UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Channel Estimation and Coding for Improved Performance in Wireless Systems

Ph.D. THESIS

Author: Alessandra Mior
Coordinator: Prof. Matteo Bertocco
Supervisor: Prof. Silvano Pupolin

15 October 2008

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

SCUOLA DI DOTTORATO DI RICERCA
IN INGEGNERIA DELL'INFORMAZIONE –
XXI CICLO
Indirizzo: Ingegneria Elettronica e delle Telecomunicazioni

Abstract

Two of the most important issues in wireless systems are addressed: channel estimation and channel coding. The frequency domain channel estimation techniques are used to estimate the channel in a WiMAX system (standard IEEE 802.16e [1]-[2]). In the same system, Low-Density Parity-Check (LDPC) codes [3, 4] are proposed and perform quite well. In this thesis decoding techniques for these codes are studied: a modification to the Message Passing Algorithm (MPA) [5] is proposed in order to reduce the number of multiplications. Then two alternative decoding methods for LDPC codes are explored: the Priority First Search Algorithm (PFSA) [6], [7], [8] and the Genetic Algorithm (GA) [9], [10], which transform the decoding problem into a search problem.

FInally, an alternative coding scheme is proposed: concatenation of short LDPC and turbo codes as outer and inner codes, respectively. Aim of this concatenation is a coding system that performs very well for all ranges of SNR's and respects some delay constraints.

Sommario

In questa tesi sono stati considerati due dei piú importanti argomenti di ricerca per i sistemi wireless: la stima e la codifica di canale. Le tecniche per la stima di canale in frequenza sono state applicate per stimare il canale in un sistema WiMAX (Standard [1] and [2]). Nello standard che definisce questo sistema, vengono proposti, come sistema di codifica opzionale, i Low-Density Parity-Check (LDPC) codes che sono codici la cui matrice di paritá ha una bassa densitá di 1 ed hanno prestazioni molto buone.

In questa tesi alcune tecniche per la decodifica di questi codici sono state studiate: allo scopo di ridurne il numero di moltiplicazioni, é stata proposta una modifica al Message Passing Algorithm (MPA), l'algoritmo di decodifica che viene comunemente usato. Sono stati, poi, analizzati due algoritmi alternativi per la decodifica degli LDPC: il Priority First Search Algorithm (PFSA) ed il Genetic Algorithm (GA), che trasformano il problema della decodifica in un problema di ricerca.

Infine viene proposto uno schema di codifica alternativo, consistente nella concatenazione di LDPC e turbo codici con parole corte allo scopo di scoprire un sistema di codifica che abbia buone prestazioni per tutti gli SNR e rispetti alcune specifiche stringenti sui tempi di ritardo.

A Federico...

Contents

1 Introduction **1**

2 Channel estimation in OFDM systems **7**

2.1 Channel model 8

2.2 Linear equalization 9

2.2.1 Peak distortion criterion 10

2.2.2 Mean square error criterion 11

2.3 Channel estimation 11

2.3.1 Least-Square (LS) Estimator 12

2.3.2 Minimum Mean Square Error (MMSE) estimator 13

2.3.3 Maximum Likelihood (ML) estimator 14

2.4 Channel tracking 14

2.4.1 Time domain 14

2.4.2 Frequency domain 25

2.5 System description 32

2.5.1 Channel estimation 33

2.5.2 Use of the described algorithm considering only pilot subcarriers 38

2.5.3 Channel tracking 39

2.6 Simulation setup 41

2.6.1 Code 41

2.6.2 Frame construction 42

2.6.3 Channel 43

2.6.4 Channel estimation 43

2.6.5 Parameters 44

2.7 Results 45

3 Expectation maximization algorithm for joint channel estimation and data detection **51**

3.1 Introduction 51

3.2 The Expectation-Maximization (EM) algorithm 52

3.2.1 Introduction 52

3.2.2 General statement of the EM algorithm 52

3.2.3 Convergence of the EM algorithm 54

3.3 Joint channel estimation and data detection for OFDM systems 54
3.3.1 System model 54
3.3.2 Channel estimation 57
3.3.3 Joint channel estimation and data detection using EM algorithm 59
3.4 System description and simulation results 64
3.4.1 Results 64
3.5 Conclusions 65

4 Channel coding - Low-Density Parity-Check (LDPC) Codes 69
4.1 Low-Density Parity-Check (LDPC) codes 69
4.1.1 Code definition and specification: Tanner's method 70
4.2 Irregular LDPC codes 73
4.2.1 Channel 74
4.2.2 Distribution and their properties 74
4.2.3 Optimization 78
4.3 LDPC encoding 82
4.3.1 Parity-check matrix' construction 82
4.3.2 Encoding operation 83
4.4 The decoding problem 85
4.5 LDPC decoding's algorithms 87
4.5.1 The decoding algorithm by Gallager 87
4.5.2 The decoding algorithm by MacKay 92

5 Alternative algorithms for LDPC decoding 99
5.1 Message passing algorithm: complexity reduction 99
5.1.1 LDPC decoding techniques 99
5.1.2 Proposed algorithm 102
5.1.3 System description 103
5.1.4 Simulation results 104
5.2 Message passing algorithm: quantized version 106
5.2.1 Representation of real numbers in fixed-point data type 106
5.2.2 Implementation of a function with fixed-point data type 107
5.2.3 Implementation of the soft decoding algorithm using fixed-point data type 108
5.3 Other decoding algorithm for LDPC codes 109
5.3.1 Priority first search algorithm 109
5.3.2 The genetic algorithm (GA) 113

6 Concatenation of short LDPC and turbo codes for improved performance 117
6.1 Introduction 117
6.1.1 Concatenation: why and expected results 118
6.2 Error pattern at the output of the turbo code and LDPC construction . 119

6.2.1 Turbo code: '3G 'type . . . 119
6.2.2 LDPC code construction . . . 124
6.3 System description . . . 125
6.4 Results . . . 125
6.4.1 Overall rate 1/3 . . . 126
6.4.2 Overall rate 5/16 . . . 126
6.4.3 Overall rate 1/4 . . . 127
6.4.4 Conclusion . . . 128

7 WiMAX standard 131
7.1 Aim and frequency allocation . . . 131
7.2 OFDMA symbol description, symbol parameters and transmitted signal 132
7.2.1 Time domain description . . . 132
7.2.2 Frequency domain description . . . 133
7.2.3 Parameters . . . 133
7.3 Frame duration . . . 135
7.4 Downlink Subcarriers Allocation . . . 137
7.4.1 Preamble . . . 138
7.4.2 Partial Usage Of Subchannel (PUSC) . . . 139
7.5 Channel coding . . . 148
7.5.1 Randomization . . . 149
7.5.2 Encoding . . . 150
7.5.3 Interleaving . . . 156
7.5.4 Modulation . . . 156

8 Performance evaluation 159
8.1 WiMAX simulator . . . 159
8.1.1 Transmitter . . . 160
8.1.2 PUSC Permutation . . . 161
8.1.3 Frame assembler . . . 161
8.1.4 Channel . . . 161
8.1.5 Frame deassembler and Data/pilots separation . . . 161
8.1.6 Channel estimation . . . 161
8.1.7 Receiver . . . 162
8.2 System parameters . . . 162
8.3 Infinite computational resource . . . 162
8.4 Finite computational resource . . . 165

9 Conclusions 173

A Convergence theorem for EM algorithm 177

B Channel length estimation **179**
B.1 Introduction . . . 179
B.2 Algorithm for the noise variance and the CIR length estimation . . . 180

C Optional subchannel allocations in WiMAX systems **185**
C.1 PUSC variations . . . 185
C.1.1 Potentially optional allocations . . . 185
C.2 Other allocation methods . . . 186
C.2.1 Full Usage of Subchannel (FUSC) . . . 186
C.2.2 Optional allocations . . . 189

D Renumbering sequence for PUSC permutation **193**

E Optional codes in WiMAX standard **195**
E.1 Block Turbo Coding (BTC) . . . 195
E.2 Convolutional Turbo Code (CTC) . . . 196
E.2.1 CTC interleaver . . . 199
E.2.2 Determination of CTC circulation state . . . 200
E.2.3 Subpacket generation . . . 200

Bibliography **205**

Chapter 1

Introduction

Channel estimation and coding are two of the most important issues in the new generation wireless systems. This work is about wireless systems and some important standard have been presented, for example: WiMAX standard [1] - [2] TGnSync [11] or WWise [12] for IEEE 802.11n and 3GPP-LTE [13], but in this work our attention will be focused on the mobile part of the first one.

The IEEE 802.16 telecommunications standard [1] envisions broadband wireless access technology as a means of providing wireless "last mile" broadband access in a metropolitan area network (MAN). The performance and services should be comparable or better than traditional DSL, cable or T1/E1 leased line services. Especially in areas beyond the reach of DSL and cable, IEEE 802.16 could offer a cost-effective broadband access solution. The term WiMax (worldwide interoperability for microwave access) has become synonymous with IEEE 802.16, promoting and certifying compatibility and interoperability of broadband wireless products. In its original release 802.16 focused on line-of-sight (LOS) applications in the licensed 10 to 66 GHz frequency range based on single carrier (SC) transmission (WirelessMAN-SC). In a first amendment non-line-of-sight (NLOS) applications in licensed and unlicensed bands in the 2 to 11 GHz frequency range were covered (WirelessMAN-SCa). To meet the requirements of a low cost solution in a multipath environment, orthogonal frequency division multiplexing (OFDM) was chosen as physical layer transmission technique (WirelessMAN-OFDM). To deliver optimum broadband wireless access performance, the concept of scalable OFDMA (orthogonal frequency division multiple access) was adopted. The architecture is based on a scalable subchannel bandwidth using a variable sized FFT according to the channel bandwidth. In the standard IEEE 802.16e, [2], there is an ongoing evolution of IEEE 802.16 addressing mobile applications thus enabling broadband access directly to portable devices like smartphones, PDAs, notebooks and laptop computers.

The Orthogonal Frequency Division Multiplexing (OFDM) is a spectrally efficient form of frequency division multiplexing (FDM) and divides its allocated channel spectrum into several subchannels. It is used to overcome the inter-symbol interference (ISI), through the cyclic prefix (CP), that is longer than the order of the channel impulse response (CIR). OFDM is inherently robust against frequency selective fading,

since each subchannel occupies a relatively narrow-band, where the channel frequency characteristic is nearly flat. We note that OFDM can be seen as an effective way of increasing data rates and simplifying the equalization in wireless communications, because only a simple one-tap channel equalizer in the frequency domain is required. In the end, we recall also that OFDM is computational efficient, because the fast Fourier transform (FFT) can be used to implement the modulation and demodulation operations. However, OFDM has some disadvantages. One is the high peak-to-average power ratio, another one is not achieved over a multipath fading channel.

In this work, first of all we deal with the problem of the channel estimation in a wireless OFDM system and then with the channel coding for the same system. We also consider some decoding techniques for the Low-Density Parity-Check (LDPC) codes,a king of codes included in most of the latter wireless systems, due to their near-capacity error correction.

Contents of the thesis are divided in Chapters as described following:

- **Chapter 2** : the problem of the channel estimation is addressed. First of all we briefly describe the known channel estimation algorithms in time and frequency domain. Then considering we are interested in their performance in an OFDM system we describe the channel estimation in the frequency domain for an OFDM system. Algorithm such as Least Square (LS), Minimum Mean Square Error (MMSE), Linear Minimum Mean Square Error (LMMSE) [14] are described assuming we know all the transmitted symbols and only pilots. Also an adaptive algorithm [15] in the frequency domain is described; this assumes to have an initial channel estimation and performs the tracking of the channel which is assumed slowly time-variant.

  Aim of this study is the selection of a channel estimation algorithm which gives good performance in a WiMAX system. To perform this selection we implemented a quasi-compliant WiMAX standard simulator[1] and valuated the performance and the Mean Square Error (MSE) for the LS algorithm, two variation of the LMMSE algorithm, one dependent on the transmitted data and the other not, and the adaptive frequency algorithm. Result show that the LS and the LMMSE dependent on the transmitted data algorithms perform very well, whereas the adaptive one performs very well for speeds below 50 km/h, but experiences a performance loss for speed equal to 130 km/h.

  Performance is evaluated per frame and we consider a system with 512 subcarriers, coded with a convolutional code of rate 1/2 and modulated through a QPSK modulation. We perform the first channel estimation on the preamble OFDM symbol through the LS or the LMMSE algorithm and than we track the channel over the 24 OFDM symbols using the LS, the LMMSE or

[1] For quasi-compliant WiMAX standard simulator we mean a simulator with the parameters given in the standards [1] and [2], but where we do not implement the subcarriers permutation and all the code-modulation combinations.

the adaptive algorithm applied to the pilots. The channel is assumed to be constant in an OFDM symbol and to vary very slowly between two adjacent symbols. These assumptions are correct until the mobile speed is less than 50 km/h, but they are not true anymore when the mobile speed is 130 km/h.

- **Chapter 3** : a way to improve the performance of the system considered in Chapter 2 is given by performing a joint channel estimation and data detection. This can be performed through the Expectation Maximization (EM) algorithm. In this chapter, first the EM algorithm [16] is described and than its application in an OFDM system to hit the mark of jointly estimate the channel and detect the data [17]. This approach can be considered a semi-blind method because known and unknown symbols are simultaneously used to estimate the channel.

  This algorithm is very complex and its usage would be accepted only if the gain over a separated estimation and detection is significant. Unfortunately results shown in this chapter, for the system described in Chapter 2 do not present this significant gain.

- **Chapter 4** : Coding for error correction is one of the many tools available for achieving reliable data transmission in communication systems. For a wide variety of channels, the noise channel coding theorem of the Information Theory proves that, if properly coded, information is transmitted at a rate below channel capacity, then the probability of decoding error can be made to approach zero exponentially with the code length.

  This Chapter is the introduction to the channel coding problem and in particular it gives a quite complete description of the Low-Density Parity-Check (LDPC) codes [3]-[4] and their decoding techniques.

  After their discovery in the early 1960s the LDPC codes were largely forgotten, possibly because computer of the time could not simulate the performance of this codes with meaningful lengths and of the computational complexity for data decoding. In our days, they have drawn much attention due to their near-capacity error correction performance. They have a big disadvantage and a big advantage: the first is the non-existence of a good construction method and the second one is given by the decoding algorithm whose complexity is linear versus the block length. In fact, the decoding operation is always performed with an Iterative Decoding based on a Belief Propagation (IDBP) [18] algorithm which is a symbol-by-symbol soft-in soft-out decoding algorithm which iteratively processes the received symbol in order to improve its reliability, based on the parity-check sums computed from the hard decision of the received symbol. The algorithm is commonly known as Message Passing Algorithm (MPA) [5].

- **Chapter 5** : The LDPC decoding algorithm has low complexity, if compared with the turbo decoding algorithm, used to decode the strong LDPC code

competitors, but actually is still quite complex. This is the reason why in this Chapter a modification to the MPA algorithm is proposed in order to reduce the number of multiplications. Results are quite promising even if the not-a-number cases need to be handled.

Observations made on this modification suggested the study of what would happen if the input to the MPA is quantized rather than the real valued. Once a time, results are shown in this Chapter.

Finally, about the LDPC decoding, we analyzed performance for two different algorithms which transform the decoding problem into a search problem. The first considered algorithm is the Priority First Search Algorithm (PFSA) [6], [7] and [8], proposed for the convolutional codes and more in general for all the codes which can be represented with trellis.

From the parity-check matrix of the LDPC code we can construct the trellis. The problem is that it is very big and the PFSA is not efficient. To deal this problem we propose a variation of the PFSA: we perform the algorithm only on the first k bits, i.e. on the systematic part, than we re-encoded the k bits and chose the codeword with the minimum distance from the found one. Results are shown.

The second proposed algorithm is a genetic algorithm [9], [10]. It applies operations from natural genetics to guide the trek through the search space. The problem here is the definition of the parameters, which are code-based. For the moment a generalization to find out these parameters doesn't exist.

- **Chapter 6** :LDPC codes have very good performance, but also turbo codes [19], [20] and [21] are very good.

  Exactly, turbo codes performs very well for BER's above 10^{-4} (waterfall performance), however they have a significant weakened performance at BER's below 10^{-5}: this phenomenon is known as error floor. Another important observation, we need to keep in our mind, is that, in turbo decoding, only information bits are decoded and they cannot be used for error detection. The poor minimum distance and lack of error detection capability make these codes perform badly in terms of block error probability. Poor block error performance also makes these codes not suitable for many communication applications. On the contrary, finite geometry LDPC codes do not have all the above disadvantages of turbo codes, except that they may not perform as well as turbo codes for BER's above 10^{-4}. Other reasons, which could justify the choice of a LDPC rather than a turbo code are given by a very low complexity decoding, compared to the complexity if the MAP algorithm used for the turbo codes, and the fact that LDPC codes have an inherent interleaving effect and so, if it's concatenated with another code, interleaver is not required.

  Straight conclusion from the above observation is that the advantages of extremely good error performance of turbo codes for BER's above 10^{-4} and the

advantages of finite geometry LDPC codes such as no error floor, possessing error detection capability after decoding and good block error performance, can be combined to form a coding system that performs well for all ranges of SNR's.

Thus, in this Chapter, we propose the concatenation of short LDPC and turbo codes, as inner and outer codes, respectively [22], [23], [24] and [25]. The choice of short codewords has two justifications: first both the codes perform well for long codewords, but this introduce some delay, which is not suitable for some kinds of system transmission, for example speech. Second, if the codeword is relatively short, we analyze the output at the turbo decoding and try to construct an "ad hoc" LDPC decoder which is suitable for decoding what the turbo code cannot correct. This idea is useful also to avoid the block interleaver usage, which is not suitable, because it introduces more delay.

Results for this concatenation scheme are represented for overall rates equal to 1/3, 5/16 and 1/4. The most promising ones are those for the 1/4 rate, the other two cases don't show interesting results and for 1/3 rate obtained with a turbo code 1/2 and a LDPC code 2/3 we observe that the LDPC cannot recover the loss of the performance due to the turbo puncturing.

- **Chapter 7** : briefly describes those parts of the WiMAX standard which have been of interest for the implementation of a downlink simulator standard compliant.

- **Chapter 8** : the parameters used in the simulated system are listed and the results for different channel estimation and coding are represented.

  The considered codes are the standard defined convolutional code and LDPC code applied with the concatenation rule defined for the convolutional code. Beside these standardized codes we plot also the performance for the concatenation scheme with overall rate equal to 1/4.

- **Chapter 9** : conclusions are listed.

Chapter 2

Channel estimation in OFDM systems

In recent years, there has been a lot of interesting applying orthogonal frequency division multiplexing in wireless and mobile communication systems because of its various advantages in lessening the severe effects of frequency selective fading.

Wireless digital communication systems using coherent signaling schemes, such as quadrature amplitude modulation (QAM), require estimation and tracking of the fading channel and so a more complex receiver than for differential modulation schemes, is needed [26].

In particular, in orthogonal frequency division multiplexing (OFDM) systems DPSK is appropriate for relative low data rates, but for more spectrally-efficient OFDM systems, coherent modulation is more appropriate and, in this situation, channel estimation and tracking are required [27]. To this purpose, known symbols, usually called *pilots*, are often multiplexed into the data and channel estimation is performed by interpolation. Channel estimation can be avoided by using differential detection [28], at the cost, however, of a 3 dB loss in signal-to-noise ratio (SNR) [29].

In the design of wireless systems the channel is usually assumed to have a finite-length impulse response. A cyclic extension, longer than this impulse response, is inserted between consecutive symbols in order to avoid inter-symbol interference and preserve the orthogonality of the subcarriers. Generally, the OFDM system is designed so that the cyclic extension is a small percentage of the total symbol length [30].

Several pilot-aided channel estimation schemes for OFDM applications have been investigated, and the ones listed above provide a good sample of the results obtained in this area. In particular, the method proposed in [31] provides channel estimates based on piecewise-constant and piecewise-linear interpolations between pilots. It is simple to implement, but it needs a large number of pilots to get satisfactory performance. In [26], a low-rank approximation to the frequency domain Linear Minimum Mean Squared Error estimator (LMMSE) is proposed, making use of singular value decomposition techniques. The drawback of this approximation is that it requires knowledge of the channel frequency correlation and the operating SNR. In practice, the system can be

designed for fixed values of SNR and channel correlation at the expense of performance losses. The LMMSE studied in [14] exploits channel correlations in time and frequency domains. Like the scheme in [26], it needs knowledge of the channel statistics and the operating SNR. Although it can work in a mismatched mode, its performance degrades if the assumed Doppler frequencies and delay spreads are smaller than the true ones. In [32], channel estimation is performed by two-dimensional interpolation between pilots. Like the method in [14], it is rather robust to Doppler, even though it exhibits performance degradations with lower Doppler frequencies. Similar techniques are presented in [33] and [34]. Finally, [35] investigates the Maximum Likelihood Estimator (MLE). No information on the channel statistics or the operating SNR is required in this scheme.

These methods do not make any assumptions about the channel model, and hence the dimension of the estimation problem can be quite large. However, the radio channel in a wireless communication system is often characterized by the multipath propagation and in large cells with high base station antenna platforms, the multipath propagation is aptly modeled by a few dominant specular paths, typically two to six [36].

A parametric channel model can be used to represent this type of channel, in fact, when the channel correlation matrix is constructed based on this channel model, the signal subspace dimension of the correlation matrix can be effectively reduced and so the channel estimation performance improved [37].

2.1 Channel model

The channel impulse response is treated as a time limited pulse train of the form [28]:

$$g(t,\tau) = \sum_{k=0}^{L-1} \alpha_k(t)\delta(\tau - \tau_k(t)) \tag{2.1}$$

where $\{\alpha_k(t)\}$ are the different path complex gains, $\{\tau_k(t)\}$ are the different path time delays and L is the number of paths. $\{\alpha_k(t)\}$ are wide-sense stationary (WSS) narrowband complex Gaussian processes with the so called Jake's power spectrum [38] and the different path gains are uncorrelated with respect to each other where the average energy of the total channel energy is normalized to one. We observe also that each $\tau_k(t)$, $k = 0, 1, \ldots, L-1$ is smaller than the length of the cyclic prefix, i.e., the entire impulse response lies inside the guard space.

The input/output relation is

$$\begin{aligned} y(t) &= \int d\tau h(t,\tau)x(t-\tau) \\ &= \sum_{k=0}^{L-1} \alpha_k(t)x(t-\tau_k(t)) \end{aligned} \tag{2.2}$$

where $y(t)$ is the received signal at time t and $x(t)$ is the transmitted signal. In the discrete time model, T_c is the sample time and we denote the sampled signals by $y(k) =$

$y(kT_c)$, $h_k(n) = T_c h(kT_c, nT_c)$, $x(n) = x(nT_c)$, so we get:

$$y(k) = \sum_{n=0}^{L-1} h_k(n)x(k-n) + \eta_k \tag{2.3}$$

where $\{\eta_k\}$ is the white Gaussian noise sequence and $\{h_k(n)\}$ is a set of L tap coefficients of an equivalent discrete-time channel impulse response.

2.2 Linear equalization

To perform the estimation and tracking of this kind of channel we could use an adaptive linear filter especially one or more Winner filters [39], [40].

In this section we propose a brief resume of the notions of adaptive linear filtering and adaptive linear equalization existing in literature. These algorithms have been proposed to minimize the performance index of a system and, since in digital communication the meaningful measure of performance is the average probability of error, we want to minimize this performance index.

First of all, considering channels with ISI, we define two parameters:

- Q is the cardinality of the alphabet,

- L is the number of interfering symbols, i.e. it is the channel dispersion length,

then we observe that the equalizer is a linear transversal filter, in fact the computational complexity is a linear function of the channel dispersion length L. The input to this filter is the received signal:

$$y(k) = \sum_{n=0}^{L-1} h_k(n)x(k-n) + \eta_k \tag{2.4}$$

where, as said above, $\{\eta_k\}$ is the white Gaussian noise sequence and $\{h_k(n)\}$ is a set of L tap coefficients of an equivalent discrete-time channel impulse response. We observe that $y(k)$, $k = 0, 1, \ldots$ are the received symbols. The output of the equalizer with coefficients c_j, $j = -K, -K+1, \ldots, K$ is the estimate of the transmitted symbols $x(k)$:

$$\hat{x}(k) = \sum_{j=-K}^{K} c_j y(k-j). \tag{2.5}$$

Unfortunately it has been shown that the dependence of the average error probability from the coefficients c_j is high non-linear, so there are two ways of optimizing the coefficients [41]:

1. peak distortion criterion,

2. mean square error criterion.

2.2.1 Peak distortion criterion

Peak distortion criterion is the worst case inter-symbol interference at the output of the equalizer and the criterion is based on the minimization of this performance index.

We observe that the cascade of the discrete time linear model having an impulse response $h(n)$ and an equalizer having an impulse response c_n can be represented by a single equivalent filter having the impulse response:

$$q(n) = \sum_{j=-\infty}^{+\infty} c_j h(n-j). \tag{2.6}$$

In this case the output is

$$\hat{x}(k) = q_0 x(k) + \sum_{n \neq k} x(n) q(n-k) + \sum_{j=-\infty}^{+\infty} c_j \eta_{k-j} \tag{2.7}$$

where the first term, on the right hand side, is the scaled version of the desired symbol (we should normalize q_0 to 1) and the second is the inter-symbol interference.

