–	0.6592	–	–	–	–
2	0.6665	0.3572	0.8451	0.3498	0.4275	0.8875	1.2396	1.4301
3	–	0.8451	–	0.8378	–	–	–	–
4	0.6592	0.3498	0.8378	0.3424	–	–	1.2322	1.4227
5	–	0.4275	–	–	0.4979	0.9579	1.3099	–
6	–	0.8875	–	–	0.9579	–	–	–
7	–	1.2396	–	1.2322	1.3099	–	–	–
8	–	1.4301	–	1.4227	–	–	–	–

Table D.2: Constant values for μ_q used in the approximation of LLR(b_2). Each column is associated to a sector obtained from the splitting to $b_2 = 0$ and each row is associated to a sector obtained from the splitting to $b_2 = 1$.

$b_2 = 0$ / $b_2 = 1$	1	2	3	4	5	6	7	8
1	–	0.1786	–	−0.3167	–	–	–	–
2	−0.1786	0	−0.4879	−0.4953	0.5849	0.3944	0.0424	−0.4176
3	–	0.4879	–	−0.0074	–	–	–	–
4	0.3167	0.4953	0.0074	0	–	–	0.5377	0.0777
5	–	−0.5849	–	–	0	−0.1905	−0.5426	–
6	–	−0.3944	–	–	0.1905	–	–	–
7	–	−0.0424	–	−0.5377	0.5426	–	–	–
8	–	0.4176	–	−0.0777	–	–	–	–

Table D.3: Constant values for c used in the approximation of LLR(b_2). Each column is associated to a sector obtained from the splitting to $b_2 = 0$ and each row is associated to a sector obtained from the splitting to $b_2 = 1$.

$b_2 = 0$ / $b_2 = 1$	1	2	3	4	5	6	7	8
1	–	0	–	0.2088	–	–	–	–
2	0	0	0	0.2088	−0.5760	−0.5760	−0.5760	−0.5760
3	–	0	–	0.2088	–	–	–	–
4	−0.2088	−0.2088	−0.2088	0	–	–	−0.7848	−0.7848
5	–	0.5760	–	–	0	0	0	–
6	–	0.5760	–	–	0	–	–	–
7	–	0.5760	–	0.7848	0	–	–	–
8	–	0.5760	–	0.7848	–	–	–	–

Table D.4: Constant values for μ_i used in the approximation of LLR(b_1). Each column is associated to a sector obtained from the splitting to $b_1 = 0$ and each row is associated to a sector obtained from the splitting to $b_1 = 1$.

$b_1 = 0$ / $b_1 = 1$	1	2	3	4	5	6	7	8
1	–	–	0.1786	−0.3167	–	–	–	–
2	–	–	–	−0.0074	–	–	–	–
3	−0.1786	−0.4880	0	−0.4953	−0.4176	0.0424	0.3944	0.5849
4	0.3167	0.0074	0.4953	0	0.0777	0.5377	–	–
5	–	–	0.4176	−0.0777	–	–	–	–
6	–	–	−0.0424	−0.5377	–	–	–	0.5426
7	–	–	−0.3944	–	–	–	–	0.1905
8	–	–	−0.5849	–	–	−0.5426	−0.1905	0

Table D.5: Constant values for μ_q used in the approximation of LLR(b_1). Each column is associated to a sector obtained from the splitting to $b_1 = 0$ and each row is associated to a sector obtained from the splitting to $b_1 = 1$.

$b_1 = 0$ / $b_1 = 1$	1	2	3	4	5	6	7	8
1	–	–	0.6665	0.6592	–	–	–	–
2	–	–	–	0.8378	–	–	–	–
3	0.6665	0.8452	0.3572	0.3498	1.4301	1.2396	0.8875	0.4275
4	0.6592	0.8378	0.3498	0.3424	1.4227	1.2322	–	–
5	–	–	1.4301	1.4227	–	–	–	–
6	–	–	1.2396	1.2322	–	–	–	1.3099
7	–	–	0.8875	–	–	–	–	0.9579
8	–	–	0.4275	–	–	1.3099	0.9579	0.4979

Table D.6: Constant values for c used in the approximation of $\mathrm{LLR}(\mathsf{b}_1)$. Each column is associated to a sector obtained from the splitting to $\mathsf{b}_1 = 0$ and each row is associated to a sector obtained from the splitting to $\mathsf{b}_1 = 1$.

$\mathsf{b}_1 = 1$ \ $\mathsf{b}_1 = 0$	1	2	3	4	5	6	7	8
1	–	–	0	0.2088	–	–	–	–
2	–	–	–	0.2088	–	–	–	–
3	0	0	0	0.2088	−0.5760	−0.5760	−0.5760	−0.5760
4	−0.2088	−0.2088	−0.2088	0	−0.7848	−0.7848	–	–
5	–	–	0.5760	0.7848	–	–	–	–
6	–	–	0.5760	0.7848	–	–	–	0
7	–	–	0.5760	–	–	–	–	0
8	–	–	0.5760	–	–	0	0	0