If we consider an equalizer with an infinite number of taps, the peak distortion, i.e. the peak value of ISI is:

$$\mathcal{D}(\boldsymbol{c}) = \sum_{n=-\infty, n\neq 0}^{+\infty} |q(n)| \tag{2.8}$$

To obtain $\mathcal{D}(c) = 0$, i.e. to minimize the peak distortion, we must impose:

$$\begin{aligned} q_0 &= 1 \\ q(n) &= 0 \qquad \forall n \neq 0 \end{aligned} \tag{2.9}$$

and using the Z-transformation we obtain:

$$Q(z) = F(z)C(z) = 1 \qquad \Rightarrow \qquad C(z) = \frac{1}{F(z)} \tag{2.10}$$

If the equalizer has a finite length, the peak distortion is:

$$\mathcal{D}(\boldsymbol{c}) = \sum_{n=-K, n\neq 0}^{K+L-1} |q(n)| \tag{2.11}$$

and we observe that it is impossible to completely eliminate ISI and that the peak distortion is a convex function of c_j, so it has a global minimum and no relative minimum.

Nevertheless, if:

$$\mathcal{D}_0 = \frac{1}{|h_0|} \sum_{n=1}^{L} |h(n)| < 1, \tag{2.12}$$

then the ISI is not severe enough to close the eye and so $\mathcal{D}(c)$ is still minimized choosing:

$$\begin{aligned} q_0 &= 1 \\ q(n) &= 0 \qquad 1 \leq |n| \leq K \end{aligned} \tag{2.13}$$

2.2.2 Mean square error criterion

The coefficients c_j are adjusted to minimize the mean square values of the error:

$$\varepsilon_k = x(k) - \hat{x}(k) \tag{2.14}$$

and so the performance index for the MSE criterion is:

$$J = E[|\varepsilon_k|^2] = E[|x(k) - \hat{x}(k)|^2] \tag{2.15}$$

which is a quadratic function of the coefficients c_j.

2.3 Channel estimation

As said in the Introduction to the Chapter, we can consider two different types of estimators: the *Maximum Likelihood* (ML) estimator, which doesn't require the knowledge of the channel statistics or the performing SNR, and the ones which need this knowledge, such as the *Minimum Mean Square Error* (MMSE) estimator and the *List Square* (LS) estimator.

These two types of estimators are based on different assumptions about the Channel Impulse Response (CIR). In the former, the CIR is viewed as a *deterministic but unknown vector*, whereas in the latter, it is regarded as a *random* vector whose particular realization we want to estimate. Correspondingly, the Mean Squared Error (MSE) in the ML estimator is understood as an average over the observed data, whereas in the MMSE estimator, the average is taken not only over the data but over the CIR probability density function as well. It follows that the MMSE estimator has the minimum MSE "on the average", i.e., with respect to all the CIR realizations.

In [27], it is shown that the ML estimator achieves the *Cramér-Rao lower bound* (CRLB) [42], and therefore, it is the minimum-variance unbiased estimator. No further improvement in MSE is possible as long as the CIR is viewed as a deterministic quantity and the estimator is unbiased. On the other hand, the MMSE estimator has prior information on the CIR and can exploit this information to do better than the ML estimator. These considerations prompt one important question: is it conceptually possible for the MMSE estimator to perform below the CRLB? The answer is affirmative since the CRLB is a bound only in the framework of the classical approach to estimation (where CIR is a deterministic quantity). When dealing with MMSE estimator, on the other hand, a Bayesian approach is adopted, and the corresponding estimation accuracy depends on prior information. In principle, performance can be as good as desired, provided that sufficient prior information is available.

Having established the MMSE estimator can do better than the CRLB, purpose of [27] is the understanding of how much better it can do in practice and under which operating conditions. After the comparison of the ML and the MMSE estimators we can say that the main advantage of ML estimator over MMSE estimator is that it does not require knowledge of the channel statistics and the SNR, and therefore, it is simpler to implement. On the other hand, under certain operating conditions, the MMSE estimator has better accuracy as it exploits prior information about the channel. Specifically, the following has been found in [27]:

- The channel estimates at the edges of the bandwidth are worse than those in the middle. A possible remedy is to adopt a denser pilot spacing at the edges.
- MMSE estimator performs better than ML estimator at low SNR.
- At intermediate and high SNRs, the two schemes have comparable performance, provided that the number of pilots is sufficiently larger than the duration of the CIRs. Comparisons have also been made with the estimators proposed in [31], which makes a piece-wise liner interpolation of the estimated CIR, and [32], which holds for a general time-varying channel. It turns out that the loss in performance of the second one with respect to ML estimator and MMSE estimator is limited, whereas that of the first one may be significant, unless the number of pilots is sufficiently high.

As said above, if we assume the knowledge of the time-domain channel statistics, MMSE and LS estimators are the algorithms commonly used. Both estimators have their drawbacks. The MMSE estimator suffers from a high complexity, whereas the LS estimate has a high MSE.

In [27], the authors compared only ML and MMSE estimators and they didn't consider the LS estimator. A justification to this can be found in [43], where it is shown that the MMSE estimate gives $10 - 15$ dB gain in SNR for the same mean square error of channel estimation over LS estimate.

To eliminate the major drawback of the MMSE estimator, the complexity, in [26] a low-rank approximation is applied to the Linear MMSE by using the frequency correlation of the channel. Another option to reduce the complexity of MMSE estimator is given in [44], by deriving an optimal low-rank estimator with singular value decomposition.

Following the three algorithms are briefly described when applied to an OFDM system. It is important to underline that these algorithms can be used both when all the transmitted subcarriers are pilot symbols and when only some of them are pilots, while the others are used to transmit information. This will be well described in Section 2.5.2.

2.3.1 Least-Square (LS) Estimator

At the receiver side of an OFDM system, we have the received sequence $\boldsymbol{y} = [y_0, y_1, \ldots, y_{N-1}]^T$ and the LS estimator for the channel impulse response $\boldsymbol{g}$ minimizes

$(\boldsymbol{y} - \boldsymbol{XFg})^H(\boldsymbol{y} - \boldsymbol{XFg})$ and generates

$$\hat{\boldsymbol{h}}_{LS} = \boldsymbol{FQ}_{LS}\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{y} \tag{2.16}$$

where

$$\boldsymbol{Q}_{LS} = (\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{XF})^{-1}, \tag{2.17}$$

$\boldsymbol{X}$ is a matrix with the transmitted elements of $\boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}]^T$ on its diagonal and

$$\boldsymbol{F} = \begin{bmatrix} W_N^{00} & \cdots & W_N^{0(N-1)} \\ \vdots & \ddots & \vdots \\ W_N^{(N-1)0} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{2.18}$$

is the FFT matrix with

$$W_N^{nk} = \frac{1}{\sqrt{N}} e^{-j2\pi\frac{nk}{N}} \qquad n, k = 0, 1, 2 \ldots, N-1. \tag{2.19}$$

Considering the matrix notation, the received vector $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = \boldsymbol{XFg} + \boldsymbol{n} \tag{2.20}$$

where $\boldsymbol{n}$ is an i.i.d. complex zero-mean Gaussian noise vector.

Note that, reduced to

$$\hat{\boldsymbol{h}}_{LS} = \boldsymbol{X}^{-1}\boldsymbol{y}, \tag{2.21}$$

the LS estimator is equivalent to what is also referred to as the zero-forcing estimator.

2.3.2 Minimum Mean Square Error (MMSE) estimator

If the channel vector $\boldsymbol{g}$ is Gaussian and uncorrelated with the channel noise $\boldsymbol{n}$, the MMSE estimate of $\boldsymbol{g}$ becomes

$$\hat{\boldsymbol{g}}_{MMSE} = \boldsymbol{R}_{\boldsymbol{gy}}\boldsymbol{R}_{\boldsymbol{yy}}^{-1}\boldsymbol{y} \tag{2.22}$$

where

$$\begin{aligned} \boldsymbol{R}_{gy} &= E\{\boldsymbol{gy}^H\} = \boldsymbol{R}_{\boldsymbol{gg}}\boldsymbol{F}^H\boldsymbol{X}^H \\ \boldsymbol{R}_{yy} &= E\{\boldsymbol{yy}^H\} = \boldsymbol{XFR}_{\boldsymbol{gg}}\boldsymbol{F}^H\boldsymbol{X}^H + \sigma_n^2\boldsymbol{I}_N \end{aligned}$$

are the cross-covariance matrix between $\boldsymbol{g}$ and $\boldsymbol{y}$ and the auto-covariance matrix of $\boldsymbol{y}$. Further, $\boldsymbol{R}_{\boldsymbol{gg}}$ is the auto covariance matrix of $\boldsymbol{g}$ and σ_n^2 denotes the noise variance $E[|n_k|^2]$. These two quantities are assumed to be known.

Since the columns in $\boldsymbol{F}$ are orthonormal, $\hat{\boldsymbol{g}}_{MMSE}$ generates the frequency-domain MMSE estimate $\hat{\boldsymbol{h}}_{MMSE}$ by

$$\hat{\boldsymbol{h}}_{MMSE} = \boldsymbol{F}\hat{\boldsymbol{g}}_{MMSE} = \boldsymbol{FQ}_{MMSE}\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{y}, \tag{2.23}$$

where $\boldsymbol{Q}_{MMSE}$ can be shown to be

$$\boldsymbol{Q}_{MMSE} = \boldsymbol{R}_{\boldsymbol{gg}}[(\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{X}\boldsymbol{F})^{-1}\sigma_n^2 + \boldsymbol{R}_{\boldsymbol{gg}}]^{-1}(\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{X}\boldsymbol{F})^{-1}. \tag{2.24}$$

If $\boldsymbol{g}$ is not Gaussian, $\hat{\boldsymbol{h}}_{MMSE}$ is not necessarily a minimum mean square error estimator. It is, however, the best linear estimator in the mean square error sense.

2.3.3 Maximum Likelihood (ML) estimator

The ML estimator is based on the assumption that $\boldsymbol{g}$ is a deterministic but unknown vector. The estimate of $\boldsymbol{g}$ is derived from the linear model [14] and is given by [45]:

$$\hat{\boldsymbol{g}}_{ML} = \boldsymbol{D}^{-1}\boldsymbol{F}^H\hat{\boldsymbol{X}}^{-1}\boldsymbol{y} \tag{2.25}$$

where $\boldsymbol{D}$ is a square matrix

$$\boldsymbol{D} = \boldsymbol{F}^H\boldsymbol{F}.$$

As for the MMSE estimator, the frequency channel estimation is given by

$$\hat{\boldsymbol{h}}_{ML} = \boldsymbol{F}\hat{\boldsymbol{g}}_{ML}. \tag{2.26}$$

2.4 Channel tracking

In most communication systems that employ equalizers, the channel characteristics are unknown a priory and, in many cases, the channel impulse response is time-varying. In such a case, the equalizers are designed to be adjustable to the channel response and, for time-variant channels, to be adaptive to the time variations in the channel impulse response.

For these reasons, algorithms for automatically adjusting the equalizer coefficients to optimize a specified performance index and to adaptively compensate for time variations in the channel characteristics have been studied.

2.4.1 Time domain

Method of steepest descent or gradient descent algorithm

Steepest descent is an old, deterministic method, which is the basis for stochastic gradient based methods. To find the minimum value of the mean-squared error, J_{min}, by the steepest descent algorithm, we proceed as follows:

1. We begin with an initial value $\boldsymbol{w}(0)$ for the tap weight vector, which provides an initial guess as to where the minimum point of the error performance surface may be located. Unless some prior knowledge is available, $\boldsymbol{w}(0)$ is usually set equal to the null vector.

2. Using this initial or present guess, we compute the gradient vector, the real and imaginary parts of which are defined as the derivative of the mean-squared error

$J(n)$, evaluated with respect to the real and imaginary parts of the tap-weight vector $\boldsymbol{w}(n)$ at time n.

3. We compute the next guess at the tap-weight vector by making a change in the initial or present guess in a direction opposite to that of the gradient vector.

4. We go back to step 2 and repeat the process.

It is intuitively reasonable that successive corrections to the tap-weight vector in the direction of the negative of the gradient vector should eventually lead to the minimum mean-squared error J_{min}, at which point the tap-weight vector assumes its optimum values $\boldsymbol{w}_0$.

Thus, we can observe that this is a feedback approach to finding the minimum of the error performance surface and

- error surface must be known

- adaptive approach converses to the optimal solution, also known as Wiener solution, $\boldsymbol{w}_0 = \boldsymbol{R}^{-1}\boldsymbol{p}$ [46] without inverting matrix.

We assume that:

- $\{x(n)\}$ are the WSS input samples

- $\{d(n)\}$ are the WSS desired output

- $\{\hat{d}(n)\}$ are the estimate of the desired signals given by

$$\hat{d}(n) = \boldsymbol{w}^H(n)\boldsymbol{x}(n) \tag{2.27}$$

where $\boldsymbol{x}(n) = [x(n),\, x(n-1),\, \ldots,\, x(n-M+1)]^T$ and $\boldsymbol{w}(n) = [w_0(n),\, w_1(n),\, \ldots,\, w_{M-1}(n)]^T$ is the filter weight vector at time n.

- estimation error:

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n) \\ &= d(n) - \boldsymbol{w}^H(n)\boldsymbol{x}(n) \end{aligned} \tag{2.28}$$

Thus the mean square error (MSE) at time n is

$$\begin{aligned} J(n) &= E[|e(n)|^2] \\ &= \sigma_d^2 - \boldsymbol{w}^H(n)\boldsymbol{p} - \boldsymbol{p}^H\boldsymbol{w}(n) + \boldsymbol{w}^H(n)\boldsymbol{R}\boldsymbol{w}(n) \end{aligned} \tag{2.29}$$

where

- σ_d^2 is the variance of desired signal

- $\boldsymbol{p}$ is the cross-correlation between $\boldsymbol{x}(n)$ and $d(n)$

- $\boldsymbol{R}$ is the correlation matrix of $\boldsymbol{x}(n)$.

When $\boldsymbol{w}(n)$ is set to the optimal Wiener solution, then

$$\boldsymbol{w}(n) = \boldsymbol{w}_0 = \boldsymbol{R}^{-1}\boldsymbol{p} \tag{2.30}$$

and

$$J(n) = J_{min} = \sigma_d^2 - \boldsymbol{p}^H \boldsymbol{w}_0 \tag{2.31}$$

Hence, starting from a point of the error surface, in order to iteratively find $\boldsymbol{w}_0$, we use the method of the steepest descent and the direction in which we change the filter direction is $-\nabla J(n)$, and so:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{2}\mu[-\nabla J(n)] \tag{2.32}$$

or, since $\nabla J(n) = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}(n)$,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}(n)] \tag{2.33}$$

for $n = 0, 1, \ldots$ and μ is called adaptation gain.

Stability of the system and rate of decay

It is important to note that, since steepest descent method uses feedback, the system could be unstable and so we need to guarantee stability with respect to the eigenvalues of $\boldsymbol{R}$. To guarantee the stability we proceed in the following way:

- define the error vector for the tap weights as:

$$\boldsymbol{c}(n) = \boldsymbol{w}(n) - \boldsymbol{w}_0$$

- using $\boldsymbol{p} = \boldsymbol{R}\boldsymbol{w}_0$, the update becomes:

$$\begin{aligned} \boldsymbol{w}(n+1) &= \boldsymbol{w}(n) + \mu[\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}(n)] \\ &= \boldsymbol{w}(n) + \mu[\boldsymbol{R}\boldsymbol{w}_0 - \boldsymbol{R}\boldsymbol{w}(n)] \\ &= \boldsymbol{w}(n) - \mu\boldsymbol{R}\boldsymbol{c}(n) \end{aligned}$$

- from $\boldsymbol{w}(n+1) - \boldsymbol{w}_0 = \boldsymbol{w}(n) - \boldsymbol{w}_0 - \mu\boldsymbol{R}\boldsymbol{c}(n)$ we obtain

$$\begin{aligned} \boldsymbol{c}(n+1) &= \boldsymbol{c}(n) - \mu\boldsymbol{R}\boldsymbol{c}(n) \\ &= [\boldsymbol{I} - \mu\boldsymbol{R}]\boldsymbol{c}(n) \end{aligned}$$

- using the eigenvalues decomposition we can write $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{H}}$ and so

$$\boldsymbol{c}(n+1) = [\boldsymbol{I} - \mu\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{H}}]\boldsymbol{c}(n)$$

- defining $\boldsymbol{v}(n) = \boldsymbol{U}^H \boldsymbol{c}(n)$, we obtain

$$\boldsymbol{v}(n+1) = [\boldsymbol{I} - \mu\boldsymbol{\Lambda}]\boldsymbol{v}(n),$$

so the k-th term in $\boldsymbol{v}(n+1)$ is given by

$$\begin{aligned} v_k(n+1) &= (1-\mu\lambda_k)v_k(n) \\ &= (1-\mu\lambda_k)^n v_k(0) \end{aligned}$$

for $k = 0, 1, \ldots, M-1$, considering λ_k the eigenvalues of $\boldsymbol{R}$.

At this point we can compute the condition for stability and the rate of decay.

For the stability we must satisfy $\lim_{n\to\infty} v_k(n) = 0$ and so it must be $|1-\mu\lambda_k| < 1$ for all k and from this one we can derive the stability condition on μ

$$0 < \mu < \frac{2}{\lambda_{max}},$$

where λ_{max} is the maximum eigenvalues of $\boldsymbol{R}$

We also observe that the k-th mode has a geometric decay and so we can characterize the rate of decay by finding the time it takes to decay to e^{-1} of the initial value. Thus

$$v_k(\tau_k) = (1-\mu\lambda_k)^{\tau_k} v_k(0) = e^{-1} v_k(0) \qquad \Rightarrow \qquad \tau_k = \frac{-1}{\ln(1-\mu\lambda_k)} \approx \frac{1}{\mu\lambda_k} \text{ for } \mu \ll 1 \tag{2.34}$$

The overall rate of decay is:

$$\frac{-1}{\ln(1-\mu\lambda_{max})} \leq \tau \leq \frac{-1}{\ln(1-\mu\lambda_{min})} \tag{2.35}$$

Now we recall the mean square error or cost function $J(n)$ defined in (2.28):

$$\begin{aligned} J(n) &= J_{min} + (\boldsymbol{w}(n) - \boldsymbol{w}_0)^H \boldsymbol{R} (\boldsymbol{w}(n) - \boldsymbol{w}_0) \\ &= J_{min} + (\boldsymbol{w}(n) - \boldsymbol{w}_0)^H \boldsymbol{U}^H \boldsymbol{\Lambda} \boldsymbol{U} (\boldsymbol{w}(n) - \boldsymbol{w}_0) \\ &= J_{min} + \boldsymbol{v}^H(n) \boldsymbol{\Lambda} \boldsymbol{v}(n) \\ &= J_{min} + \sum_{k=0}^{M-1} \lambda_k |v_k(n)|^2 \end{aligned} \tag{2.36}$$

and thus:

$$\lim_{n\to\infty} J(n) = J_{min} \tag{2.37}$$

where J_{min} is defined in (2.31).

Zero forcing algorithm

We have seen that, when $\mathcal{D}_0 < 1$, the $\mathcal{D}(c)$ output of the equalizer is minimized by forcing the equalizer response to be $q_0 = 1$ and $q(n) = 0$, $1 \leq |n| \leq K$. Then the zero forcing solution is achieved by forcing the cross-correlation between the error

sequence $\varepsilon_k = x(k) - \hat{x}(k)$ and the desired sequence $x(k)$ to be zero for shift in the range $0 \leq |n| \leq K$.

A simple recursive algorithm for adjusting the equalizer coefficients is:

$$c_j(k+1) = c_j(k) + \Delta\varepsilon_k x^*(k-j) \tag{2.38}$$

where

- $c_j(k)$ is the jth coefficient at time k
- Δ is the scale factor that controls the rate of adjustment
- $\varepsilon_k = x(k) - \hat{x}(k)$ is the error in the detected symbol.

Least Mean Square (LMS) Algorithm

The error performance surface used by the steepest descent method is not always known a priori, so we can use estimated values. The estimates are random variables and thus this leads to a stochastic approach, in fact the LMS algorithm is an important member of the family of stochastic gradient algorithm.

The term "stochastic gradient" is intended to distinguish the LMS algorithm from the method of steepest descent that uses a deterministic gradient in a recursive computation of the Wiener filter for stochastic inputs.

We will use the following instantaneous estimates that are based on the sample values of the tap-input vector and desired response:

$$\hat{\boldsymbol{R}}(n) = \boldsymbol{x}(n)\boldsymbol{x}^H(n) \tag{2.39}$$

$$\hat{\boldsymbol{p}}(n) = \boldsymbol{x}(n)d^*(n) \tag{2.40}$$

At this point we remember that the steepest descent update is:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{2}\mu[-\nabla(J(n))] \tag{2.41}$$

where the gradient of the error surface at $\boldsymbol{w}(n)$ was shown to be

$$\nabla(J(n)) = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}(n), \tag{2.42}$$

but here we use the instantaneous estimates, so we obtain

$$\begin{aligned}
\hat{\nabla}(J(n)) &= -2\boldsymbol{x}(n)d^*(n) + 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\hat{\boldsymbol{w}}(n) \\
&= -2\boldsymbol{x}(n)[d^*(n) - \boldsymbol{x}^H(n)\hat{\boldsymbol{w}}(n)] \\
&= -2\boldsymbol{x}(n)[d^*(n) - \hat{d}^*(n)] \\
&= -2\boldsymbol{x}(n)e^*(n)
\end{aligned} \tag{2.43}$$

where $e^*(n)$ is the complex conjugate of estimate error.

Putting (2.43) in the update equation (2.41), we obtain:

$$\begin{aligned}\hat{\boldsymbol{w}}(n+1) &= \hat{\boldsymbol{w}}(n) + \mu \boldsymbol{x}(n)[d^*(n) - \boldsymbol{x}^H(n)\hat{\boldsymbol{w}}(n)] \\ &= \hat{\boldsymbol{w}}(n) + \mu \boldsymbol{x}(n)e^*(n). \end{aligned} \tag{2.44}$$

Thus LMS algorithm belongs to the family of stochastic gradient algorithms, in fact, the update is extremely simple while the instantaneous estimates may have large variance, the LMS algorithm is recursive and effectively averages these estimates.

Stability and performance analysis of LMS algorithm

The LMS algorithm, from a statistical point of view, can be analyzed by invoking the independence theory, which states:

1. the vectors $\boldsymbol{x}(1)$, $\boldsymbol{x}(2)$, ..., $\boldsymbol{x}(n)$ are statistically independent vectors
2. at time n, $\boldsymbol{x}(n)$ is independent on $d(1)$, $d(2)$, ..., $d(n-1)$
3. at time n, $d(n)$ is statistically dependent on $\boldsymbol{x}(n)$, but it is statistically independent on $d(1)$, $d(2)$, ..., $d(n-1)$
4. $\boldsymbol{x}(n)$ and $d(n)$ are mutually Gaussian distributed random variables for all n.

The independence theorem is justified in some cases, i.e. beamforming where we receive independent vector observations. In other cases it is not well justified, but allows the analysis to proceeds.

Observing Equation. (2.44) we note that the tap weight vector $\hat{\boldsymbol{w}}(n+1)$ at time $n+1$ depends only on three inputs:

1. the previous sample vectors of the input process, $\boldsymbol{x}(n)$, $\boldsymbol{x}(n-1)$, ..., $\boldsymbol{x}(1)$
2. the previous samples of the desired response, $d(n)$, $d(n-1)$, ..., $d(1)$
3. the initial value of the tap-weight vector, $\hat{\boldsymbol{w}}(0)$.

Using the independence theory we can show that $\boldsymbol{w}(n)$ converges to the optimal solution in the mean, i.e.:

$$\lim_{n\to\infty} E[\boldsymbol{w}(n)] = \boldsymbol{w}_0. \tag{2.45}$$

where $\boldsymbol{w}_0$ is the Wiener solution.

The starting point is the (2.44)[41]:

$$\begin{aligned}\boldsymbol{w}(n+1) - \boldsymbol{w}_0 &= \boldsymbol{w}(n) - \boldsymbol{w}_0 + \mu \boldsymbol{x}(n)e^*(n) \\ &= [\boldsymbol{I} - \mu \boldsymbol{x}(n)\boldsymbol{x}^H(n)]\boldsymbol{c}(n) + \mu \boldsymbol{x}(n)e_0^*(n) \end{aligned} \tag{2.46}$$

Now, note that since $\boldsymbol{w}(n)$ is based on past inputs desired responses $d(1)$, ..., $d(n-1)$, $\boldsymbol{w}(n)$ (and $\boldsymbol{c}(n)$) is independent on $\boldsymbol{x}(n)$, thus $\mu E[\boldsymbol{x}(n)e_0^*(n)] = 0$ and so:

$$E[\boldsymbol{c}(n+1)] = (\boldsymbol{I} - \mu \boldsymbol{R})E[\boldsymbol{c}(n)]. \tag{2.47}$$

Reasoning as explained for the case of the steepest descent algorithm we find that:

$$\lim_{n\to\infty} E[\boldsymbol{c}(n)] = 0 \qquad \text{if} \qquad 0 < \mu < \frac{2}{\lambda_{max}} \tag{2.48}$$

which is equivalent to write

$$\lim_{n\to\infty} E[\boldsymbol{w}(n)] = \boldsymbol{w}_0 \qquad \text{if} \qquad 0 < \mu < \frac{2}{\lambda_{max}}. \tag{2.49}$$

Noting that

$$\lambda_{max} \leq \text{trace}[\boldsymbol{R}] = Nr(0) = N\sigma_x^2 \tag{2.50}$$

a more conservative bound is

$$0 < \mu < \frac{2}{N\sigma_x^2} \tag{2.51}$$

and observing that also convergence in the mean (2.45) is a weak condition that says nothing about the variance, which may even grow.

A stronger condition is the convergence in the mean square, which says:

$$\lim_{n\to\infty} E[|\boldsymbol{c}(n)|^2] = \text{constant} \tag{2.52}$$

and it is equivalent to show that:

$$\lim_{n\to\infty} J(n) = \lim_{n\to\infty} E[|e(n)|^2] = \text{constant}. \tag{2.53}$$

If we write $e(n)$ as [41]:

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n) \\ &= e_0(n) - \boldsymbol{c}^H(n)\boldsymbol{x}(n) \end{aligned} \tag{2.54}$$

then

$$\begin{aligned} J(n) &= E[|e(n)|^2] \\ &= J_{min} + E[\boldsymbol{c}^H(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{c}(n)] \\ &= J_{min} + J_{ex}(n) \end{aligned} \tag{2.55}$$

Since $J_{ex}(n)$ is a scalar, we can write [41]

$$\begin{aligned} J_{ex}(n) &= E[\boldsymbol{c}^H(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{c}(n)] \\ &= \text{trace}[\boldsymbol{R}\boldsymbol{K}(n)] \end{aligned} \tag{2.56}$$

where $\boldsymbol{K}(n) = E[\boldsymbol{c}(n)\boldsymbol{c}^H(n)]$.

If we define $\boldsymbol{S}(n) \triangleq \boldsymbol{U}^H\boldsymbol{K}(n)\boldsymbol{U}$, where $\boldsymbol{U}$ is the unitary matrix obtained from the eigenvalues decomposition of $\boldsymbol{R}$ and remembering that $\boldsymbol{\Lambda}$ is a diagonal matrix, we find:

$$J_{ex}(n) = \text{trace}[\boldsymbol{\Lambda S(n)}] = \sum_{i=1}^{M} \lambda_i s_i(n), \tag{2.57}$$

where $s_1(n)$, $s_2(n)$, ..., $s_M(n)$ are the diagonal elements of $\boldsymbol{S}(n)$.

The recursion expression (2.44) can be modified to yield a recursion on $\boldsymbol{S}(n)$, which is:

$$\boldsymbol{S}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{\Lambda})\boldsymbol{S}(n)(\boldsymbol{I} - \mu\boldsymbol{\Lambda}) + \mu^2 J_{min}\boldsymbol{\Lambda} \tag{2.58}$$

and its diagonal elements are

$$s_i(n+1) = (1 - \mu\lambda_i)^2 s_i(n) + \mu^2 J_{min}\lambda_i, \qquad i = 1, 2, \ldots, M \tag{2.59}$$

Suppose $J_{ex}(n)$ converges, i.e. $s_i(n+1) = s_i(n)$, so, from this equality, we find

$$s_i(n) = \frac{\mu J_{min}}{2 - \mu\lambda_i}, \qquad i = 1, 2, \ldots, M \tag{2.60}$$

and then

$$\lim_{n\to\infty} J_{ex}(n) = J_{min} \sum_{i=0}^{M-1} \frac{\mu\lambda_i}{2 - \mu\lambda_i} \tag{2.61}$$

where

$$\mathcal{M} = \sum_{i=0}^{M-1} \frac{\mu\lambda_i}{2 - \mu\lambda_i} \tag{2.62}$$

is the LMS misadjustment and if its value is 10% or less it is generally considered acceptable.

Normalized LMS algorithm

In the standard LMS algorithm the correction is proportional to $\mu\boldsymbol{x}(n)e^*(n)$:

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \mu\boldsymbol{x}(n)e^*(n) \tag{2.63}$$

and if $\boldsymbol{x}(n)$ is large, the LMS algorithm experiences a gradient noise amplification problem. The normalized LMS algorithm seeks to avoid gradient noise amplification: the adaptation gain is made time varying, $\mu(n)$, and optimized to minimize error, i.e., choose $\mu(n)$, such that the updated $\hat{\boldsymbol{w}}(n+1)$ produces the minimum MSE $J(n+1) = E[|e(n+1)|^2]$, where $e(n+1) = d(n+1) - \boldsymbol{w}^H(n+1)\boldsymbol{x}(n+1)$.

The optimal adaptation gain, $\mu_0(n)$ will be a function of $\boldsymbol{R}$ and $\nabla(n)$. To compute it we expand $J(n+1)$ and differentiate it with respect to $\mu(n)$ and set it equal to 0.

After some computation we find:

$$\mu_0(n) = \frac{\nabla^H(n)\nabla(n)}{\nabla^H(n)\boldsymbol{R}\nabla(n)} \tag{2.64}$$

and using the instantaneous estimates:

$$\begin{aligned} \hat{\boldsymbol{R}} &= \boldsymbol{x}(n)\boldsymbol{x}^H(n) \\ \hat{\nabla}(n) &= -2\boldsymbol{x}(n)e^*(n) \end{aligned}$$

we obtain

$$\mu_0(n) = \frac{1}{\boldsymbol{x}^H(n)\boldsymbol{x}(n)} = \frac{1}{\|\boldsymbol{x}(n)\|^2} \tag{2.65}$$

and the NLMS update is:

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \frac{\tilde{\mu}}{a + \|\boldsymbol{x}(n)\|^2}\boldsymbol{x}(n)e^*(n) \tag{2.66}$$

where $a > 0$ is an offset added to avoid problems when $\|\boldsymbol{x}(n)\|^2 \approx 0$.

Following a reason similar to that used for steepest descent method we find that NLMS update will converge if

$$0 < \tilde{\mu} < 2. \tag{2.67}$$

To conclude this section we observe that:

- the NLMS has a simpler convergence criteria than the LMS
- the NLMS generally converges faster than the LMS algorithm.

The LMS equalizer

Defining

- Γ the covariance matrix of the input sequence $y(k)$,
- $\boldsymbol{C}$ the vector of the equalizer coefficients,
- ξ the vector of the cross-correlation between the input $y(k)$ and the equalizer coefficients c_k

we find that for the minimization of the MSE we must choose:

$$\boldsymbol{C}_{opt} = \Gamma^{-1}\boldsymbol{H}. \tag{2.68}$$

The method of the steepest descent offers an alternative to the computation of Γ^{-1}. We start choosing a vector $\boldsymbol{C}$ said $\boldsymbol{C}_0$, which corresponds to some point on the quadratic MSE surface, then we compute the gradient vector:

$$G_0 = \Big[\frac{1}{2}\frac{\partial J}{\partial C_{0k}}\Big], \qquad k = -K, -K+1, \ldots, -1, 0 \ \ldots, K \tag{2.69}$$

At this point each tap is changed in the opposite to its corresponding gradient component:

$$C_{k+1} = C_k - \Delta G_k, \tag{2.70}$$

$$G_k = \frac{1}{2}\frac{\partial J}{\partial C_k} = \Gamma C_k - \xi = -E[\varepsilon_k \boldsymbol{X}_k^*] \tag{2.71}$$

where $\boldsymbol{X}_k^*$ is the vector of the complex conjugated received signal that makes up the estimate $\hat{x}(k)$.

It is interesting to make some observation on the convergence properties of this algorithm.

First of all we observe that Γ is an Hermitian matrix and so, using the eigenvalues decomposition, we can write $\Gamma = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^H$ where $\boldsymbol{U}$ is a unitary matrix and $(\cdot)^H$ denotes the Hermitian transpose.

It can be seen [41] that the convergence is ensured by:

$$0 < \Delta < \frac{2}{\lambda_{max}} \tag{2.72}$$

where λ_{max} is the maximum eigenvalue of $\boldsymbol{\Gamma}$.

Recursive least-square (RLS) algorithm for adaptive equalization

As we have seen, the gradient algorithm has only a single adjustable parameter for controlling the convergence rate: Δ, and so the slowly convergence is due to it. Consequently, we need more complex algorithms to obtain faster convergence.

In this case, the performance index is expressed in terms of a time average instead of statistical average.

Before explaining the algorithm, we give some definitions:

- estimation of the transmitted symbol at sample n:

$$\hat{x}(n) = \sum_{j=-K}^{K} c_j(n-1)y(n-j), \qquad j = 0,\, 1,\, \ldots,\, N-1 \tag{2.73}$$

- $v(n) = y(n+K)$

- the estimate $\hat{x}(n)$ becomes:

$$\begin{aligned} \hat{x}(n) &= \sum_{j=0}^{N-1} c_j(n-1)v(n-j) \\ &= \boldsymbol{C}_N(n-1)\boldsymbol{V}_N(n) \end{aligned} \tag{2.74}$$

where $\boldsymbol{C}_N(n-1)$ is the row vector of the equalizer at sample $n-1$ and $\boldsymbol{V}_N(n) = [v(n),\, v(n-1),\, \ldots,\, v(n-N+1)]^T$ is the input signal to the equalizer.

Recursive least square algorithm

Following we describe the recursive least-square (RLS) algorithm, also known as Kalman algorithm.

We have observed the vectors $\boldsymbol{V}_N(n)$ for $n = 0, 1, \ldots, t$ and we wish to determine the coefficient vector $\boldsymbol{C}_N(t)$ that minimizes the time average weighted square error:

$$\varepsilon_N^{LS} = \sum_{n=0}^{t} w^{t-n} |e_N(n,t)|^2 \tag{2.75}$$

where w is the weighting factor ($0 < w < 1$) and $e_N(n,t) = x(n) - \boldsymbol{C}_N(t)\boldsymbol{V}_N(n)$ is the error between the estimated and the transmitted symbol at sample n. In this expression we have introduced exponential weighting into past data, appropriated when the channel characteristics are time-variant.

The minimization of ε_N^{LS} with respect to the coefficients $\boldsymbol{C}_N(t)$ yields the set of linear equations:

$$\boldsymbol{R}_N(t)\boldsymbol{C}_N(t) = \boldsymbol{D}_N(t) \tag{2.76}$$

where:

- $\boldsymbol{R}_N(t) = \sum_{n=0}^{t} w^{t-n} \boldsymbol{V}_N^H(n)\boldsymbol{V}_N(t)$ is the signal correlation matrix (not a Toeplitz matrix)
- $\boldsymbol{D}_N(t) = \sum_{n=0}^{t} w^{t-n} x(n)\boldsymbol{V}_N^*(t)$ is the cross-correlation vector.

The solution is:

$$\boldsymbol{C}_N(t) = \boldsymbol{R}_N^{-1}(t)\boldsymbol{D}_N(t). \tag{2.77}$$

The recursive algorithm

Now, we have $\boldsymbol{C}_N(t-1)$ and we wish to compute $\boldsymbol{C}_N(t)$. The algorithm's steps are:

- $\boldsymbol{R}_N(t)$ may be computed recursively, so the time-update equation for $\boldsymbol{R}_N(t)$ is:

$$\boldsymbol{R}_N(t) = w\boldsymbol{R}_N(t-1) + \boldsymbol{V}_N^H(t)\boldsymbol{V}_N(t) \tag{2.78}$$

- recursive computation for $\boldsymbol{R}_N^{-1}(t)$ is

$$\boldsymbol{R}_N^{-1} = \frac{1}{w}\Big[\boldsymbol{R}_N^{-1}(t-1) - \frac{\boldsymbol{R}_N^{-1}(t-1)\boldsymbol{V}_N^H(t)\boldsymbol{V}_N(t)\boldsymbol{R}_N^{-1}(t-1)}{w + \boldsymbol{V}'_N(t)\boldsymbol{R}_N^{-1}(t-1)\boldsymbol{V}_N^*(t)}\Big] \tag{2.79}$$

- for convenience we define $\boldsymbol{P}_N(t) = \boldsymbol{R}_N^{-1}(t-1)$
- we define also Kalman gain vector:

$$\boldsymbol{K}_N(t) = \frac{1}{w + \mu_N(t)}\boldsymbol{P}_N(t-1)\boldsymbol{V}_N^*(t) \tag{2.80}$$

where $\mu_N(t) = \boldsymbol{V}_N^H(t)\boldsymbol{P}_N(t-1)\boldsymbol{V}_N(t)$

- $\Rightarrow \boldsymbol{P}_N(t) = \frac{1}{w}[\boldsymbol{P}_N(t-1) - \boldsymbol{K}_N(t)\boldsymbol{V}'_N(t)\boldsymbol{P}_N(t-1)]$

- we observe that $\boldsymbol{P}_N(t)\boldsymbol{V}_N^*(t) = \boldsymbol{K}_N(t)$

- now we use the matrix inversion identity to compute $\boldsymbol{C}_N(t)$ from $\boldsymbol{C}_N(t-1)$:

  - $\boldsymbol{C}_N(t) = \boldsymbol{P}_N(t)\boldsymbol{D}_N(t)$
  - $\boldsymbol{D}_N(t) = w\boldsymbol{D}_N(t-1) + x(t)\boldsymbol{V}_N^*(t)$
  - $\boldsymbol{C}_N(t) = \boldsymbol{C}_N(t-1) + \boldsymbol{K}_N(t)[x(t) - \boldsymbol{V}_N^T(t)\boldsymbol{C}_N(t-1)]$, where $\boldsymbol{V}_N^T(t)\boldsymbol{C}_N(t-1)$ is the output of the equalizer at time t, $\hat{x}(t)$
  - $e_N(t, t-1) = x(t) - \hat{x}(t) \equiv e_N(t)$

$$\Rightarrow \boldsymbol{C}_N(t) = \boldsymbol{C}_N(t-1) + \boldsymbol{K}_N(t)e_N(t) \tag{2.81}$$

- The residual MSE is:

$$\xi_{N,min}^{LS} = \sum_{n=0}^{t} w^{t-n}|x(n)|^2 - \boldsymbol{C}_N(t)\boldsymbol{D}_N^*(t) \tag{2.82}$$

If we suppose to know $\boldsymbol{C}_N(t-1)$, $\boldsymbol{P}_N(t-1)$ and $\boldsymbol{V}_N(t)$, then the algorithm's steps are:

- compute $\hat{x}(t) = \boldsymbol{V}_N^T(t)\boldsymbol{C}_N(t-1)$

- compute $e_N(t) = x(t) - \hat{x}(t)$

- compute $\boldsymbol{K}_N(t) = \frac{1}{w+\mu_N(t)}\boldsymbol{P}_N(t-1)\boldsymbol{V}_N^*(t)$

- compute $\boldsymbol{P}_N(t) = \frac{1}{w}[\boldsymbol{P}_N(t-1) - \boldsymbol{K}_N(t)\boldsymbol{V}_N'(t)\boldsymbol{P}_N(t-1)]$

- update coefficients $\boldsymbol{C}_N(t) = \boldsymbol{C}_N(t-1) + \boldsymbol{K}_N(t)e_N(t)$

This algorithm has two principal disadvantage:

1. computational complexity,

2. sensitivity to round off noise that accumulates due to recursive computations.

2.4.2 Frequency domain

Recognizing that the Fourier transform maps time-domain signals into the frequency domain and that the inverse Fourier transform provides the inverse mapping that takes us back into time domain, it is equally feasible to perform the adaptation of filter parameters in the frequency domain. In such a case we speak of *Frequency-Domain Adaptive Filtering* (FDAF).

There are two main reasons for seeking the use of frequency domain adaptive filtering in one form or another [41]:

1. In certain application, such as acoustic echo cancellation in teleconferencing, for example, the adaptive filter is required to have a long impulse response to cope with an equally long echo duration. When the LMS algorithm is adapted in the time domain, we find that the requirement of a long memory results in a significant increasing of the computational complexity of the algorithm. How then do we deal with this problem? There are two options available to us. We may choose an infinite-duration impulse response (IIR) filter and adapt it in the time domain; the difficulty with this approach is that we inherit a new problem, namely, the *filter instability*. Alternatively, we may use a particular type of frequency-domain adaptive filtering that combines two complementary methods widely used in digital signal processing:

   - block implementation of a FIR filter, which allows the efficient use of parallel processing and thereby results in a gain in computational speed,
   - fast Fourier transform (FFT) algorithms for performing fast convolution (filtering) which permits adaptation of filter parameters in the frequency domain in a computationally efficient manner.

   This approach to frequency-domain adaptive filtering builds on the so-called block LMS algorithm that includes the standard LMS algorithm as a special case.

2. Frequency-domain adaptive filtering, mechanized in a different way from that described under point 1, is used to improve the convergence performance of the standard LMS algorithm.

Block adaptive filters

In a block adaptive filter, the incoming data sequence $x(n)$ is sectioned into B-point blocks by means of a serial to parallel converter, and the blocks of input data so produced are applied to a FIR filter of length M, one block at a time. Generally we assume $M = L$ as we see at the end of this subsection. The adaptation of the filter proceeds on a block-by-block basis rather than on a sample-by-sample basis as in the standard LMS algorithm. Let k refer to a block time and $\hat{\boldsymbol{w}}(k)$ denote the tap-weight vector of the filter for the k-th block:

$$\hat{\boldsymbol{w}}(k) = [\hat{w}_0(k),\, \hat{w}_1(k),\, \ldots,\, \hat{w}_{M-1}(k)]^T \qquad k = 0,\, 1,\, \ldots \tag{2.83}$$

The index n is reserved for the original sample time, written in terms of the block size as follows:

$$n = kB + i, \qquad i = 0,\, 1,\, \ldots,\, M-1 \qquad k = 0,\, 1,\, \ldots \tag{2.84}$$

Let the input signal vector $\boldsymbol{x}(n)$ at time n be written as:

$$\boldsymbol{x}(n) = [x(n),\, x(n-1),\, \ldots,\, x(n-M+1)]^T. \tag{2.85}$$

Accordingly, at time n the output $\hat{y}(n)$ produced by the filter in response to the input signal vector $\boldsymbol{x}(n)$ is defined by the inner product:

$$\hat{y}(n) = \hat{\boldsymbol{w}}^T(k)\boldsymbol{x}(n). \tag{2.86}$$

or equivalently:

$$\begin{aligned} y(kB+i) &= \hat{\boldsymbol{w}}^T(k)\boldsymbol{x}(kB+i) \\ &= \sum_{l=0}^{M-1} \hat{w}_l(k)x(kB+i-l), \qquad i = 0,\, 1,\, \ldots,\, M-1. \end{aligned} \tag{2.87}$$

Let $y(n) = y(kB+i)$ denote the corresponding value of the desired response. An error $e(n)$ is produced by comparing the filter output $\hat{y}(n)$ against the desired response $y(n)$; the error signal is defined by:

$$e(n) = y(n) - \hat{y}(n) \tag{2.88}$$

or equivalently

$$e(kB+i) = y(kB+i) - \hat{y}(kB+i). \tag{2.89}$$

Recognizing that in the block LMS algorithm the error signal is allowed to vary at the sampling rate, it follows that for each block of data we have different values of the error signal for use in the adaptive process. Accordingly, for the k-th block, we may sum the product $\boldsymbol{x}(kB+i)e(kB+i)$ over all possible value of i, and so define the following update equation for the tap-weight vector of the block LMS algorithm operating on real-valued data:

$$\hat{\boldsymbol{w}}(k+1) = \hat{\boldsymbol{w}}(k) + \mu \sum_{i=0}^{B-1} \boldsymbol{x}(kM+i)e(kM+i) \tag{2.90}$$

where μ is the *adaptation gain*. For convenience of presentation we define

$$\boldsymbol{\phi}(k) = \sum_{i=0}^{B-1} \boldsymbol{x}(kM+i)e(kM+i), \tag{2.91}$$

and so we rewrite (2.90) in the form:

$$\hat{\boldsymbol{w}}(k+1) = \hat{\boldsymbol{w}}(k) + \mu\boldsymbol{\phi}(k). \tag{2.92}$$

A distinctive feature of the block LMS algorithm described herein is that its design incorporates an averaged estimate of the gradient vector, as shown by

$$\hat{\nabla}(k) = -\frac{2}{B} \sum_{i=0}^{B-1} \boldsymbol{x}(kB+i)e(kB+i) \tag{2.93}$$

where the factor 2 is included to be consistent with the definition of the gradient vector

[1] and the factor $1/B$ is included for $\hat{\nabla}(k)$ to be an unbiased time average. Then in terms of $\hat{\nabla}(k)$ we may reformulate the block LMS algorithm as follows:

$$\hat{\boldsymbol{w}}(k+1) = \hat{\boldsymbol{w}}(k) - \frac{1}{2}\mu_B \hat{\nabla}(k) \tag{2.94}$$

where μ_B may be viewed as the *effective adaptation gain* of the block LMS algorithm; it is defined by:

$$\mu_B = B\mu. \tag{2.95}$$

For the convergence properties of the block LMS algorithm we observe that they are similar to those of the standard LMS algorithm, in that they both attempt to minimize the same mean-square error function:

$$J = \frac{1}{2}E[e^2(n)]. \tag{2.96}$$

It can be shown that the block LMS algorithm uses a more accurate estimate of the gradient vector because of the time averaging, with the estimation accuracy increasing as the block size B is increased. However, this improvement does not imply faster adaptation. We may proceed through a convergence analysis of the block LMS algorithm in a manner similar to that used for conventional LMS algorithm [41].

We may thus summarize the convergence properties of the block LMS algorithm as follows:

1. **Condition for convergence**. The mean of the tap-weighted vector $\hat{\boldsymbol{w}}(k)$ computed by using the block LMS algorithm converges to the optimum Wiener solution $\boldsymbol{w}_0$ as the number of block iterations k approaches infinity, as shown by

$$\lim_{k\to\infty} E[\hat{\boldsymbol{w}}(k)] = \boldsymbol{R}^{-1}\boldsymbol{p} = \boldsymbol{w}_0 \tag{2.97}$$

where

$$\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}^H(n)] \tag{2.98}$$

$$\boldsymbol{p} = E[\boldsymbol{x}(n)y(n)]. \tag{2.99}$$

[1] We define the gradient operation ∇ the k-th element of which is written in terms of first-order partial derivatives with respect to the real part a_k and the imaginary part b_k, for the k-th filter coefficient, as

$$\nabla_k = \frac{\partial}{\partial a_k} + j\frac{\partial}{\partial b_k}.$$

The gradient vector $\nabla J(n)$ is given by:

$$\begin{aligned}\nabla J(n) &= \begin{bmatrix} \frac{\partial J(n)}{\partial a_0(n)} + j\frac{\partial J(n)}{\partial b_0(n)} \\ \frac{\partial J(n)}{\partial a_1(n)} + j\frac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \frac{\partial J(n)}{\partial a_{M-1}(n)} + j\frac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} \\ &= -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}(n)\end{aligned}$$

The condition that has to be satisfied by the adaptation gain μ for convergence of the block LMS algorithm in the mean value is

$$0 < \mu < \frac{2}{B\lambda_{max}} \tag{2.100}$$

where B is the block size and λ_{max} is the largest eigenvalue of the correlation matrix $\boldsymbol{R}$ of the input signal vector $\boldsymbol{x}(n)$.

2. **Misadjustment**. Remembering the definition of the excess mean-squared error $J_{ex}(k)$ and the minimum mean-squared error J_{min}, we note that for the $J_{ex}(k)$ computed by the block LMS algorithm to converge to a constant value $J_{ex}(k) < J_{min}$ as the number of block iterations k approaches infinity, the adaptation gain μ has to satisfy the more stringent condition:

$$0 < \mu < \frac{2}{B\sum_{i=1}^{M}\lambda_i} \tag{2.101}$$

and the corresponding value of the misadjustment is

$$\mathcal{M} = \frac{\mu}{2}\sum_{i=1}^{M}\lambda_i. \tag{2.102}$$

Comparing the results described here for the block LMS algorithm with the corresponding results for LMS algorithm 2.4.1, we may make the following observations when operating in a wide-sense stationary environment:

- The converged mean weight vector and misadjustment of the block LMS algorithm are identical to those of the standard LMS algorithm. The same holds for the average time constant.

- For an input signal vector $\boldsymbol{x}(n)$ whose correlation matrix $\boldsymbol{R}$ has a prescribed eigenstructure, the condition imposed on the block LMS for convergence in the mean square is more restrictive than the corresponding condition for the standard LMS algorithm. In particular the tighter bound on the adaptation gain μ may cause the block LMS algorithm to converge more slowly than the standard LMS algorithm, particularly when the eigenvalue spread $\chi = \frac{\lambda_{max}}{\lambda_{min}}$ of the correlation matrix $\boldsymbol{R}$ is small.

We can conclude this section with an observation about the block size. The operation of the block LMS algorithm holds true for any integer value of $B \geq 1$. Nevertheless, the option of choosing the block size B equal to the filter length M is preferred in most applications of block adaptive filtering. This choice may be justified on the following grounds:

- when $B > M$, redundant operations are involved in the adaptive process, because then the estimation of the gradient vector uses more information than the filter itself

- when $B < M$, some of the tap weights in the filter are wasted, because the sequence of tap inputs is not long enough to feed the whole filter.

It thus appears that the most practical choice is $B = M$.

Fast LMS algorithm

The description of the fast LMS algorithm presented here uses the overlap-save method [2] with 50% overlap.

According to this method, the M tap weights of the filter are padded with an equal number of zeros, and an N-point FFT is used for the computation, where $N = 2M$. Thus let the $N \times 1$ vector $\hat{\boldsymbol{W}}(k)$ denote the FFT coefficients of the zero-padded, tap-weight vector $\hat{\boldsymbol{w}}(k)$, as follows:

$$\hat{\boldsymbol{W}}(k) = FFT \begin{bmatrix} \hat{\boldsymbol{w}}(k) \\ \mathbf{0} \end{bmatrix} \tag{2.103}$$

where $\mathbf{0}$ is the $M \times 1$ null vector and $FFT[\cdot]$ denotes the fast Fourier transformation.

Let $\boldsymbol{X}(k)$ denote an $N \times N$ diagonal matrix derived from the input data as following:

$$\boldsymbol{X}(k) = \text{diag}\{FFT[\underbrace{x(kM-M),\ \ldots,\ x(kM-1)}_{(k-1)\text{th block}}, \underbrace{x(kM),\ \ldots,\ x(kM+M-1)}_{k\text{th block}}]\}. \tag{2.104}$$

Now, applying the overlap-save method to the linear convolution of (2.87) yields the $M \times 1$ vector

$$\begin{aligned} \hat{\boldsymbol{y}}^T &= [\hat{y}(kM),\ \hat{y}(kM+1),\ \ldots,\ \hat{y}(kM+M-1)] \\ &= \text{last M elements of } IFFT[\boldsymbol{X}(k)\hat{\boldsymbol{W}}(k)]. \end{aligned} \tag{2.105}$$

where $IFFT[\cdot]$ denotes inverse fast Fourier transformation and only the last M elements are retained, because the first M elements correspond to a circular convolution.

Now define the $M \times 1$ desired response vector

$$\boldsymbol{y}(k) = [y(kM),\ y(kM+1),\ \ldots,\ y(kM+M-1)]^T \tag{2.106}$$

and the corresponding $M \times 1$ error signal vector:

$$\begin{aligned} \boldsymbol{e}(k) &= [e(kM),\ e(kM+1),\ \ldots,\ e(kM+M-1)]^T \\ &= \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k). \end{aligned} \tag{2.107}$$

[2] Implementing convolution using the DFT, the overlap-save method involves overlapping input section rather than output sections. If the overlap is of $P-1$ samples, the first $P-1$ samples of each output sequence are ignored, because they are due to the wraparound (end) effect of the circular convolution.

We may transform the error signal vector $\boldsymbol{e}(k)$ into the frequency domain as follows:

$$\boldsymbol{E}(k) = FFT \begin{bmatrix} \mathbf{0} \\ \boldsymbol{e}(k) \end{bmatrix}. \tag{2.108}$$

At this point we observe that:

$$\boldsymbol{\phi}(k) = \text{first } M \text{ elements of } IFFT[\boldsymbol{X}^H(k)\boldsymbol{E}(k)]. \tag{2.109}$$

The update equation in frequency domain becomes:

$$\hat{\boldsymbol{W}}(k+1) = \hat{\boldsymbol{W}}(k) + \mu FFT \begin{bmatrix} \boldsymbol{\phi}(k) \\ \mathbf{0} \end{bmatrix} \tag{2.110}$$

Computational complexity

Consider the standard LMS algorithm with M tap weights operating on real data. In this case, M multiplications are performed to compute the output and a further M multiplications are performed to update the tap weights, making for a total of $2M$ multiplications per iteration. Hence, for a block of M output samples, the total number of multiplications is $2M^2$.

Consider next, the fast LMS algorithm. Each N-point FFT requires approximately $N \log_2 N$ real multiplications, where $N = 2M$. According to the structure of the fast LMS algorithm, there are five frequency transformations performed, which therefore account for $5N \log_2 N$ multiplications. In addition the computation of the frequency domain output vector requires $4N$ multiplications, and so does the computation of the cross-correlations relating to the gradient vector estimation. Hence, the total corresponding number of multiplications performed in the fast LMS algorithm is:

$$5N \log_2 N + 8N = 10M \log_2 M + 26M, \tag{2.111}$$

so if we compute the complexity ratio we find:

$$\text{Complexity ratio} = \frac{5 \log_2 M + 13}{M} < 1 \tag{2.112}$$

and then we can say that fast LMS algorithm is faster than standard LMS algorithm, as soon as $M \geq 64$.

Unconstrained frequency-domain adaptive algorithm

The fast LMS algorithm may be viewed as a constrained form of frequency-domain adaptive filtering. The time domain constraint consists of the following operations:

- discarding the last M elements of the inverse FFT of $\boldsymbol{X}^H(k)\boldsymbol{E}(k)$
- replacing the elements so discarded by a block of M zeros before reapplying the FFT.

In the unconstrained frequency-domain adaptive filter the gradient constraint is removed completely and the implementation becomes simpler:

$$\hat{\boldsymbol{W}}(k+1) = \hat{\boldsymbol{W}}(k) + \mu \boldsymbol{X}^H(k)\boldsymbol{E}(k). \tag{2.113}$$

It is important to note, however, that the estimate of the gradient vector computed here no longer corresponds to a linear correlation as specified in (2.93); rather, we now have a circular correlation.

Consequently, we find that in general the unconstrained frequency-domain adaptive filtering algorithm deviates from the fast LMS algorithm, in that the tap weight vector no longer converges to the Wiener solution as the number of block iterations approaches infinity. Another important point is that although the convergence rate of the unconstrained frequency-domain adaptive filtering algorithm is increased with time-varying adaptation gain, the improvement is offset by a worsening of the misadjustment. Indeed, the unconstrained algorithm requires twice as many iterations as the constrained algorithm to produce the same level of misadjustment.

2.5 System description

In Figure 2.1 the OFDM base band model considered in this work is represented. We assume that the use of the cyclic prefix both preserves the orthogonality of the subcarriers and eliminates inter-symbol interference (ISI) between consecutive OFDM symbols.

The channel $g(t, \tau)$ is assumed to be slowly fading, so it is considered to be constant during one OFDM symbol. The number of subcarriers in the system is N, and the cyclic prefix is made of N_{CP} samples.

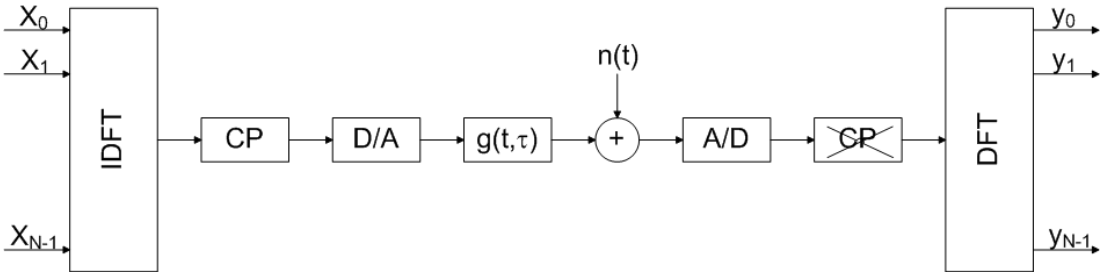

Figure 2.1: Base band model of an OFDM system. CP denotes the cyclic prefix.

Under these assumptions we can describe the system as a set of parallel Gaussian channels, with correlated attenuation h_k, $k = 0, 1, \ldots, N-1$.

In matrix notation we describe the OFDM system as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{h} + \boldsymbol{n} \tag{2.114}$$

where $\boldsymbol{y}$ is the received vector, $\boldsymbol{X}$ is a matrix containing, on its diagonal, the transmit-

ted signaling points, taken from a multi-amplitude signal constellation, $\boldsymbol{h}$ is a channel attenuation vector and $\boldsymbol{n}$ is a vector of i.i.d. complex, zero-mean, Gaussian noise with variance σ_n^2.

The channel impulse response, as explained in Section 2.1 is treated as a time limited pulse train of the form:

$$g(t,\tau) = \sum_{k=0}^{L-1} \alpha_k(t)\delta(\tau - \tau_k(t))$$

where $\{\alpha_k(t)\}$ are the different path complex gain, $\{\tau_k(t)\}$ are the different path time delays and L is the number of paths. $\{\alpha_k(t)\}$ are wide-sense stationary (WSS) narrow-band complex Gaussian processes with the so called Jake's power spectrum [38] and the different path gains are uncorrelated with respect to each other where the average energy of the total channel energy is normalized to one. We observe also that each $\tau_k(t)$, $k = 0, 1, \ldots, L-1$ is minor than the length of the cyclic prefix, i.e., the entire impulse response lies inside the guard space.

The transmitted data are divided into frame which is composed of N subcarriers and N_{sym} OFDM symbols. The first OFDM symbol is a preamble, i.e. its symbols are known at the receiver and are employed to obtain a channel estimation through some estimation algorithm. The following OFDM symbols are made of some left and right null subcarriers, while the remaining ones are pilot and data subcarriers. Obviously, the pilot subcarriers are known at the receiver and are introduced to perform the channel tracking.

2.5.1 Channel estimation

We will analyze several estimators based on the scheme presented in Figure 2.2

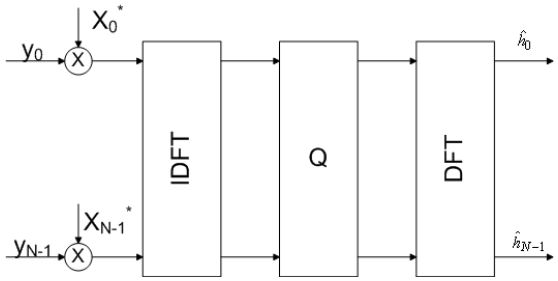

Figure 2.2: General scheme used in many channel estimation algorithms.

Least Square (LS) estimator

The LS estimator for the frequency response $\hat{\boldsymbol{h}}_{LS}$ is given by:

$$\hat{\boldsymbol{h}}_{LS} = \boldsymbol{X}^{-1}\boldsymbol{y} \tag{2.115}$$

and from (2.135) it is clear why it is also referred to as the zero forcing estimator.

The LS estimator is described in Section 2.3.1; here we remind it generates the estimated frequency response [43]:

$$\hat{\boldsymbol{h}} = \boldsymbol{F}\boldsymbol{Q}_{LS}\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{y} \tag{2.116}$$

where $\boldsymbol{Q}_{LS} = (\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{X}\boldsymbol{F})^{-1}$ and $\boldsymbol{F}$ is the $N \times N$ DFT matrix, whose elements are $[\boldsymbol{F}]_{n,k} = W_N^{nk} = \frac{1}{\sqrt{N}} \exp(-j2\pi\frac{nk}{N})$.

At this point we observe that the equivalence between (2.135) and (2.116) holds when we suppose to know all the tones in an OFDM symbol, because $\boldsymbol{F}^H\boldsymbol{F} = \boldsymbol{F}\boldsymbol{F}^H = \boldsymbol{I}$.

Minimum Mean Square Error (MMSE) estimator

If the channel vector $\boldsymbol{g}$ is Gaussian and uncorrelated with the channel noise $\boldsymbol{n}$, the MMSE estimate of $\boldsymbol{g}$, described in Section 2.3.2, becomes:

$$\hat{\boldsymbol{g}}_{MMSE} = \boldsymbol{R}_{\boldsymbol{g}\boldsymbol{y}}\boldsymbol{R}_{\boldsymbol{y}\boldsymbol{y}}^{-1}\boldsymbol{y} \tag{2.117}$$

where

$$\begin{aligned} \boldsymbol{R}_{\boldsymbol{g}\boldsymbol{y}} &= E[\boldsymbol{g}\boldsymbol{y}^H] = \boldsymbol{R}_{\boldsymbol{g}\boldsymbol{g}}\boldsymbol{F}^H\boldsymbol{X}^H \\ \boldsymbol{R}_{\boldsymbol{y}\boldsymbol{y}} &= E[\boldsymbol{y}\boldsymbol{y}^H] = \boldsymbol{X}\boldsymbol{F}\boldsymbol{R}_{\boldsymbol{g}\boldsymbol{g}}\boldsymbol{F}^H\boldsymbol{X}^H + \sigma_n^2\boldsymbol{I} \end{aligned}$$

are the cross-covariance matrix between $\boldsymbol{g}$ and $\boldsymbol{y}$ and the auto-covariance matrix of $\boldsymbol{y}$, respectively. Further, $\boldsymbol{R}_{\boldsymbol{g}\boldsymbol{g}}$ is the auto-covariance matrix of $\boldsymbol{g}$ and σ_n^2 denotes the noise variance $E[|n_k|^2]$. For the moment these two quantities are assumed to be known. Since the columns in $\boldsymbol{F}$ are orthonormal, $\hat{\boldsymbol{g}}_{MMSE}$ generates the frequency domain MMSE estimate $\hat{\boldsymbol{h}}_{MMSE}$ by

$$\hat{\boldsymbol{h}}_{MMSE} = \boldsymbol{F}\hat{\boldsymbol{g}}_{MMSE} = \boldsymbol{F}\boldsymbol{Q}_{MMSE}\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{y} \tag{2.118}$$

where $\boldsymbol{Q}_{MMSE}$ can be shown to be [43]

$$\boldsymbol{Q}_{MMSE} = \boldsymbol{R}_{\boldsymbol{g}\boldsymbol{g}}[(\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{X}\boldsymbol{F})^{-1}\sigma_n^2 - \boldsymbol{R}_{\boldsymbol{g}\boldsymbol{g}}]^{-1}(\boldsymbol{F}^H\boldsymbol{X}^H\boldsymbol{X}\boldsymbol{F})^{-1}. \tag{2.119}$$

We observe that the MMSE estimator requires the calculation of an $N \times N$ matrix $\boldsymbol{Q}_{MMSE}$, which implies a high complexity when N is large. A straightforward way of decreasing the complexity is to reduce the size of $\boldsymbol{Q}_{MMSE}$.

Linear minimum mean square (LMMSE) estimator

We start considering the LS estimate:

$$\hat{\boldsymbol{h}}_{LS} = \boldsymbol{X}^{-1}\boldsymbol{y} = \boldsymbol{h} + \tilde{\boldsymbol{n}}$$

where $\tilde{\boldsymbol{n}} = \boldsymbol{X}^{-1}\boldsymbol{n}$ is a vector of independent Gaussian noise variables with covariance matrix[3]

$$\boldsymbol{R}_{\tilde{\boldsymbol{n}}\tilde{\boldsymbol{n}}} = \sigma_n^2(\boldsymbol{X}\boldsymbol{X}^H)^{-1}. \tag{2.120}$$

The LS estimate is a noisy observation of the channel attenuation and can be smoothed using correlation properties of the channel. The optimal linear estimator in terms of mean square error (MSE) is

$$\hat{\boldsymbol{h}} = \boldsymbol{W}_X\hat{\boldsymbol{h}}_{LS}, \tag{2.121}$$

where

$$\boldsymbol{W}_X = \boldsymbol{R}_{\boldsymbol{h}\boldsymbol{h}}(\boldsymbol{R}_{\boldsymbol{h}\boldsymbol{h}} + \sigma_n^2(\boldsymbol{X}\boldsymbol{X}^H)^{-1})^{-1}, \tag{2.122}$$

and $\boldsymbol{R}_{\boldsymbol{h}\boldsymbol{h}} = E[\boldsymbol{h}\boldsymbol{h}^H]$ is the auto-covariance matrix of the channel vector $\boldsymbol{h}$.

At this point we recognize that the weighting matrix $\boldsymbol{W}_X$, of size $N \times N$, depends on the transmitted data $\boldsymbol{X}$.

As a first step towards low complexity estimators we want to find a weighting matrix that is independent on the transmitted data. This can be obtained by considering $\hat{\boldsymbol{h}}_{LS}$ to be our observation and derive an LMMSE estimator that considers $\boldsymbol{X}$ to be stochastic with independent and uniformly distributed constellation points. In this case we define a constellation factor $\beta = E[|x_k|^2]E[|x_k|^{-2}]$ and remembering that the signal-to-noise ratio is defined as $SNR \triangleq E[|x_k|^2]/\sigma_n^2$ we find

$$\boldsymbol{R}_{\tilde{\boldsymbol{n}}\tilde{\boldsymbol{n}}} = \frac{\beta}{SNR}\boldsymbol{I}. \tag{2.123}$$

Now, the LMMSE estimate of the $\hat{\boldsymbol{h}}_{LMMSE}$ from $\hat{\boldsymbol{h}}_{LS}$ becomes

$$\hat{\boldsymbol{h}}_{LMMSE} = \boldsymbol{W}\hat{\boldsymbol{h}}_{LS} \tag{2.124}$$

where the fixed weighting matrix is given by

$$\boldsymbol{W} = \boldsymbol{R}_{\boldsymbol{h}\boldsymbol{h}}\Big(\boldsymbol{R}_{\boldsymbol{h}\boldsymbol{h}} + \frac{\beta}{SNR}\boldsymbol{I}\Big)^{-1} \tag{2.125}$$

The LMMSE estimator still requires N multiplications per estimated attenuation and we use it both as a reference and as a starting point in the derivation of the DFT-based low complexity estimators.

To derive the new estimators we observe that $\boldsymbol{h}$ is the sampled frequency response of a channel with short time duration compared to the OFDM symbol length and, hence, its associated cyclic impulse response $\boldsymbol{g} = IDFT(\boldsymbol{h})$ has only a few taps with significant power. If we perform the estimation in the time domain, we can reduce the complexity of the estimation by using this power concentration.

This prompts the estimator structure in Figure 2.2, where the block named Q represents a time elaboration of the signal, as it is evident from the presence of the DFT

[3] $\boldsymbol{R}_{\tilde{\boldsymbol{n}}\tilde{\boldsymbol{n}}} = E[(\boldsymbol{X}^{-1}\boldsymbol{n})^H(\boldsymbol{X}^{-1}\boldsymbol{n})] = \mathrm{trace}(E[\boldsymbol{n}\boldsymbol{n}^H(\boldsymbol{X}\boldsymbol{X}^H)^{-1}]) = \sigma_n^2(\boldsymbol{X}\boldsymbol{X}^H)^{-1}$

and IDFT blocks. The LS estimate is transformed into its time domain equivalent $\hat{\boldsymbol{g}}_{LS} = IDFT(\hat{\boldsymbol{h}}_{LS})$, remembering that when we are using a number of pilot subcarriers (N_p) less than the number of total subcarriers (N) the equivalent time domain response is given by (2.134). The smoothing is then performed by a linear transformation:

$$\hat{\boldsymbol{g}} = \boldsymbol{Q}\hat{\boldsymbol{g}}_{LS} \tag{2.126}$$

and the result is transformed back to the frequency domain: $\hat{\boldsymbol{h}} = DFT(\hat{\boldsymbol{g}})$.

The important benefit of this estimator structure in terms of complexity is the low complexity of the DFT/IDFT (implemented as fast transformations) and the time domain power concentration, which offers a simplification of (2.126), without sacrificing too much in performance.

The scope of the following consideration is to find sparse approximations of the LMMSE estimator's equivalent time domain smoothing matrix

$$\boldsymbol{Q} = \boldsymbol{F}^H \boldsymbol{W} \boldsymbol{F} \tag{2.127}$$

where $\boldsymbol{F}$ is the DFT matrix and $\boldsymbol{W}$ is defined in (2.125). A straightforward way is to simply ignore the coefficients in $\hat{\boldsymbol{g}}_{LS}$ that contain more noise than channel power and only transform the remaining elements back to the frequency domain.

Now, we present three different low complexity estimators, obtained reducing the number of non-zero elements in the time domain matrix multiplication (2.126), with the aim of reducing the computational complexity and preserving the performance.

Before the description of the algorithms it is important to underline a drawback of these estimators: for all of them we suppose the knowledge of the channel length.

Estimator A

The simplest idea is to choose the L coefficients in $\hat{\boldsymbol{g}}_{LS}$ that have the highest channel power, where L is the length of the channel impulse response and thus it is much smaller than N (the number of subcarriers) and N_p (the number of pilots).

In this case we find

$$\boldsymbol{Q}_A = \begin{bmatrix} \boldsymbol{Q}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.128}$$

where

$$\boldsymbol{Q}_{L\times L} = R_{gg,L}\Big(R_{gg,L} + \frac{\beta}{SNR}\boldsymbol{I}\Big)^{-1} \tag{2.129}$$

and $R_{gg,L}$ is the upper left $L \times L$ corner of R_{gg}.

Estimator B Further reductions in complexity can be done by ignoring cross correlation between the L chosen taps in $\hat{\boldsymbol{g}}_{LS}$ and only weighting them individually. In this case the matrix $\boldsymbol{Q}$ becomes

$$\boldsymbol{Q}_B = \begin{bmatrix} \boldsymbol{D}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.130}$$

where

$$\boldsymbol{D}_{L\times L} = \mathrm{diag}(\delta_0, \delta_1, \ldots, \delta_{L-1}) \tag{2.131}$$

and

$$\delta_k = \frac{\gamma_k}{\gamma_k + \frac{\beta}{SNR}}, \qquad k = 0, 1, \ldots, L-1$$

and $\gamma_k = E[|g_k|^2]$ which are the diagonal elements of $\boldsymbol{R_{gg}} = E[\boldsymbol{g}\boldsymbol{g}^H] = \boldsymbol{F}^H\boldsymbol{R_{hh}}\boldsymbol{F}$.

Estimator C In this case we reduce the complexity using the L chosen coefficients directly as input to the DFT. The $\boldsymbol{Q}$ matrix simply becomes

$$\boldsymbol{Q}_c = \begin{bmatrix} \boldsymbol{I}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.132}$$

It is interesting to underline that in this case we obtain

$$\hat{\boldsymbol{h}}_C = \boldsymbol{F}\hat{\boldsymbol{g}}_{LS} \tag{2.133}$$

where $\hat{\boldsymbol{g}}_{LS}$ is defined in (2.134) and, when we are in presence of Gaussian white noise, as in this case, this is the maximum likelihood (ML) estimator ([47] and [48]).

In table 2.1 we summarize the linear transformation and the required multiplication per attenuation coefficient (h_k).

Estimator	**Linear Transformation**	**Required mult./attenuation**
LMMSE	N.A.	$N+1$
A	$\boldsymbol{Q}_A = \begin{bmatrix} \boldsymbol{Q}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix}$	$\log_2 N + \frac{L^2}{N} + 1$
B	$\boldsymbol{Q}_B = \begin{bmatrix} \boldsymbol{D}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix}$	$\log_2 N + \frac{L}{N} + 1$
C	$\boldsymbol{Q}_C = \begin{bmatrix} \boldsymbol{I}_{L\times L} & 0 \\ 0 & 0 \end{bmatrix}$	$\log_2 N + 1$
LS	N.A.	1

Table 2.1: Linear transformations and computational complexity for the proposed estimator.

Until this moment we develop the discussion with no attention about the number of tones we know at the receiver. If we know all the OFDM symbol the channel impulse response can be computed using the complete DFT matrix, but when we know only some tones, the pilots, we must use a truncated version of the DFT matrix. To avoid the problem of this distinction during our description we use the notation $\boldsymbol{F}$ for both the truncated or complete version of the DFT matrix. Particular attention must be paid when we talk about complete DFT matrix: we know the channel impulse response has only few significant taps and so if we assume the channel length L known, then the DFT matrix is said to be complete even if it has only the first L columns. This assumption avoids the problem of selecting the significant taps. A way to avoid the truncation of the $\boldsymbol{F}$ matrix we can choose the first L taps of the estimated impulse channel response and then perform a zero-padding.

After these observations, we define the channel impulse response by

$$\hat{\boldsymbol{g}_{LS}} = (\boldsymbol{F}^H\boldsymbol{F})^{-1}\boldsymbol{F}^H\hat{\boldsymbol{h}}_{LS}, \tag{2.134}$$

in fact, if we come back to the frequency domain through the expression $\hat{\boldsymbol{h}}_{LS} = \boldsymbol{F}\hat{\boldsymbol{g}}_{LS} = \boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{F})^{-1}\boldsymbol{F}^H\hat{\boldsymbol{h}}_{LS}$ we find the matrix $\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{F})^{-1}\boldsymbol{F}^H$ is equal to the identity one only when we consider the entire DFT matrix and not a truncated version, i.e. when the pilot tones are only a fraction of subcarriers ($N_p \leq N$).

2.5.2 Use of the described algorithm considering only pilot subcarriers

Often, in OFDM systems there are some subcarriers, i.e. the pilot subcarriers, whose number N_p is smaller than the total number of subcarriers N, which are known at the receiver.

In this section we use the following notation:

- $\boldsymbol{F}_L$ is a $N \times L$ matrix containing the first L columns of the DFT matrix $\boldsymbol{F}$
- $\boldsymbol{F}_{L,p}$ is a $N_p \times L$ matrix obtained from $\boldsymbol{F}_L$ eliminating the lines not corresponding to pilot subcarriers
- $\boldsymbol{y}_p$ is the vector of the training received symbols
- $\boldsymbol{X}_p$ is a $N_p \times N_p$ diagonal matrix containing the training symbols.

LS estimator

In the frequency domain the LS estimate is given by

$$\hat{\boldsymbol{h}}_{LS} = \boldsymbol{X}_p^{-1}\boldsymbol{y}_p \tag{2.135}$$

and in the time domain it is

$$\hat{\boldsymbol{g}}_{LS} = (\boldsymbol{F}_{L,p}^H\boldsymbol{F}_{L,p})^{-1}\boldsymbol{F}_{L,p}^H\hat{\boldsymbol{h}}_{LS} \tag{2.136}$$

MMSE estimator

The expression for the MMSE estimator considering only pilot subcarriers is

$$\begin{aligned}\hat{\boldsymbol{g}}_{MMSE} &= \boldsymbol{R}_{\boldsymbol{gg}}\boldsymbol{F}_{L,p}^H\boldsymbol{X}_p^H(\boldsymbol{X}_p\boldsymbol{F}_{L,p}\boldsymbol{R}_{\boldsymbol{gg}}\boldsymbol{F}_{L,p}^H\boldsymbol{X}_p^H + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{y}_p \\ &= \left(\boldsymbol{R}_{\boldsymbol{gg}}^{-1} + \boldsymbol{F}_{L,p}^h\boldsymbol{X}_p^H\frac{1}{\sigma_n^2}\boldsymbol{X}_p\boldsymbol{F}_{L,p}\right)^{-1}\boldsymbol{F}_{L,p}^H\boldsymbol{X}_p^H\frac{1}{\sigma_n^2}\boldsymbol{y}_p\end{aligned} \tag{2.137}$$

LMMSE estimator

We define the auto-covariance matrix of the frequency response channel as

$$\boldsymbol{R}_{\boldsymbol{hh}} = E[\boldsymbol{h}\boldsymbol{h}^H] = \boldsymbol{F}_{L,p}\boldsymbol{R}_{\boldsymbol{gg}}\boldsymbol{F}_{L,p}^H. \tag{2.138}$$

where $\boldsymbol{R_{gg}}$ is the auto-covariance matrix of the channel impulse response and it is $\boldsymbol{R_{gg}} = \mathrm{diag}(1/L,\, 1/L,\, \ldots,\, 1/L)$.

At this point there are the two version of the LMMSE estimator: one depending on the training symbols:

$$\hat{\boldsymbol{h}}_{LMMSE} = \boldsymbol{R_{hh}}(\boldsymbol{R_{hh}} + \sigma_n^2(\boldsymbol{X}_p\boldsymbol{X}_p^H)^{-1})^{-1}\hat{\boldsymbol{h}}_{LS} \tag{2.139}$$

and the other one independent on the training symbols:

$$\hat{\boldsymbol{h}}_{LMMSE} = \boldsymbol{R_{hh}}\Big(\boldsymbol{R_{hh}} + \frac{\beta}{SNR}\boldsymbol{I}\Big)^{-1}\hat{\boldsymbol{h}}_{LS} \tag{2.140}$$

Estimator A Here, we define the matrix $\boldsymbol{R_{gg},}_{LS} = \boldsymbol{F}_{L,p}^H\boldsymbol{R_{hh}}\boldsymbol{F}_{L,p}$ and compute the matrix $\boldsymbol{Q}_A$

$$\boldsymbol{Q}_A = \boldsymbol{R_{gg},}_{LS}\Big(\boldsymbol{R_{gg},}_{LS} + \frac{\beta}{SNR}\boldsymbol{I})^{(-1)} \tag{2.141}$$

and so we find

$$\hat{\boldsymbol{h}}_A = \boldsymbol{F}_{L,p}\hat{\boldsymbol{g}}_A = \boldsymbol{F}_{L,p}\boldsymbol{Q}_A\hat{\boldsymbol{g}}_{LS} \tag{2.142}$$

where $\hat{\boldsymbol{g}}_{LS}$ is computed in (2.136).

Estimator B In this estimator we consider the values on the diagonal of the $\boldsymbol{R_{gg},}_{LS}$ matrix (named γ_k, $k = 0,\, 1,\, \ldots,\, L-1$) and $\boldsymbol{Q}_B$ is a diagonal matrix with the following values

$$q_{B,kk} = \frac{\gamma_k}{\gamma_k + \frac{\beta}{SNR}} \tag{2.143}$$

and so the estimated frequency response is

$$\hat{\boldsymbol{h}}_B = \boldsymbol{F}_{L,p}\hat{\boldsymbol{g}}_B = \boldsymbol{F}_{L,p}\boldsymbol{Q}_B\hat{\boldsymbol{g}}_{LS} \tag{2.144}$$

Estimator C Finally, in this case the matrix $\boldsymbol{Q}_C$ is simply an $L \times L$ identity matrix, so we find $\hat{\boldsymbol{g}}_C = \hat{\boldsymbol{g}_{LS}}$ and in the frequency domain

$$\hat{\boldsymbol{h}}_C = \boldsymbol{F}_{L,p}\hat{\boldsymbol{g}}_C = \boldsymbol{F}_{L,p}\hat{\boldsymbol{g}}_{LS} \tag{2.145}$$

which is also the expression of the ML estimator.

2.5.3 Channel tracking

The algorithms presented in Section 2.4 allow the equalization of a channel, for a single carrier system, in the time domain.

If we consider an OFDM system we can try to track a time-varying channel using, in the frequency domain, the same idea which is behind the RLS algorithm.

The basic idea is to perform channel estimation on the first OFDM symbol by employing a linear minimum mean square error (LMMSE) or a least square error (LS) algorithm. Then, to perform the tracking we can use an adaptive filter whose working principle is the same as RLS algorithm.

Since we have an OFDM system the first idea is to perform the tracking in the frequency domain. The adaptive filter in frequency domain are described in Section 2.4.2, but we observe that our case is a little bit different from the ones proposed there. It is for this reason that we refer to [15] in the following analysis.

In [15] the authors consider a block of N input symbols and perform the FFT on them obtaining $\boldsymbol{X}(n)$, they perform also the FFT on blocks of N desired symbols obtaining $\boldsymbol{Y}(n)$ and then update the frequency response of the adaptive filter $\boldsymbol{W}(n)$ in the frequency domain. The k-th complex weight for the k-th frequency is then updated according to:

$$W_k(n+1) = W_k(n) + \mu E_k(n) X^*(n), \tag{2.146}$$

where $\boldsymbol{E}(n) = \boldsymbol{Y}(n) - \boldsymbol{W}(n)\boldsymbol{X}(n)$. We observe that this is similar to the unconstrained frequency-domain adaptive filter.

In Figure 2.3 the scheme adopted for the tracking of the channel is shown.

In the frequency domain, we define:

- $\boldsymbol{Y}_p(k)$ the vector of the symbol received on pilot subcarriers at time k
- $\boldsymbol{H}_p(k)$ the vector of the real channel attenuation coefficients on pilot subcarriers at time k
- $\hat{\boldsymbol{H}}_{\boldsymbol{p}}(k)$ the vector of the estimated channel attenuation coefficients on pilot subcarriers at time k
- $\boldsymbol{X}_p(k)$ the transmitted pilot signal at time k
- $\tilde{\boldsymbol{Y}}_{\boldsymbol{p}}(k) = \hat{\boldsymbol{H}}_{\boldsymbol{p}}(k)\boldsymbol{X}_p(k)$ is the estimate of the received symbol
- $\boldsymbol{E}_p(k) = \boldsymbol{Y}_p(k) - \tilde{\boldsymbol{Y}}_{\boldsymbol{p}}(k)$ is the vector of the errors between the received symbols and their estimates on pilot subcarriers

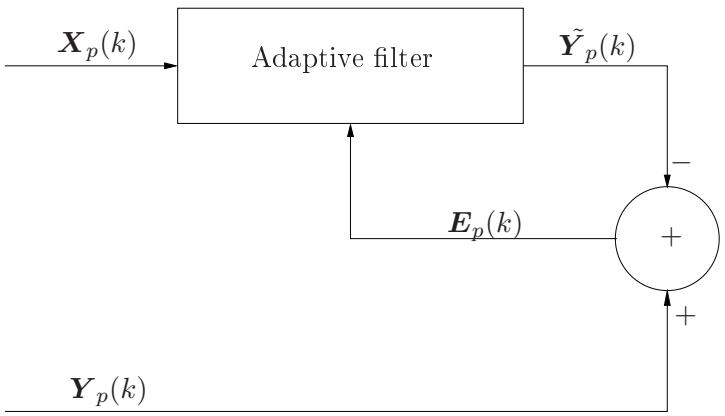

Figure 2.3: Adaptive scheme for channel tracking in frequency domain.

Since in each OFDM symbol there are N_p pilot subcarriers we consider them and find the update equation for the channel estimation at every pilot:

$$\hat{H}_i(k+1) = \hat{H}_i(k) + \mu E_i(k) X_i^*(k), \qquad i = 0, 1, \ldots, N_p - 1 \tag{2.147}$$

where μ is the *adaptation gain*.

In this way we know the channel only on a subset of subcarriers, so, to have an estimation of the all frequency channel response we transform the channel into the time domain, eliminate the taps with high noise and re-transform it to frequency domain considering all the N subcarriers and we obtain the frequency channel response estimation $\hat{\boldsymbol{H}}(k+1)$.

Choice of the adaptation gain

Parameter μ has the same mean of the parameter Δ in the LMS algorithm (2.38) which must satisfy the following condition:

$$0 < \Delta < \frac{2}{\lambda_{max}}, \tag{2.148}$$

where λ_{max} is the maximum eigenvalue of the matrix correlation of the input vector.

At this point we recall and use a result provided in [15]. Starting from Equation. (2.146) the authors observe that since each weight is adapted only once for each N-point data block, the number of adaptations required to obtain output data similar to those obtained with the conventional time domain filter is reduced by a factor of N and so the value of the adaptation gain μ may accordingly be increased by a factor of N.

For this reason, considering that we perform the adaptive of the filter on N_p subcarriers, we define

$$\mu = \frac{N_p}{\delta_{max}}, \tag{2.149}$$

where δ_{max} is the maximum eigenvalue of the correlation matrix of the input signal $\boldsymbol{X}_p$.

2.6 Simulation setup

A brief description of the considered system was given in Section 2.5 and Figure 2.1.

Simulation results are obtained assuming we are working in the frequency domain, so Figure 2.1 can be changed as follows:

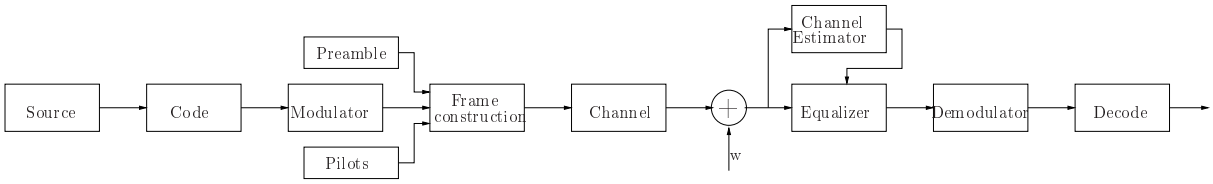

Figure 2.4: Simulator scheme.

2.6.1 Code

The code is the convolutional code defined in the WiMAX standard [1]-[2]. The generated bits are encoded in blocks of 288; we took this decision because the concatenation rule for a QPSK rate 1/2 code leads to the coding of blocks of this size.

Each FEC block is encoded by the binary convolutional encoder, which shall have native rate 1/2, and a constraint length equal to $k = 7$ and shall use the following generator polynomials codes to derive its two code bits:

$$G_1 = 171_{\mathrm{OCT}} \quad X \tag{2.150}$$

$$G_2 = 133_{\mathrm{OCT}} \quad Y \tag{2.151}$$

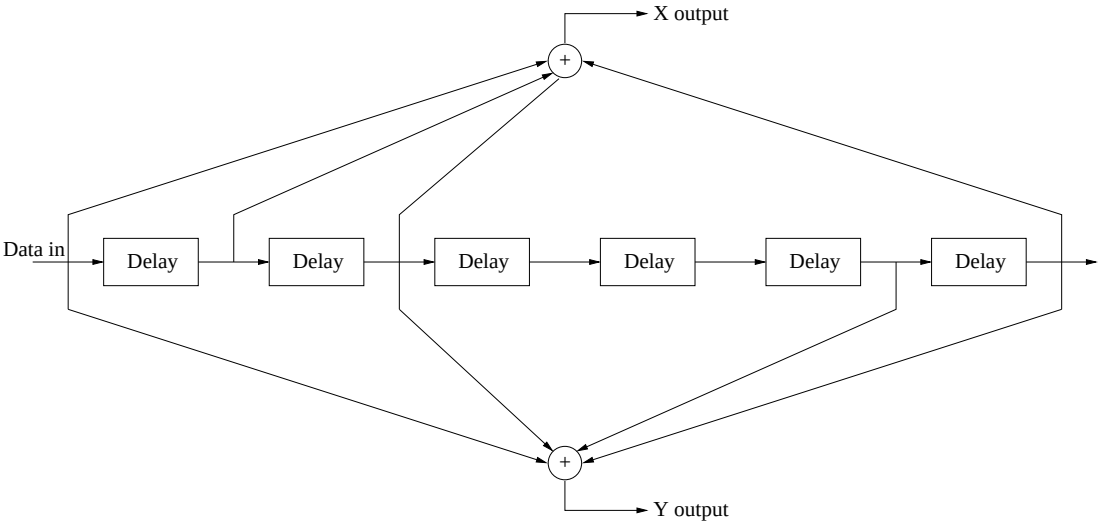

Figure 2.5: Convolutional encoder of rate 1/2.

Each FEC block is encoded by a *tail-biting convolutional encoder*, which is achieved by initializing the encoder's memory with the last data bits of the FEC block being encoded.

2.6.2 Frame construction

Also in the construction of the downlink frame we consider the IEEE 802.16e standard and the frame structure is represented in Figure 2.6.

We construct the preamble and pilot sequences standard compliant, but we change the way of assembling data and pilots. We don't considering any permutation system, but assume the pilots uniformly distributed between data subcarriers.

On the other hand , we keep unchanged the number of virtual, data and pilot subcarriers.

Preamble

The first symbol of the downlink transmission is the preamble. There are three types of preamble carrier-sets, those are defined by allocation of different subcarriers for each one of them and those subcarriers are modulated using a boosted BPSK modulation with specific Pseudo-Noise (PN) code.

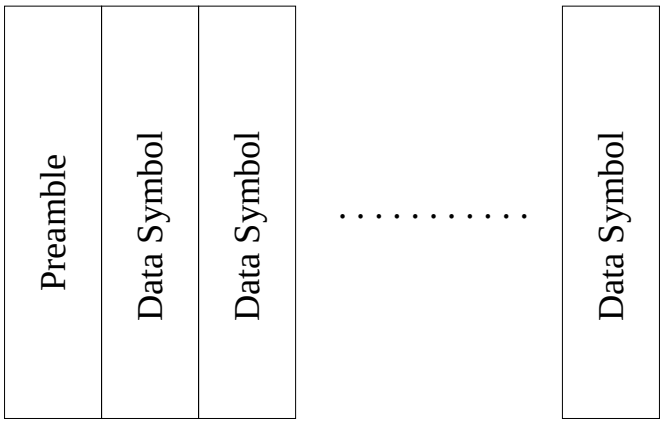

Figure 2.6: Basic frame structure.

The preamble carrier-sets are defined using the following equation:

$$PreambleCarriesSet_n = n + 3 \cdot k, \qquad n = 0,\ 1,\ 2 \text{ and } k = 0,\ \ldots,\ 567 \tag{2.152}$$

where $PreambleCarrierSet_n$ specifies all subcarriers allocated to the specific preamble.

Pilot subcarriers

For pilot subcarriers we use something similar to the sequences used in the preamble. Pilots are BPSK modulated and boosted.

2.6.3 Channel

Simulations, performed in Matlab environment, can be run with two different channel types: SCME and exponential.

The first one is the Spatial Channel Model-Extended. It is an extension to the 3GPP [49] Spatial Channel Model (SCM) and its Matlab implementation was downloaded from `http://www.mathwork.com/products/matlab/`.

The exponential channel is implemented using the Rayleigh model [50] and has an exponential Power Delay Profile (PDP) [28].

2.6.4 Channel estimation

In our simulations we consider the following channel estimators:

- **LS** Least square estimator: this is the simplest one: through Equation (2.135) we compute the channel in the frequency domain on the pilot subcarriers, then through Equation (2.136) we compute the estimated impulse response and, finally, we compute the channel frequency response on all the subcarriers with a simple FFT operation.

- **LMMSE** Linear Minimum Mean Square Error independent on the transmitted data: once we have computed the LS estimation on pilot subcarriers through Equation (2.135), we can compute the LMMSE estimation on pilots independent on the transmitted data through the Equation (2.139) and then, performing as described for the LS, compute the channel on all the subcarriers.

- **LMMSE_X** Linear Minimum Mean Square Error dependent on the transmitted data: as for the LMMSE, once we have computed the LS estimation on pilot subcarriers through Equation (2.135), we can compute the LMMSE estimation on pilots dependent on the transmitted data through the Equation (2.140) and then the channel on all the subcarriers.

- **alpha** This is the channel tracking method described in Section 2.5.3, where the adaptation gain is chosen as defined in Equation (2.149).

2.6.5 Parameters

In this Section we list the parameters used in the simulations:

- OFDM system with 512 subcarriers:
  - ⋆ $46 + 45$ null subcarriers
  - ⋆ 1 DC subcarrier
  - ⋆ 60 pilot subcarriers
  - ⋆ 360 data subcarriers
- 25 OFDM symbols:
  - ⋆ 1 preamble
  - ⋆ 24 data and pilot symbols
- 64 cyclic prefix' length
- 5 MHz bandwidth
- 2 GHz carrier frequency
- QPSK modulation (for data subcarriers)
- SCME channel
- convolutional code

2.7 Results

In our simulations we consider three different speeds for the mobiles: 3, 50 and 130 km/h and we analyze the loss of performance in terms of bit error rate at different speeds due to the channel estimation. Figures 2.7, 2.8 and 2.9 represent the performance for the system described above: red and black lines represent the performance of the system when we assume a perfect channel knowledge with and without coding respectively. Cyan line represents performance of the system if we consider the LS estimator, green and magenta lines represent the performance when the channel estimation is performed through the LMMSE algorithm considering the independence of the estimator from the transmitted data and not. Finally the blue line represents the performance for the adaptive tracking.

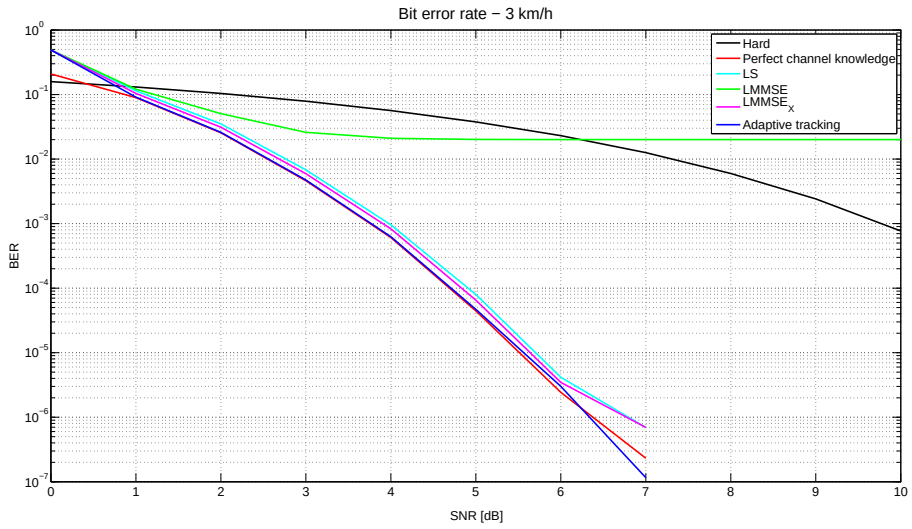

Figure 2.7: Bit error rate with SCME channel and $v = 3$ km/h.

From all the Figures 2.7, 2.8 and 2.9 we observe that the performance of the LMMSE independent on the transmitted data are very bad and for SNR greater than 6.2 dB it performs worst than the uncoded system with perfect channel knowledge. Adaptive tracking is a good algorithm for low speeds and so for channels which vary very slowly in the time. LS and LMMSE dependent on the transmitted pilots have both good performance in terms of bit error rate, because we have a good number of pilots.

The same results are proved in Figures 2.10, 2.11 and 2.12, where we can see the mean square error for the impulse channel estimation: LS error is represented with a cyan line, LMMSE independent on transmitted data with a green line, LMMSE dependent on transmitted data with a magenta line and, finally, the blue line represents the error for the adaptive tracking.

From Figures 2.7 and 2.10 we can observe that for speed equal to 3 km/h the adaptive algorithm is the one which presents the best MSE, than we have the LS, the LMMSE dependent on the transmitted data and the one independent on them

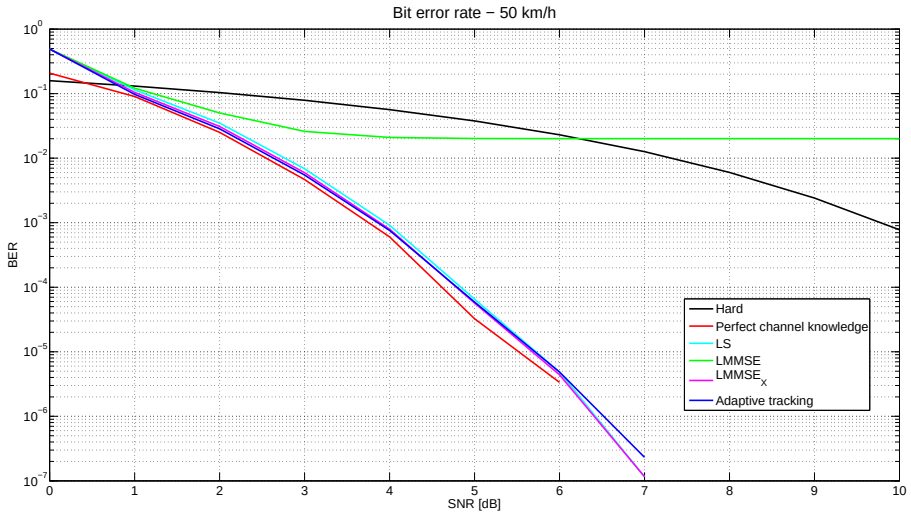

Figure 2.8: Bit error rate with SCME channel and $v = 50$ km/h.

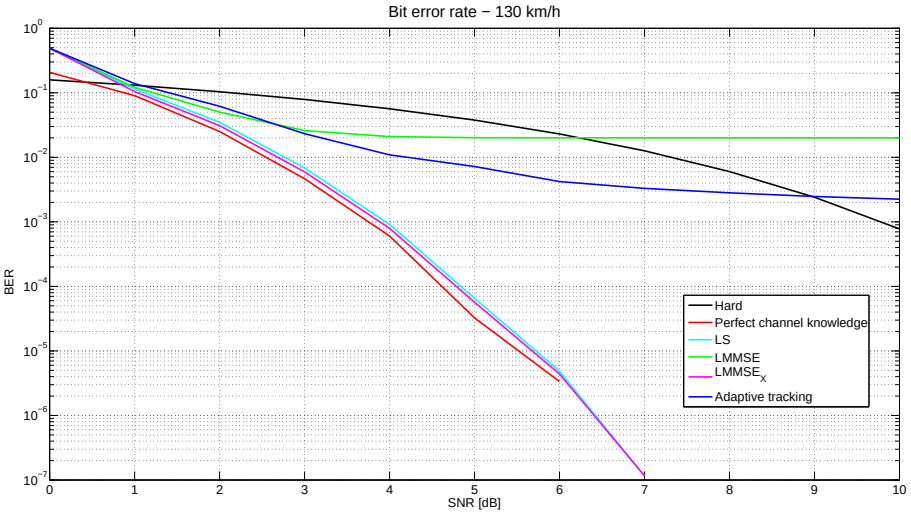

Figure 2.9: Bit error rate with SCME channel and $v = 130$ km/h.

algorithms. However, comparing Figures 2.7 and 2.10 we observe that these MSE values don't present big differences when we represent the BER of the system.

Similar observations hold for speed equal to 50 km/h. Different results are presented for speed equal to 130 km/h. From Figures 2.9 and 2.12 we understand that the adaptive algorithm can't be used, because of the Doppler spread. LS and LMMSE dependent on the transmitted data give both very good performance.

Results similar to the ones proposed in Figures 2.10, 2.11 and 2.12 can be seen in Figures 2.13, 2.14 and 2.15 where the mean square error for every subcarrier is plotted. In these Figures the legend is the same described above.

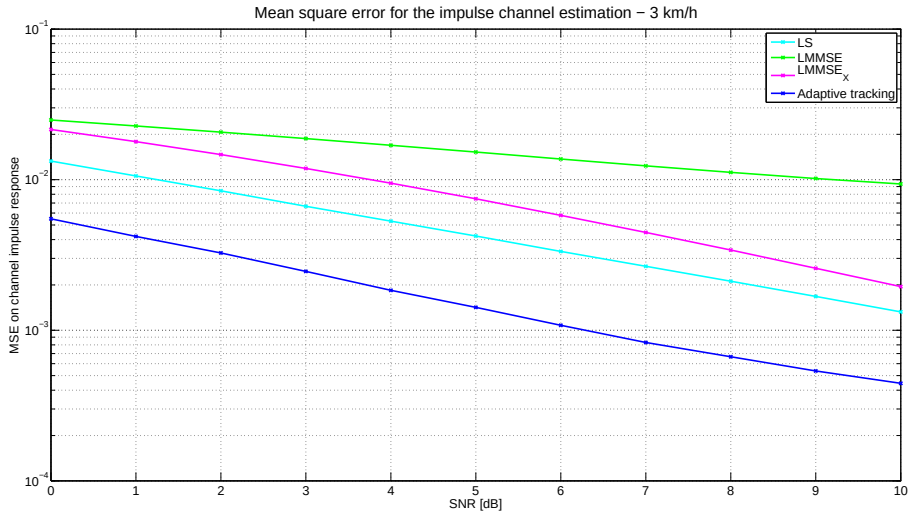

Figure 2.10: Mean square error for the impulse channel estimation: SCME channel and $v = 3$ km/h.

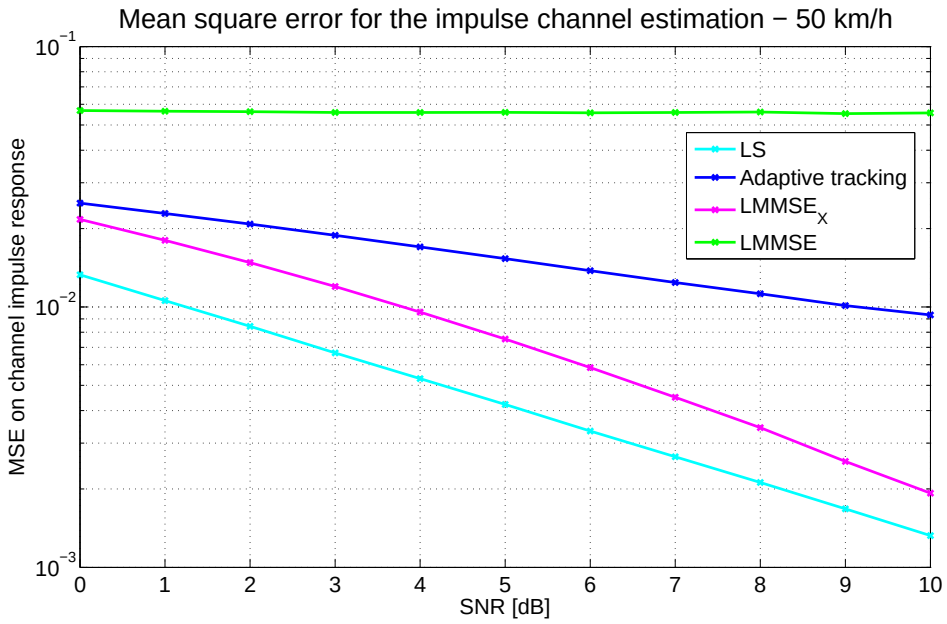

Figure 2.11: Mean square error for the impulse channel estimation: SCME channel and $v = 50$ km/h.

Considering the presented results and the fact the LMMSE is computational more complex than the LS, we conclude that, when we consider a WiMAX system and we need the estimation of the channel, we should use the LS algorithm.

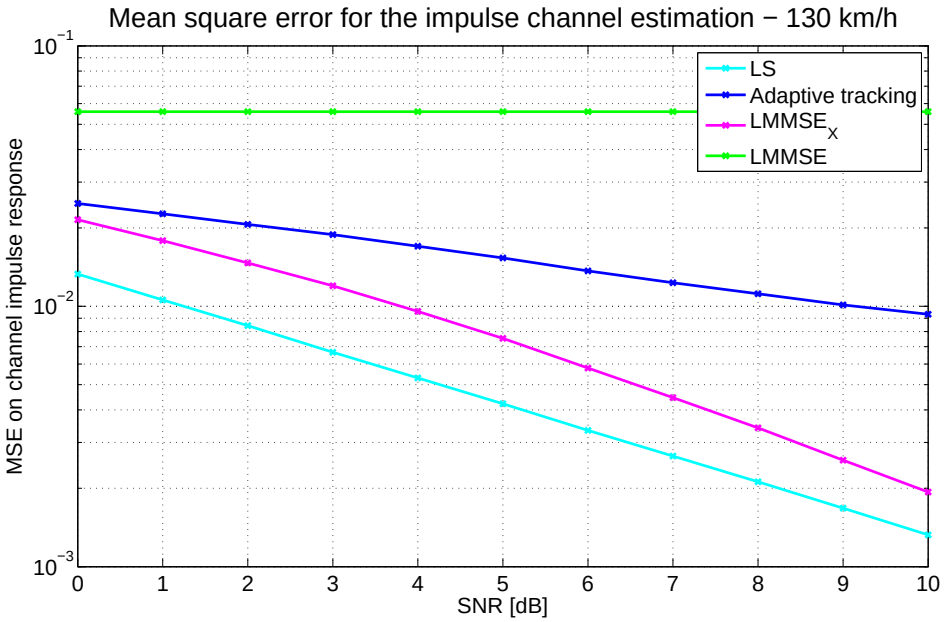

Figure 2.12: Mean square error for the impulse channel estimation: SCME channel and $v = 130$ km/h.

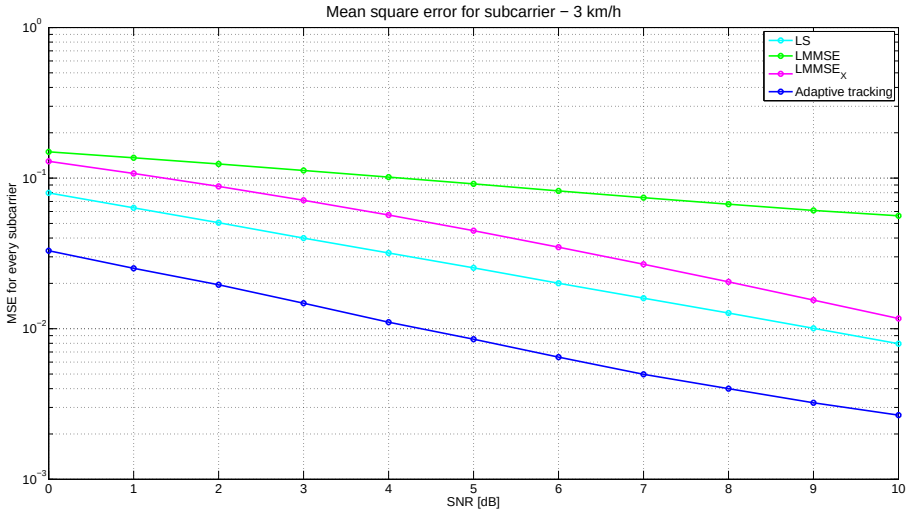

Figure 2.13: Mean square error for every subcarrier: SCME channel and $v = 3$ km/h.

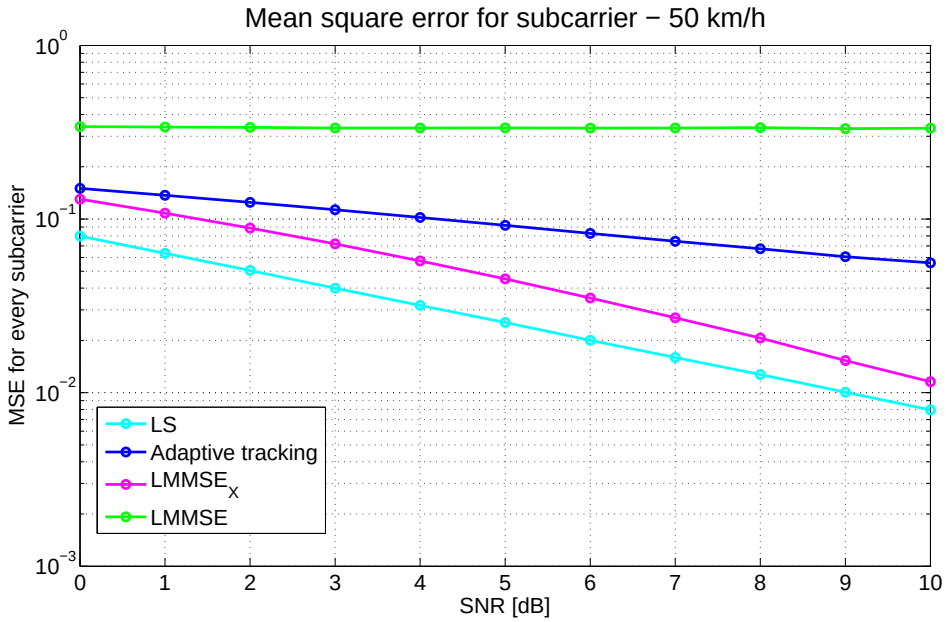

Figure 2.14: Mean square error for every subcarrier: SCME channel and $v = 50$ km/h.

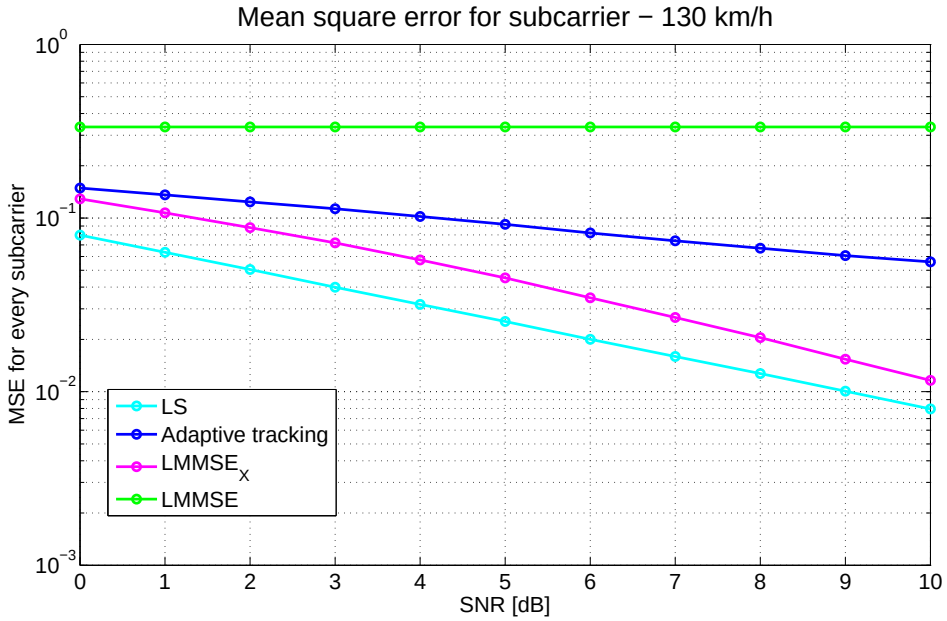

Figure 2.15: Mean square error for every subcarrier: SCME channel and $v = 130$ km/h.

Chapter 3

Expectation maximization algorithm for joint channel estimation and data detection

3.1 Introduction

As said in the Introduction and extensively explained in Chapter 2, in an OFDM system the channel estimation is very important at the receiver, because we need to detect the transmitted data.

In this Chapter we want to investigate the joint detection and channel estimation for the OFDM systems in the frequency domain. Since the detection performance depends on the accuracy of the channel estimate, the joint processing is employed with the Expectation-Maximization (EM) algorithm to improve the accuracy of the channel estimate.

The EM algorithm was discovered and employed independently by several different researchers until Dempster and others [51] brought their ideas together, proved convergence, and coined the term *EM algorithm*. Since that seminal work, hundreds of papers employing the EM algorithm in many areas have been published. A large list of references is found in [52]. A typical application area of the EM algorithm is in genetics, where the observed data (the phenotype) is a function of the underlying, unobserved gene pattern (the genotype), e.g. [53]. Another area is estimating parameters of mixture distributions, e.g. [54]. The EM algorithm has also been widely used in econometric, clinical, and sociological studies that have unknown factors affecting the outcomes [55]. Some applications to the theory of statistical methods are found in [56].

In the area of signal processing applications, the largest area of interest in the EM algorithm is in maximum likelihood tomographic image reconstruction, e.g. [57], [58]. Another commonly cited application is training of hidden Markov models, especially for speech recognition, e.g. [59]. The books [60], [61] have chapters with extensive development of hidden Markov models (HMMs).

Other signal processing and engineering applications began appearing in about 1985.

These include: parameter estimation [62] [63]; ARMA modeling [64], [65]; simultaneous detection and estimation [66], [67], [68]; pattern recognition and neural network training [69], [70], [71]; direction finding [72]; noise suppression [73]; spectroscopy [74]; signal and sequence detection [75]; time-delay estimation [76]; and specialized developments of the EM algorithm itself [77]. The EM algorithm has been the subject for multiprocessing algorithm development [78]. The EM algorithm is also related to algorithms used in information theory to compute channel capacity and rate distortion functions [79], [80], since the expectation step in the EM algorithm produces a result similar to entropy.

Since the known and unknown symbols are simultaneously used to estimate the channel, the following approach is a semi-blind method.

3.2 The Expectation-Maximization (EM) algorithm

3.2.1 Introduction

The EM algorithm produces Maximum Likelihood (ML) estimates of the parameter when there is many-to-one mapping from an underlying distribution to the distribution governing the observation.

The EM algorithm consists of two major steps:

1. **Expectation step (E-step)**: it is performed with respect to the unknown underlying variables using the current estimate of the parameters and conditioned upon the incomplete observation

2. **Maximization step (M-step)**: it provides a new estimation of the parameters that maximize the expectation of the log-likelihood function defined over complete data, conditioned over the most recent observation and the last estimate.

These two steps are iterated until the estimated values converge.

As said in the Introduction to the Chapter, the EM algorithm is related to algorithms used in information theory, to compute channel capacity and rate distortion function, since the E-step produces a result similar to entropy.

The EM algorithm is philosophically similar to ML detection in the presence of unknown phase (incoherent detection) or other unknown parameters: the likelihood function is averaged with respect to the unknown quantity (i.e. the expected value of the likelihood function is computed) before detection, which is a maximization step (see [41]).

3.2.2 General statement of the EM algorithm

Define:

- Y the sample space of the observation

- $\boldsymbol{y} \in \mathbb{R}^m$ an observation from Y

- $\mathcal{X}$ the underlying space of Y

- $\boldsymbol{x} \in \mathbb{R}^n$ the outcome from $\mathcal{X}$, $m < n$; it is referred as complete data and it is not observed directly, but only by means of $\boldsymbol{y}$, where $\boldsymbol{y} = \boldsymbol{y}(x)$, which is a many-to-point mapping.

- $\mathcal{X}(\boldsymbol{y})$ a subset of $\mathcal{X}$ determined through the inverse map of the observed $\boldsymbol{y}$

- $f(\boldsymbol{x}|\theta)$ probability density function (pdf) of the complete data, where $\theta \in \Theta \in \mathbb{R}^r$ is the set of the parameters of the density

- $g(\boldsymbol{y}|\theta) = \int_{\mathcal{X}(\boldsymbol{y})} f(\boldsymbol{x}|\theta)d\boldsymbol{x}$ pdf of the incomplete data

- $\ell_y(\theta) = g(\boldsymbol{y}|\theta)$ likelihood function

- $L_y(\theta) = \log \ell_y(\theta)$ log-likelihood function

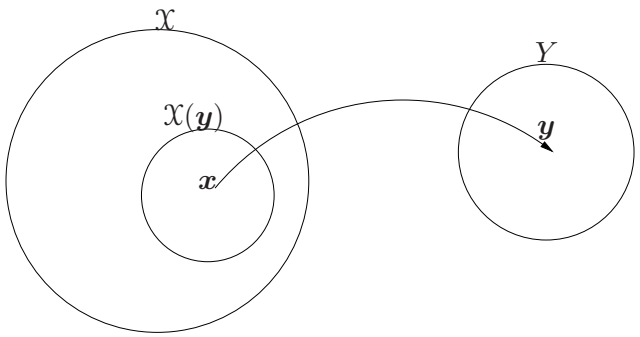

Figure 3.1: Illustration of many to one mapping from $\mathcal{X}$ to Y. The point $\boldsymbol{y}$ is the image of $\boldsymbol{x}$, and the set $\mathcal{X}(\boldsymbol{y})$ is the inverse map of $\boldsymbol{y}$.

The basic idea behind the EM algorithm is that we would like to find θ to maximize $\log f(\boldsymbol{x}|\theta)$, but we do not have the data $\boldsymbol{x}$ to compute the log-likelihood. So, instead we maximize the expectation of $\log f(\boldsymbol{x}|\theta)$ given the data $\boldsymbol{y}$ and our current estimate of θ. This can be expressed in two steps.

Let $\theta^{[k]}$ be our estimate of the parameters at the kth iteration.

For the E-step compute

$$Q(\theta|\theta^{[k]}) = \mathrm{E}[\log f(\boldsymbol{x}|\theta)|\boldsymbol{y}, \theta^{[k]}]; \tag{3.1}$$

and, for the M-step, let $\theta^{[k+1]}$ be that value of θ which maximizes $Q(\theta|\theta^{[k]})$:

$$\theta^{[k+1]} = \arg\max_{\theta} Q(\theta|\theta^{[k]}). \tag{3.2}$$

It is important to note that the maximization is with respect to the first argument of the Q function, the conditioner of the complete data likelihood.

The EM algorithm consists of choosing an initial $\theta^{[k]}$, then performing the E-step and M-step successively until convergence. Convergence may be determined by examining when the parameters quite changing, i.e., stop when $\|\theta^{[k]} - \theta^{[k+1]}\| < \epsilon$ for some ϵ and some appropriate distance measure $\|\cdot\|$.

3.2.3 Convergence of the EM algorithm

For every iterative algorithm, the question of convergence needs to be addressed and for the EM algorithm it may be stated simply: at every iteration of the algorithm, a value of the parameter is computed so that the likelihood function does not decrease. That is, at every iteration the estimated parameter provides an increase in the likelihood function until a local maximum is achieved. In [16], the authors provided a *convergence theorem for the EM algorithm* which is reported in Appendix A.

Despite this convergence theorem, there is no guarantee that the convergence will be to a global maximum. For likelihood function with multiple maxima, convergence will be to a local maximum which depends on the initial starting point $\theta^{[0]}$.

The convergence rate of the EM algorithm is also of interest. Based on mathematical and empirical examinations, it has been determined that the convergence rate is usually slower than the quadratic convergence typically available with a Newton's-type method [54].

However, as observed by Dempster [51], the convergence near the maximum depends upon the eigenvalues of the Hessian of the update function M, so that rapid convergence may be possible.

In any event, even with potentially slow convergence there are advantages to EM algorithms over Newton's algorithms.

In the first place, no Hessian needs to be computed. Also there is no chance of "overshooting" the target or diverging away from the maximum.

The EM algorithm is guaranteed to be stable and to converge to an ML estimate. Further discussion of convergence appears in [81], [82].

3.3 Joint channel estimation and data detection for OFDM systems

3.3.1 System model

Let N be the number of subcarriers and $\boldsymbol{x}(n) = [x_0(n), x_1(n), \ldots, x_{N-1}(n)]^T$ the nth signal vector of size $N \times 1$ to be transmitted, where $x_m(n) \in \mathcal{S}$ and $\mathcal{S}$ is the set of modulated symbols.

After the OFDM modulator the signal is

$$\tilde{\boldsymbol{x}}(n) = \boldsymbol{F}^H \boldsymbol{x}(n) \tag{3.3}$$

where $\boldsymbol{F}$ is the Fourier transform matrix: $F_{n,m} = \frac{1}{\sqrt{N}} e^{-j2\pi(n-1)(m-1)/N}$, $n, m = 1, 2, \ldots$ $\ldots, N$.

We assume that the channel impulse response (CIR) is time-invariant for an OFDM symbol and for an nth signal vector it is an $L \times 1$ vector: $\{h_l\}_{l=0}^{L-1}$. We assume also that the CIR's length L is smaller than the number of subcarriers N ($L << N$) and also than the cyclic prefix' length CP ($L << CP$).

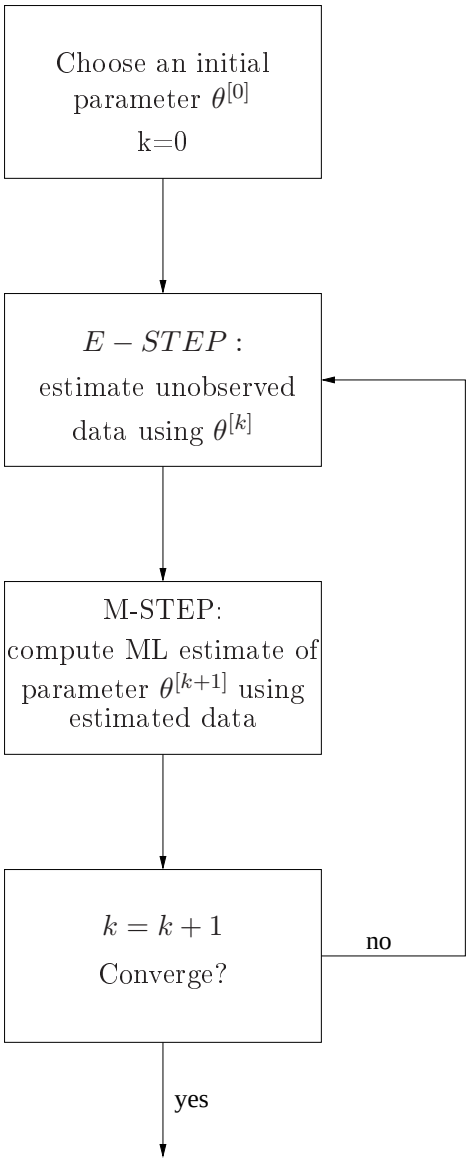

Figure 3.2: An overview of the EM algorithm. After initialization, the E-step and M-step are iteratively computed until the parameter estimate has come to convergence.

The discrete Fourier transform (DFT) of the CIR is:

$$\tilde{h}_p(n) = \sum_{l=0}^{L-1} h_l(n) e^{-j2\pi lp/N}, \qquad p = 0, 1, \ldots, N-1 \tag{3.4}$$

and we define $\tilde{\boldsymbol{H}}(n) = \text{diag}[\tilde{h}_0(n), \tilde{h}_1(n), \ldots, \tilde{h}_{N-1}(n)]$.

Hence, the received signal after removing cyclic prefix is:

$$\tilde{\boldsymbol{y}}(n) = \mathscr{H}(n)\tilde{\boldsymbol{x}}(n) + \tilde{\boldsymbol{w}}(n), \tag{3.5}$$

where $\mathscr{H}(n)$ is the $N \times N$ cyclic matrix of $[h_0(n), h_1(n), \ldots, h_{L-1}(n), \underbrace{0, \ldots, 0}_{N-L}]^T$ and

$\tilde{\boldsymbol{w}}(n)$ is the background noise in the time domain.

After the DFT the received signal is:

$$\begin{aligned}\boldsymbol{y}(n) &= \boldsymbol{F}\tilde{\boldsymbol{y}}(n) \\ &= \boldsymbol{F}(\mathscr{H}(n)\tilde{\boldsymbol{x}}(n) + \tilde{\boldsymbol{w}}(n)) \\ &= \underbrace{\boldsymbol{F}\mathscr{H}(n)\boldsymbol{F}^H}_{\tilde{\boldsymbol{H}}(n)}\boldsymbol{x}(n) + \boldsymbol{F}\tilde{\boldsymbol{w}}(n) \\ &= \text{diag}[x_0(n),\, x_1(n),\, \ldots,\, x_{N-1}(n)]\tilde{\boldsymbol{h}}(n) + \boldsymbol{w}(n) \end{aligned} \tag{3.6}$$

where $\boldsymbol{w}(n)$ is the noise in the frequency domain and $\tilde{\boldsymbol{h}}(n) = [\tilde{h}_0(n),\, \tilde{h}_1(n),\, \ldots,\, \tilde{h}_{N-1}(n)]^T$ is the frequency channel response computed in Equation (3.4).

We can also define the $N \times N$ covariance matrix of noise:

$$\text{E}[\boldsymbol{w}(n)\boldsymbol{w}^H(n)] = \sigma_w^2 I_{N\times N}.$$

Now, we consider a comb-type OFDM system, in which, in each OFDM symbol, some subcarriers are used to transmit pilots, as we can see in Figure 3.3.

Figure 3.3: Comb-type structure for an OFDM system.

Let $\boldsymbol{x}_p(n)$ be the signal block which contains pilot tones. In addition, we denote by $\boldsymbol{x}_d(n)$ the signal block which contains data tones. We have:

$$\boldsymbol{x}(n) = \boldsymbol{x}_p(n) + \boldsymbol{x}_d(n). \tag{3.7}$$

Let $\mathcal{P}$ denote the index set for pilot tones. Then, the index set for data tones, $\mathcal{D}$, becomes the complementary set of $\mathcal{P}$. From Equation (3.7) we understand that $\boldsymbol{x}_p(n)$ and $\boldsymbol{x}_d(n)$ must have the same dimension, so we need to define the following rule: every element in $\boldsymbol{x}_p(n)$ whose index is in $\mathcal{P}$ has some values from $\mathcal{S}$. Otherwise, the value becomes zero. The same rule is applied to $\boldsymbol{x}_d(n)$, considering the elements whose indexes are in $\mathcal{D}$. Attention must be paid when, as usually happens, pilots and data are modulated with different modulations. In this case values are picked from two different sets; we define them $\mathcal{S}_p$ and $\mathcal{S}_d$.

From (3.6), the received signal vector is rewritten as

$$\boldsymbol{y}(n) = \tilde{\boldsymbol{H}}(n)\boldsymbol{x}_p(n) + \tilde{\boldsymbol{H}}(n)\boldsymbol{x}_d(n) + \boldsymbol{w}(n). \tag{3.8}$$

Note that $\boldsymbol{x}_p(n)$ is known at the receiver and so the channel matrix $\tilde{\boldsymbol{H}}(n)$ and the data vector $\boldsymbol{x}_d(n)$ shall be estimated.

For notational convenience, we omit the symbol index n. Let $\boldsymbol{h} = [h_0, h_1, \ldots, h_{L-1}]$ be the channel vector to be estimated jointly with the data symbols. In addition, we define the $N \times L$ DFT matrix $\boldsymbol{F}_L$ as:

$$[F_L]_{n,m} = e^{-j2\pi(n-1)(m-1)/N}, \qquad n = 1, 2, \ldots, N, \quad m = 1, 2, \ldots, L. \tag{3.9}$$

Then, from (3.4), we can show that

$$\tilde{\boldsymbol{h}} = \boldsymbol{F}_L \boldsymbol{h}.$$

For notation convenience, we define:

- $\boldsymbol{A}$ =diag$(\boldsymbol{x}_p)\boldsymbol{F}_L$
- $\boldsymbol{B}(\boldsymbol{h})$ =diag$(\boldsymbol{F}_L\boldsymbol{h})$
- $\boldsymbol{K} = \boldsymbol{J}_d$, where $\boldsymbol{J}_d$ is an $N \times N_d$ matrix whose elements are given by

$$[J_d]_{m,n} = \begin{cases} 1 & m = \beta_n, \ n = 1, 2, \ldots, N_d \\ 0 & \text{otherwise} \end{cases}$$

  and β_n is the nth smallest element of the index set of data tones $\mathcal{D}$

- $\boldsymbol{d} = \boldsymbol{J}_d^T \boldsymbol{x}_d$

and so, rewrite the received signal as [17]:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}\boldsymbol{d} + \boldsymbol{w}. \tag{3.10}$$

It is noteworthy that $\boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}$ is decoupled, this means that each column vector has only one non zero elements, while the other element are zero.

3.3.2 Channel estimation

To develop the algorithm of interest, we consider an OFDM symbol and we suppose that the channel vector $\boldsymbol{h}$ is a symbol-deterministic channel vector[1] to be estimated through the paper.

Cramer-Rao Lower Bound (CRLB)

The Cramer-Rao lower bound (CRLB) is an important criterion to evaluate the reliability of any unbiased estimator, because it provides the MMSE bound among all unbiased estimators.

We assume that the channel vector $\boldsymbol{h}$ is a deterministic channel vector.

[1] By the expression symbol-deterministic channel vector we mean a channel as described in Section 2.1 which is assumed constant in an OFDM symbol and varies very slowly from symbol to symbol.

The CRLB can be found if the vector of data symbols, $\boldsymbol{d}$, is known. Since $\boldsymbol{B}(\boldsymbol{h}) = \text{diag}(\boldsymbol{F}_L\boldsymbol{h})$, we can show that:

$$\boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}\boldsymbol{d} = \text{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L\boldsymbol{h}. \tag{3.11}$$

From this, we have a linear model for the received signal vector $\boldsymbol{y}$:

$$\begin{aligned} \boldsymbol{y} &= \text{diag}(x_0, x_1, \ldots, x_{N-1})\boldsymbol{F}_L\boldsymbol{h} + \boldsymbol{w} \\ &= \boldsymbol{Q}\boldsymbol{h} + \boldsymbol{w}, \end{aligned} \tag{3.12}$$

where

$$\begin{aligned} \boldsymbol{Q} &= \text{diag}(x_0, x_1, \ldots, x_{N-1})\boldsymbol{F}_L \\ &= \boldsymbol{A} + \text{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L. \end{aligned}$$

If the vector of data symbols, $\boldsymbol{d}$, is known, the matrix $\boldsymbol{Q}$ becomes a known matrix. Hence, from [45], we can have the CRLB as:

$$\begin{aligned} \text{E}\left[\begin{bmatrix} \Re(\hat{\boldsymbol{h}} - \boldsymbol{h}) \\ \Im(\hat{\boldsymbol{h}} - \boldsymbol{h}) \end{bmatrix} [(\Re(\hat{\boldsymbol{h}} - \boldsymbol{h}))^T (\Im(\hat{\boldsymbol{h}} - \boldsymbol{h})^T)]\right] &\geq CRLB(\boldsymbol{h}) \\ = \frac{\sigma_w^2}{2}\begin{bmatrix} \Re\{(\boldsymbol{Q}^H\boldsymbol{Q})^{-1}\} & -\Im\{(\boldsymbol{Q}^H\boldsymbol{Q})^{-1}\} \\ \Im\{(\boldsymbol{Q}^H\boldsymbol{Q})^{-1}\} & \Re\{(\boldsymbol{Q}^H\boldsymbol{Q})^{-1}\} \end{bmatrix}, \end{aligned} \tag{3.13}$$

where $\hat{\boldsymbol{h}}$ is an unbiased estimate of $\boldsymbol{h}$. It follows that:

$$\text{E}[\|\hat{\boldsymbol{h}} - \boldsymbol{h}\|^2] \geq \text{trace}(CRLB(h)). \tag{3.14}$$

Note that this bound is achievable if $\boldsymbol{d}$ is known or correctly detected. If N is sufficiently large and pilot and data symbols are random, $\boldsymbol{Q}^H\boldsymbol{Q}$ can be approximate by $N\sigma_x^2\boldsymbol{I}$, where $\sigma_x^2 = \text{E}[|x_m(n)|^2]$. Hence we can have:

$$\text{trace}(CRLB(\boldsymbol{h})) = \frac{L}{N}\frac{\sigma_w^2}{\sigma_x^2}. \tag{3.15}$$

Note that this bound is independent of the data symbols.

Maximum Likelihood (ML) estimate

In OFDM, according to [41], a simple estimate of $\boldsymbol{h}$ can be found only utilizing pilot tones. It can be shown that the two subspaces, Range($\boldsymbol{A}$) and Range($\boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}$), are orthogonal for any $\boldsymbol{h}$. If $\boldsymbol{A}$ is full-rank and $N_p \geq L$, the ML estimate is given by:

$$\hat{\boldsymbol{h}}_{OF,p} = (\boldsymbol{A}^H\boldsymbol{A})^{-1}\boldsymbol{A}^H\boldsymbol{y} \tag{3.16}$$

and it is not affected by the data tones. It can be shown that the uniform distribution of pilot subcarriers provides the best performance by minimizing the mean square error

(MSE).

As we will see later, this estimate can be the initial channel estimation for the EM algorithm.

Frequency domain equalization

Suppose that the channel estimate $\hat{\boldsymbol{h}}$ is given, we need to equalize the received signal. To do this we can adopt more than one technique. One of these is the zero-forcing equalization in the frequency domain, which can be written as:

$$\hat{\boldsymbol{d}}_{zf} = \boldsymbol{P}_{zf}(\hat{\boldsymbol{h}})(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{h}}), \tag{3.17}$$

where $\boldsymbol{P}_{zf}(\hat{\boldsymbol{h}}) = ((\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})^H(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K}))^{-1}(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})^H$ is an $N_d \times N$ matrix, $\boldsymbol{B}(\hat{\boldsymbol{h}}) = \text{diag}(\boldsymbol{F}_L\hat{\boldsymbol{h}})$ and $\hat{\boldsymbol{h}}$ is the best estimate we known at the moment (i.e. $\hat{\boldsymbol{h}}_{OF,p}$ or $\hat{\boldsymbol{h}}_{(k)}$).

In OFDM this method can lead to infinite tones' estimates if there exist spectral nulls. To overcome this, the *minimum mean square error* (MMSE) equalizer can be used as:

$$\hat{\boldsymbol{d}}_{mmse} = \boldsymbol{P}_{mmse}(\hat{\boldsymbol{h}})(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{h}}), \tag{3.18}$$

where $\boldsymbol{P}_{mmse}(\hat{\boldsymbol{h}}) = ((\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})^H(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K}) + \sigma_w^2 I)^{-1}(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})^H$.
Note that $(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})^H(\boldsymbol{B}(\hat{\boldsymbol{h}})\boldsymbol{K})$ is a diagonal matrix and its elements are non-negative.

3.3.3 Joint channel estimation and data detection using EM algorithm

From (3.10) and the probability density function (pdf) of $\boldsymbol{y}$ conditioned on $\boldsymbol{d}$, $\boldsymbol{h}$ and σ_w^2: $f(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{h}, \sigma_w^2)$, the joint ML estimation of the channel vector and data symbol vector can be formulated as:

$$\begin{aligned} \{\hat{\boldsymbol{h}}, \hat{\boldsymbol{d}}, \hat{\sigma}_w^2\} &= \arg \max_{\boldsymbol{h}\in\mathcal{C}^L, \boldsymbol{d}\in\mathcal{S}^{N_d}, \sigma_w^2} f(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{h}, \sigma_w^2) \\ &= \arg \max_{\boldsymbol{h}\in\mathcal{C}^L, \boldsymbol{d}\in\mathcal{S}^{N_d}, \sigma_w^2} \frac{1}{(\pi\sigma_w^2)^N} \exp\Big(-\frac{\psi(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{d})}{\sigma_w^2}\Big) \end{aligned} \tag{3.19}$$

where $\psi(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{d}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{h} - \boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}\boldsymbol{d}\|^2$, $\mathcal{C}^L$ is the L-dimensional complex vector space and $\mathcal{S}^{N_d}$ is the N_d dimensional Cartesian product of $\mathcal{S}$. But the complexity of (3.19) is too high, so, to solve it, we use the EM algorithm.

We define:

- the received signal $\boldsymbol{y}$ as the incomplete data,
- the couple of the received signal $\boldsymbol{y}$ and the transmitted data $\boldsymbol{d}$, $\{\boldsymbol{y}, \boldsymbol{d}\}$, as the complete data,
- the noise variance σ_w^2 and the CIR $\boldsymbol{h}$ as the unknown parameters to be estimated.

Note that $\boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}$ is decoupled and so the joint detection of the data vector $\boldsymbol{d}$ is replaced by the individual detection of the data symbols and this significantly reduces the complexity of the detection algorithm.

E-step

We start considering the kth iteration and we have the estimates $\hat{\boldsymbol{h}}_{(k)}$ and $\hat{\sigma}^2_{w,(k)}$. Let $f(\boldsymbol{y},\boldsymbol{d}|\boldsymbol{h},\sigma_w^2)$ be the joint pdf of the complete data $\{\boldsymbol{y},\boldsymbol{d}\}$ conditioned on $\boldsymbol{h}$ and σ_w^2. For the E-step we need to find $\mathrm{E}[\log f(\boldsymbol{y},\boldsymbol{d}|\boldsymbol{h},\sigma_w^2)|\hat{\boldsymbol{h}}_{(k)},\hat{\sigma}^2_{w,(k)},\boldsymbol{y}]$. For convenience, let be $\mathcal{H}_{(k)}=\{\hat{\boldsymbol{h}}_{(k)},\hat{\sigma}^2_{w,(k)},\boldsymbol{y}\}$. From (3.10), we have:

$$\begin{aligned}\mathrm{E}[\log f(\boldsymbol{y},\boldsymbol{d}|\boldsymbol{h},\sigma_w^2)|\mathcal{H}_{(k)}] &= \mathrm{E}\Big[\log\Big\{\frac{1}{(\pi\sigma_w^2)^N}\exp\Big(-\frac{\psi(\boldsymbol{y},\boldsymbol{h},\boldsymbol{d})}{\sigma_w^2}\Big)\Big\}|\mathcal{H}_{(k)}\Big]\\ &= -N\log(\pi\sigma_w^2)-\frac{\mathrm{E}[\psi(\boldsymbol{y},\boldsymbol{h},\boldsymbol{d})|\mathcal{H}_{(k)}]}{\sigma_w^2} \qquad (3.20)\end{aligned}$$

Note that the expectation is carried out over $\boldsymbol{d}$ since $\boldsymbol{y}$ is given.

From $\boldsymbol{B}(\boldsymbol{h})\boldsymbol{K}\boldsymbol{d}=\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L\boldsymbol{h}$, we have:

$$\begin{aligned}\mathrm{E}[\psi(\boldsymbol{y},\boldsymbol{h},\boldsymbol{d})|\mathcal{H}_{(k)}] &= \mathrm{E}[\|\boldsymbol{y}-(\boldsymbol{A}+\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L)\boldsymbol{h}\|^2|(H_{(k)})]\\ &\triangleq v_{(k)}(\boldsymbol{h},\boldsymbol{y}). \qquad (3.21)\end{aligned}$$

It follows that:

$$v_k(\boldsymbol{h},\boldsymbol{y})=\boldsymbol{y}^H\boldsymbol{y}-2\Re\{\boldsymbol{y}^H\boldsymbol{G}_1\boldsymbol{h}\}+\boldsymbol{h}^H\boldsymbol{G}_2\boldsymbol{h} \qquad (3.22)$$

where

$$\boldsymbol{G}_1=\boldsymbol{A}+\mathrm{E}[\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L|\mathcal{H}_{(k)}]$$

and

$$\begin{aligned}\boldsymbol{G}_2=\boldsymbol{A}^H\boldsymbol{A}+\boldsymbol{A}^H\mathrm{E}[\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L|\mathcal{H}_{(k)}]+(\mathrm{E}[diag(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L|\mathcal{H}_{(k)}])^H\boldsymbol{A}\\ +\mathrm{E}[(\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L)^H(\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L)|\mathcal{H}_{(k)}]\end{aligned}$$

For convenience, we define $\boldsymbol{G}_0=\mathrm{E}[\mathrm{diag}(\boldsymbol{K}\boldsymbol{d})\boldsymbol{F}_L|\mathcal{H}_{(k)}]$. Hence, it can be easy to show that:

$$\boldsymbol{G}_0=\mathrm{diag}(\boldsymbol{K}\mathrm{E}[\boldsymbol{d}|\mathcal{H}_{(k)}])\boldsymbol{F}_L, \qquad (3.23)$$

$$\boldsymbol{G}_1=\boldsymbol{A}+\boldsymbol{G}_0 \qquad (3.24)$$

and

$$\begin{aligned}\boldsymbol{G}_2 &= \boldsymbol{A}^H\boldsymbol{A}+\boldsymbol{A}^H\boldsymbol{G}_0+\boldsymbol{G}_0^H\boldsymbol{A}+\boldsymbol{F}_L^H(\mathrm{diag}(\boldsymbol{K}\mathrm{E}[\boldsymbol{d}\boldsymbol{d}^H|\mathcal{H}_{(k)}])\boldsymbol{K}^H)\boldsymbol{F}_L\\ &= \boldsymbol{G}_1^H\boldsymbol{G}_1 \qquad (3.25)\end{aligned}$$

where $\mathrm{diag}(\boldsymbol{X})$ denotes a diagonal matrix which takes all diagonal elements from $\boldsymbol{X}$.

Hence, in order to find $v_k(\boldsymbol{h},\boldsymbol{y})$, we need to compute the mean vector and covariance matrix of $\boldsymbol{d}$ conditioned on $\mathcal{H}_{(k)}$. We assume that the symbols in $\boldsymbol{d}$ are independent.

The conditional mean of $\boldsymbol{d}$ can be obtained as follows:

$$[\mathrm{E}[\boldsymbol{d}|\mathcal{H}_{(k)}]]_m = \sum_{l=1}^{P} q_l Pr(d_m = q_l|\mathcal{H}_{(k)}), \qquad m = 1, 2, \ldots, N_d \tag{3.26}$$

where d_m stands for the mth element of $\boldsymbol{d}$ and q_l is the lth element of $\mathbb{S}$. We also need the conditioned covariance matrix $\mathrm{E}[\boldsymbol{d}\boldsymbol{d}^H|\mathcal{H}_{(k)}]$, so we have:

$$\mathrm{E}[d_m d_r^*|\mathcal{H}_{(k)}] = \sum_{l=1}^{P}\sum_{l'=1}^{P} q_l q_{l'}^* Pr(d_m = q_l|\mathcal{H}_{(k)}) Pr(d_r = q_l'|\mathcal{H}_{(k)}), \qquad m \neq r \tag{3.27}$$

and

$$\mathrm{E}[|d_m|^2|\mathcal{H}(k)] = \sum_{l=1}^{P} |q_l|^2 Pr(d_m = q_l|\mathcal{H}_{(k)}), \qquad m = 1, 2, \ldots, N_d \tag{3.28}$$

From above, we need to find the conditional probability $Pr(d_m = q_l|\mathcal{H}_{(k)})$. Let $\bar{\boldsymbol{d}}_m$ be a sub-vector obtained from $\boldsymbol{d}$ by removing the mth element, d_m ($\bar{\boldsymbol{d}}_m = \langle \boldsymbol{d} \rangle_m$ denotes this operation). Define $\boldsymbol{y}_{(k)} = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{h}}_{(k)}$ and $\boldsymbol{U}_{(k)} = \boldsymbol{B}(\hat{\boldsymbol{h}}_{(k)})\boldsymbol{K}$, where, as usual, $\boldsymbol{B}(\hat{\boldsymbol{h}}_{(k)}) = \mathrm{diag}(\boldsymbol{F}_L \hat{\boldsymbol{h}}_{(k)})$. Furthermore, denote by $\bar{\boldsymbol{U}}_{m,(k)}$ the sub-matrix of $\boldsymbol{U}_{(k)}$ obtained by removing the mth column vector. Define also $\Lambda(d_m = q_l) = \|\boldsymbol{y}_{(k)} - \bar{\boldsymbol{U}}_{m,(k)}\bar{\boldsymbol{d}}_{m,(k)} - \boldsymbol{u}_{m,(k)} q_l\|^2$, where $\boldsymbol{u}_{m,(k)}$ is the mth column of the matrix $\boldsymbol{U}_{(k)}$.

In [17] it is shown that we can find the following expression for $Pr(d_m = q_l|\mathcal{H}_{(k)})$, which results not affected by $\hat{\boldsymbol{d}}_m$:

$$\begin{aligned} Pr(d_m = q_l|\mathcal{H}_{(k)}) &= \frac{\exp\left(-\frac{1}{\sigma^2_{w,(k)}}\Lambda(d_m = q_l)\right)}{\sum_{l'=1}^{P}\exp\left(-\frac{1}{\sigma^2_{w,(k)}}\Lambda(d_m = q_l')\right)} \\ &\triangleq \hat{p}_{(k)}(d_m = q_l), \qquad m = 1, 2, \ldots, N_d \end{aligned} \tag{3.29}$$

Once $\hat{p}_{(k)}(d_m = q_l)$ is known, we can find $\mathrm{E}[\boldsymbol{d}|\mathcal{H}_{(k)}]$, $\mathrm{E}[\boldsymbol{d}\boldsymbol{d}^H|\mathcal{H}_{(k)}]$, $\boldsymbol{G}_0$, $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ and $v_{(k)}(\boldsymbol{h}, \boldsymbol{y})$.

In above, we have found an appropriate expression of $\mathrm{E}[\log f(\boldsymbol{y}, \boldsymbol{d}|\boldsymbol{h}, \sigma_w^2)|\mathcal{H}_{(k)}]$ and with this, the M-step can be carried out to update the estimates of the channel vector and the noise variance.

M-step

The next channel estimate, $\hat{\boldsymbol{h}}_{(k+1)}$, can be found by minimizing $v_{(k)}(\boldsymbol{h}, \boldsymbol{y})$, the result is:

$$\hat{\boldsymbol{h}}_{(k+1)} = \boldsymbol{G}_2^{-1}\boldsymbol{G}_1^H \boldsymbol{y}. \tag{3.30}$$

It can be easily shown that the estimate of the noise variance which maximize (3.20) is given by:

$$\hat{\sigma}^2_{w,(k+1)} = \frac{1}{N} v_k(\hat{\boldsymbol{h}}_{(k+1)}, \boldsymbol{y}). \tag{3.31}$$

Initialization

The estimate of the channel vector based only on pilots tones, written in equation (3.16) can be used for the initial estimate of the channel vector, $\hat{\boldsymbol{h}}_{(0)} = \hat{\boldsymbol{h}}_{OF,p}$.

The initial variance can be given by:

$$\sigma^2_{w,(0)} = \frac{\|\boldsymbol{y}\|^2}{N}. \tag{3.32}$$

In addition, the initial vector for $\bar{\boldsymbol{d}}_m$ can be given by the hard decision from the MMSE frequency-domain equalizer in (3.18) with the initial channel estimate.

The iteration is terminated when $\sigma^2_w(k+1) \geq \sigma^2_w(k)$.

Figure 3.4 gives a schematic description of the EM algorithm.

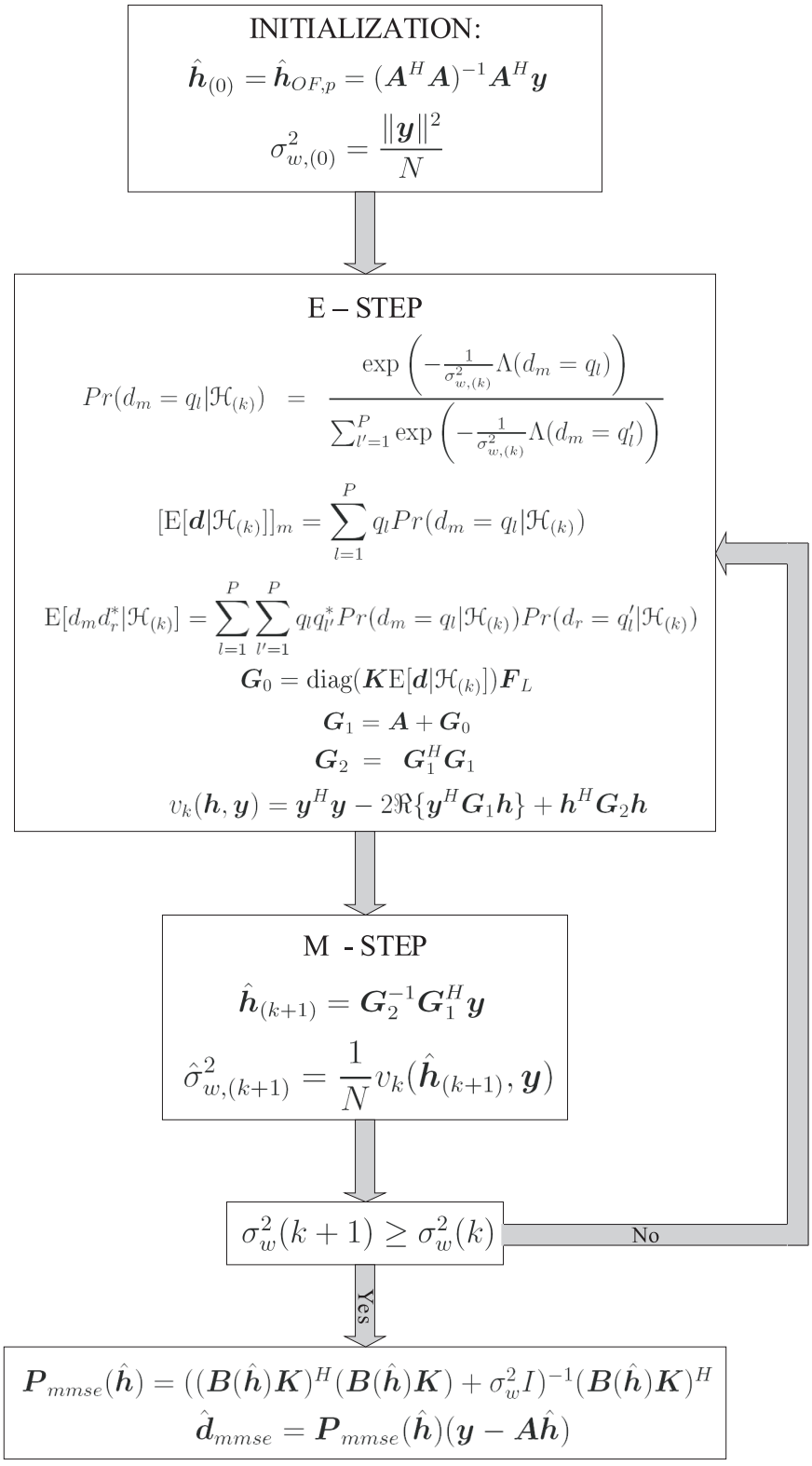

Figure 3.4: Brief resume of the evolution of the EM algorithm.

3.4 System description and simulation results

The results presented in this Section are obtained with the setup described in Section 2.6. There's only a difference: in this case we don't consider the possibility of concatenating more bits before coding. Here we encode bits symbol-by-symbol, so the sourceword length is 360.

3.4.1 Results

As in Chapter 2, also for the EM algorithm we present the performance in terms of bit error rate. We consider two different types of channel: AWGN (Figure 3.5) and SCME channel (Figures 3.6, 3.7 and 3.8), and for the last one we analyze the results for three different speeds of the mobiles: 3, 50 and 130 km/h. To have some terms of comparison, the performance of the system are represented when we assume perfect channel knowledge and the data bits are coded and not. For the same reason, we also represent the BER when the channel estimation is performed through the ML estimation.

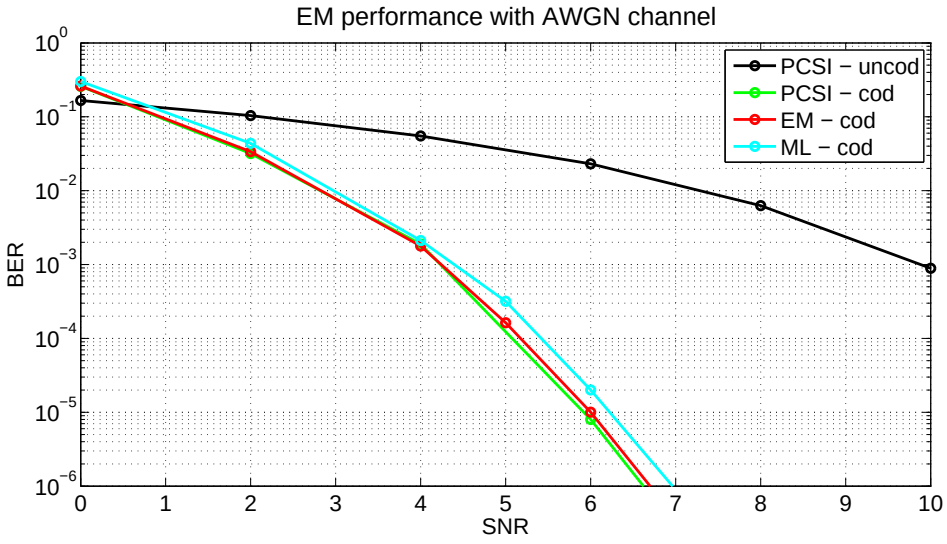

Figure 3.5: Bit error rate with AWGN channel.

Figures 3.5, 3.6, 3.7 and 3.8 represent the results obtained through numerical simulations: the black and green lines represent the bit error rate when we assume perfect channel knowledge and we consider an uncoded and a coded system respectively. The cyan line represents performance for the ML algorithm and the red one the performance for the EM algorithm.

From all these Figures, we can see that the EM algorithm always gives better performance than the ML one, but it is very complex from a computational point of view.

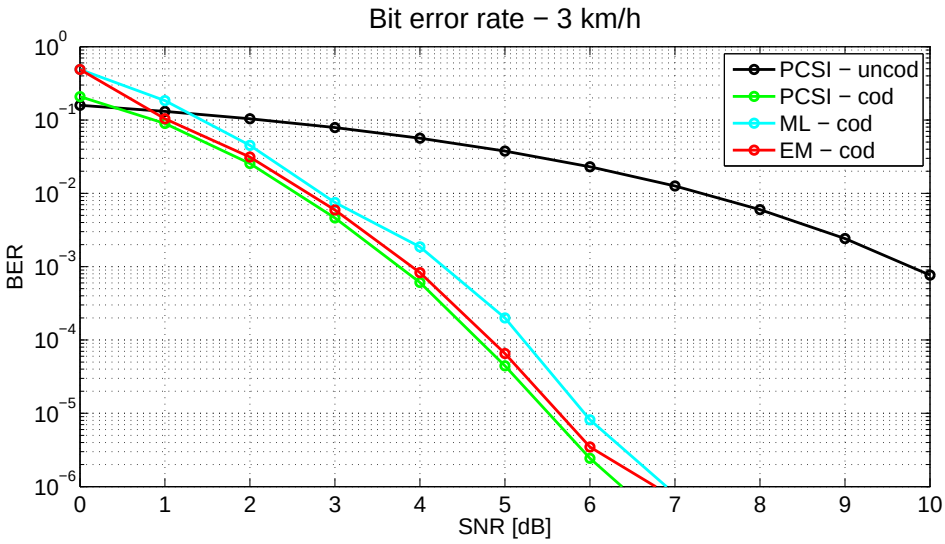

Figure 3.6: Bit error rate with SCME channel and $v = 3$ km/h.

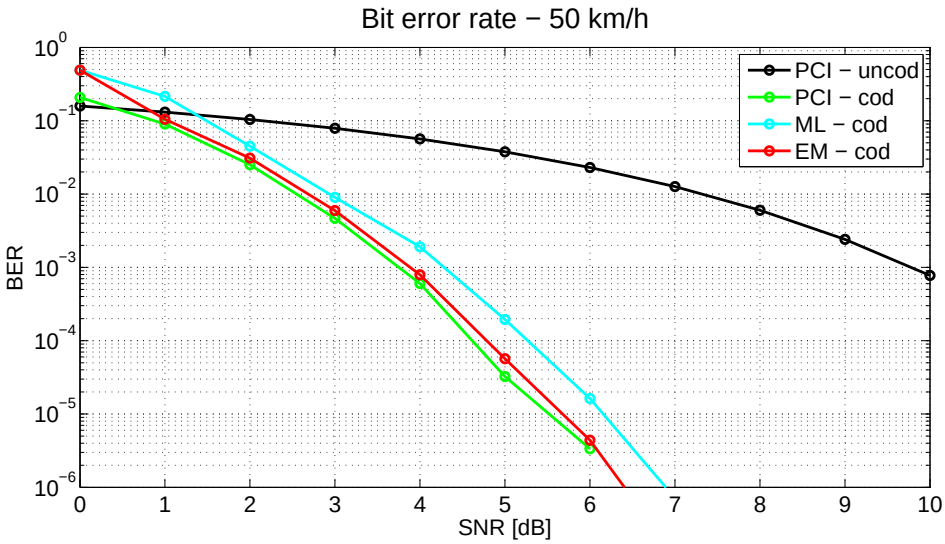

Figure 3.7: Bit error rate with SCME channel and $v = 50$ km/h.

3.5 Conclusions

Some considerations can be made to conclude this part on the channel estimation in wireless systems. First of all we observe that all the estimator presented in these two Chapters require the knowledge of the channel impulse response length. We can try to estimate it, but performance experiences a loss in term of bit error rate (see Appendix B for more details).

Another important parameter is the noise variance. This must be known when we perform a MMSE or an EM estimation: while in EM algorithm it is the variable used to decide when stopping iterating and so it is estimated, in the MMSE we assume it

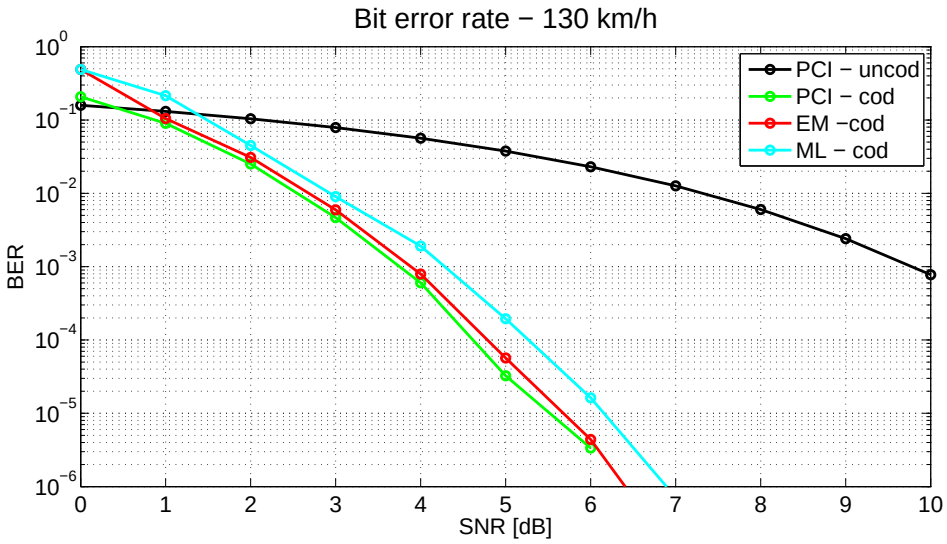

Figure 3.8: Bit error rate with SCME channel and $v = 130$ km/h.

known.

Forgiven for a moment these drawbacks, and remembering that in the conclusion of the Chapter 2 we stated that for a WiMAX system we can use the LS algorithm when the channel estimation in needed, here in Figures 3.9, 3.10, 3.11 and 3.12, we compare LS and EM performance.

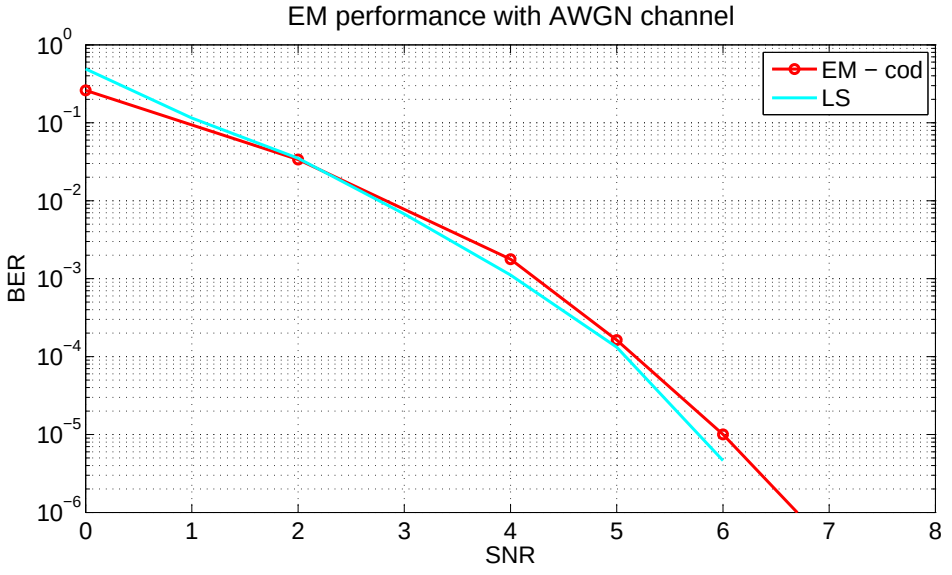

Figure 3.9: Bit error rate comparison between LS and EM algorithms when the channel is AWGN.

Except for the AWGN channel, in all the other cases the EM algorithm performs better than the LS one, but the performance gain is very small and it doesn't justify the choice of a complex algorithm such as the EM one.

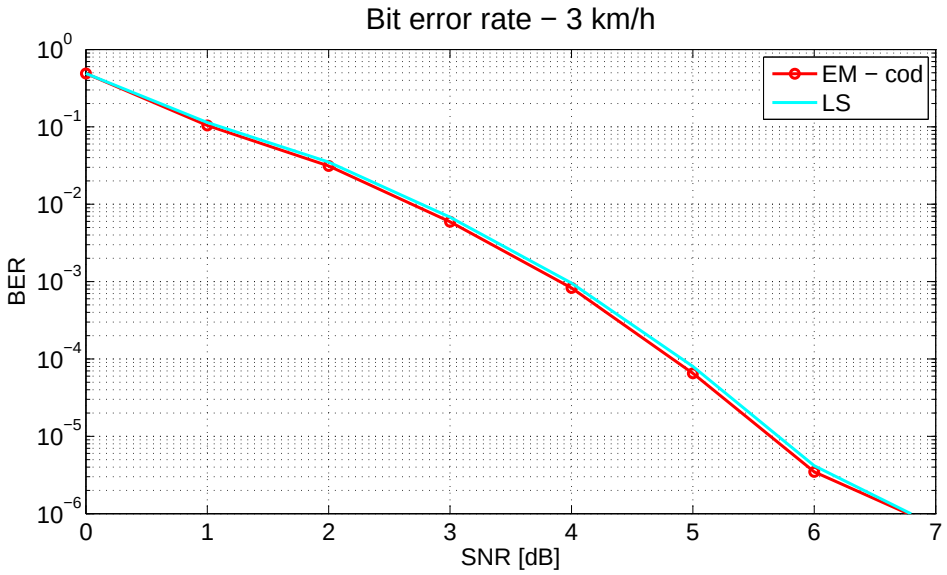

Figure 3.10: Bit error rate comparison between LS and EM algorithms when the channel is SCME and $v = 3$ km/h.

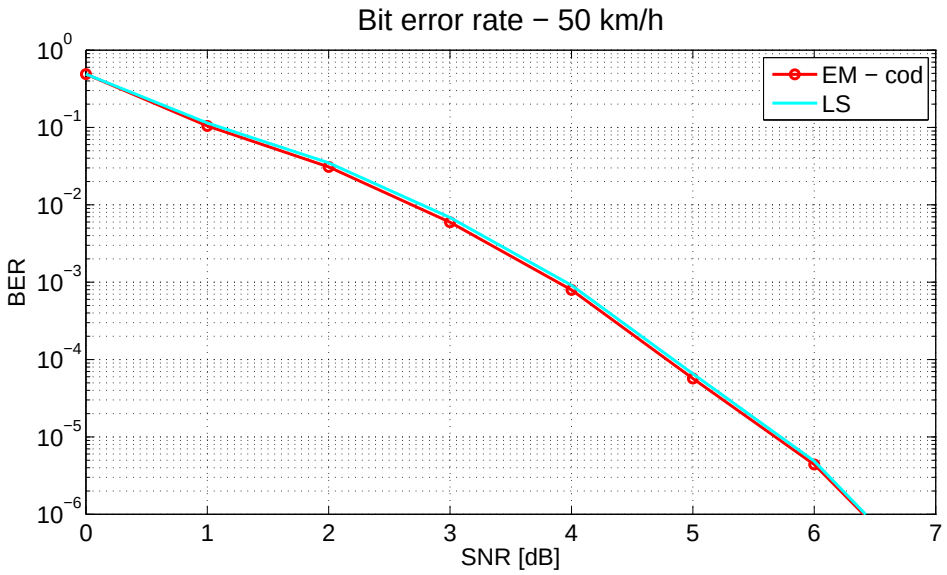

Figure 3.11: Bit error rate comparison between LS and EM algorithms when the channel is SCME and $v = 50$ km/h.

Once again we conclude that for a WiMAX system the channel can be estimated through the LS algorithm.

To conclude this discussion we want to underline that all this study has been developed to understand which is the best algorithm to be used to estimate the channel in a WiMAX system. Aim of the study was a trade-off between the performance in terms of bit error rate and the computational complexity. The results we found, as said before, suggest us the LS algorithm.

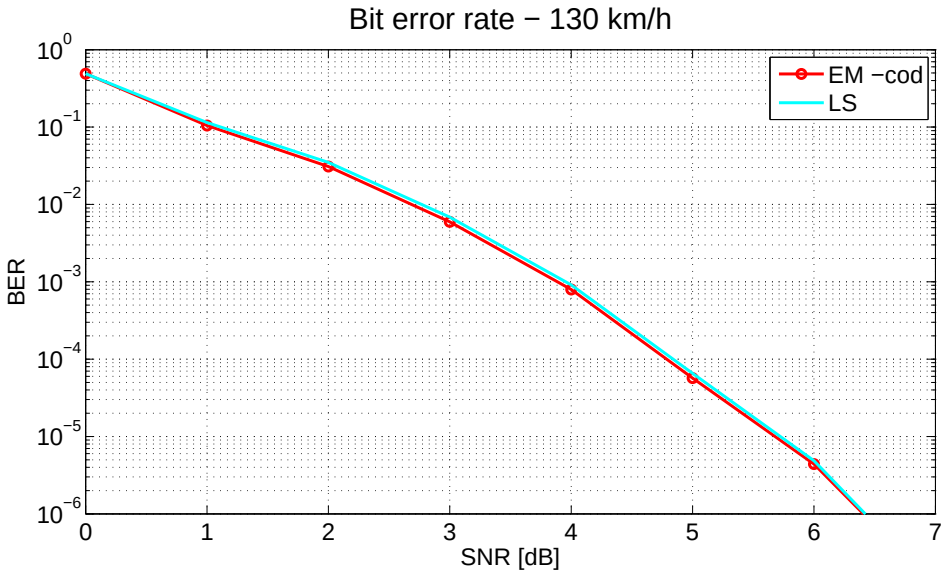

Figure 3.12: Bit error rate comparison between LS and EM algorithms when the channel is SCME and $v = 130$ km/h.

Chapter 4

Channel coding - Low-Density Parity-Check (LDPC) Codes

Coding for error correction is one of the many tools available for achieving reliable data transmission in communication systems. For a wide variety of channels, the noise channel coding theorem of the Information Theory proves that, if properly coded, for information transmitted at a rate below channel capacity, the probability of decoding error approaches zero exponentially with the code length.

In this context another important factor is the relation between the code length and the computation time or the equipment costs necessary to achieve this low error probability.

The low-density parity-check codes, considered here, are special example of parity check codes [83]. The codewords of a parity-check code are formed by combining a block of binary information digits with a block of check digits. Each check digit is the modulo two sum of the pre-specified set of information digits. These formation rules for the check digits can be conveniently represented by a parity-check matrix, which represents a set of linear homogeneous modulo 2 equations called parity-check equations and the set of codewords is the set of solution of these equations.

The use of parity-check codes makes coding relatively simple to implement; unfortunately, the decoding of parity check codes is not inherently simple to implement, and thus we must look for special classes of parity-check codes, for which reasonable decoding procedures exist.

4.1 Low-Density Parity-Check (LDPC) codes

Low-Density Parity-Check codes (LDPC) have recently drawn much attention due to their near-capacity error correction performance and are currently in the focus of many standardization activities. These codes can rightfully take their stand next to turbo codes as the most powerful error control codes known. They offer performance spectacularly close to theoretical limits when decoded using iterative soft-decision decoding algorithms based on factor graphs.

LDPC codes and a corresponding iterative decoding algorithm were first introduced by Gallager more than forty years ago [3, 4]. However, for the next several decades LDPC codes were largely forgotten, possibly because computers of the time could not simulate the performance of these codes with meaningful block lengths at low error rates. Following the discovery of turbo codes, LDPC codes were rediscovered through the work of MacKay and Neal [84, 5] and have become a major research topic. LDPC codes significantly differ from the more conventional trellis and block codes. First, they are constructed in a random manner, and second, they have a decoding algorithm whose complexity is linear in the block length of the code, which allows the decoding of large codes.

The random construction is one of their problems, because we only have a general method for constructing a class of pseudorandom LDPC codes, but we can't construct good codes algebrically or systematically. Thus, all the good codes we know are largely computer generated and their encoding is very complex.

On the other hand, their decoding is always performed through the Message Passing Algorithm (MPA). This is an Iterative Decoding based on a Belief Propagation (IDBP) [18] algorithm and, even if its computational complexity is very high, it results very efficient in LDPC decoding. It is a symbol-by-symbol soft-in soft-out decoding algorithm which iteratively processes the received symbol in order to improve its reliability, based on the parity-check sums computed from the hard decision of the received symbol.

4.1.1 Code definition and specification: Tanner's method

A low-density parity-check code is a code specified by a parity check matrix $\boldsymbol{H}$ of dimension $M \times N$, where $M = N - K$ is the number of parity-check digits, K is the sourceword length and N is the codeword length; the matrix $\boldsymbol{H}$ has the following properties:

- each column contains a small fixed number $p \geq 3$ of 1's,
- each row contains a small fixed number $q > p$ of 1's,

so it contains mostly 0's and only a small number of 1's. In particular an (N, p, q) low density code is a code of block length N and in its matrix each column contains a small fixed number p of 1's and each row contains a small fixed number q of 1's. A code defined in this way has a rate

$$\begin{aligned} R &= \frac{K}{N} \\ &= 1 - \frac{M}{N} \\ &= 1 - \frac{p}{q} \end{aligned} \tag{4.1}$$

An important property of the parity-check matrix is that the number of its non-zero entries must be the same whether calculated by rows or by columns, i.e. the equation $M \cdot q = N \cdot p$ is satisfied and so it must be $p < q$ to have $R < 1$.

These codes are not optimum in the sense of minimizing the probability of decoding errors, however, a very simple decoding scheme exists and this compensates for their lack of optimality.

The analysis of low-density code of long block length is difficult because of the immense number of codewords involved. It is simpler to analyze a whole ensemble of such codes because the statistics of an ensemble permit one to average over quantities that are not tractable in individual codes.

In order to define an ensemble of (N, p, q) low-density codes, given the parity-check matrix, we can observe that it can be divided into p submatrices, each containing a single 1 in each column. The first of this submatrices contains all its 1's in descending order and the other matrices are merely columns permutations of the first, as we can see in expression (4.35).

Definition 1. *Define an* ***ensemble of*** (N, p, q) ***codes*** *as an ensemble resulting from random permutation of the columns of each of the bottom* $p-1$ *submatrices of a parity-check matrix, with equal probability assigned to each permutation.*

There are two interesting results that can be proved using this ensemble, the first concerning the minimum distance of the member codes, and the second concerning the probability of decoding error.

Definition 2. *The* ***minimum distance of a code*** *is the number of positions in which the two nearest codewords differ.*

The minimum distance of a member code is a random variable and the distribution function of a random variable can be overbounded by a function as sketched in Fig. 4.1. As the block length increases, for fixed $p \geq 3$ and $q > p$, this function approaches a unit step at a fixed fraction δ_{pq} of the block length.

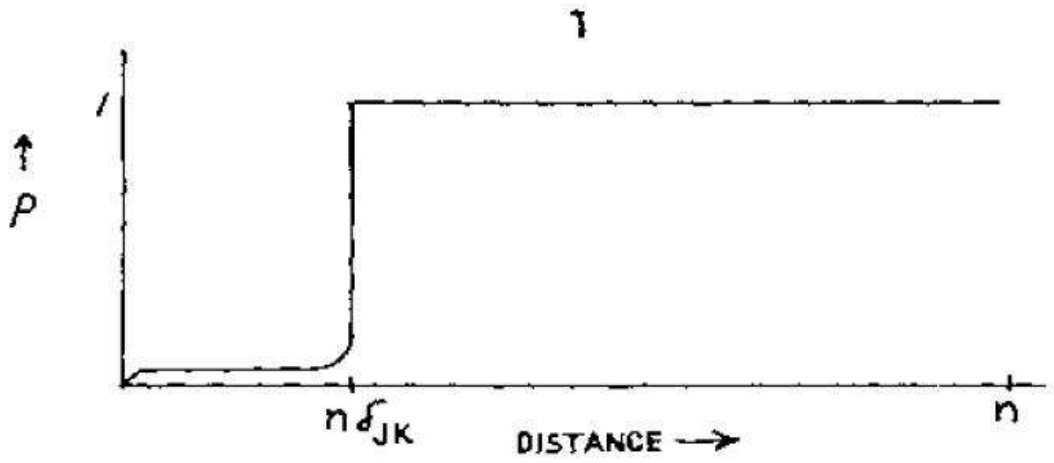

Figure 4.1: Sketch of bound to minimum distance distribution function.

The probability of error using maximum likelihood decoding for low-density codes clearly depends upon the particular channel on which the code is being used. The results are particularly simple for the case of the binary symmetric channel (BSC), which is a binary-input binary-output, memoryless channel with a fixed probability of transition from either input to the opposite output.

Definition 3 (Regular LDPC)**.** *A regular low-density parity-check code is completely defined by a permutation $\pi(i)$ of the natural numbers $1 \leq i \leq pN$. The index i refers to the socket number at the variable nodes, and $\pi(i)$ to the socket number at the check nodes to which socket i connects.*

Definition 4 (Irregular LDPC)**.** *An irregular low-density parity-check code is completely defined by a permutation $\pi(i)$ of the natural numbers $1 \leq i \leq pN$ from variable to check node socket numbers and two degree distributions $\lambda(x)$ and $\rho(x)$ for the variable nodes and check nodes, respectively.*

The code specification lies in the interconnection network, which is composed by N variable nodes with p connections from every one to check nodes and M check nodes with q connections from every one to variable nodes.

We assume that the block length N and the code rate are determined by the application, then it remains to determine appropriate values for p and q.

Since Gallager showed that the minimum distance of the typical regular LDPC codes increases linearly with N, provided $p \geq 3$, then regular LDPC are constructed with p on the order of 3 or 4.

The code specification given above is the starting point to introduce the Tanner's method. It is, in fact, based on a bipartite graph to provide a graphical representation of the parity-check matrix.

Definition 5. *The **bipartite graph** is a graph in which the nodes may be partitioned into two subsets such that there are no edges connecting nodes within a subset.*

For LDPC we have two subsets: variable nodes, one for each of the N columns of $\boldsymbol{H}$ and check nodes, one for each of the M rows of $\boldsymbol{H}$; an edge exists between the i-th variable node and the j-th check node if and only if $h_{ij} = 1$.

Definition 6. *The number of edges incident upon a node is the **degree of the node**.*

When we consider a regular LDPC, the bipartite graph of a (N, p, q) LDPC code contains N variable nodes of degree p and M check nodes of degree q. We observe also that (N, p, q) defines an ensemble of codes $C^N(p, q)$. Once the degree of the nodes are chosen, we are still free to choose which particular connections are made in the graph.

An irregular LDPC code can not be defined in terms of degree parameters p and q. We must, instead, use the degree distributions to describe the variety of node degrees in the graph.

Definition 7. *A **degree distribution** is a polynomial in x:*

$$\gamma(x) = \sum_i \gamma_i x^{i-1}$$

such that $\gamma(1) = 1$. *The coefficients* γ_i *equal the fraction of edges in the graph which are connected to a node of degree* i.

Thus, to define an ensemble $C^N(\lambda, \rho)$ of irregular LDPC codes we need: the code length N and the degree distribution for the variable nodes $\lambda(x)$ and for the check nodes $\rho(x)$.

Called N_e the number of edges in the code and the ratios λ_i/i and ρ_i/i the number of variable and check nodes of degree i respectively, we can define:

- the number of variable nodes is

$$N = N_e \sum_i \frac{\lambda_i}{i} \tag{4.2}$$

- the number of check nodes is

$$M = N_e \sum_i \frac{\rho_i}{i} \tag{4.3}$$

- the rate is

$$R = \frac{N - M}{M} = 1 - \frac{M}{N} = 1 - \frac{\sum_i \frac{\rho_i}{i}}{\sum_i \frac{\lambda_i}{i}} \tag{4.4}$$

Observation 1. *We observe that large irregular codes can approach the Shannon limit, but very large levels of irregularity are required.*

4.2 Irregular LDPC codes

In [85] the authors present irregular low-density parity-check (LDPC) codes which exhibit a performance extremely close to the best possible as determined by Shannon capacity formula.

Irregular LDPC codes were introduced in [86], [87] and were further studied in [88]-[89]. As we can see in Definition 4, for such an irregular LDPC code, the degrees of each set of nodes are chosen according to some distribution.

For a given length and a given degree distribution, we define an ensemble of codes by choosing edges, i.e., the connections between variable and check nodes, randomly. Assume that the number of edges is E, then a code can be identified with a permutation on E letters. By definition, all elements in this ensemble are equiprobable. In practice, the edges are not chosen entirely randomly, since certain potentially unfortunate events in the graph construction can be easily avoided.

We say that a polynomial $\gamma(x)$ of the form

$$\gamma(x) = \sum_{i \geq 2} \gamma_i x^{i-1}$$

is a degree distribution if $\gamma(x)$ has non-negative coefficients and $\gamma(1) = 1$. Given a degree distribution pair (λ, γ) associate to it a sequence of code ensembles $\mathcal{C}^N(\lambda, \gamma)$,

where N is the length of the code and where

$$\lambda(x) = \sum_{i\geq 2}^{d_v} \lambda_i x^{i-1} \qquad \left(\gamma(x) = \sum_{i\geq 2}^{d_c} \gamma_i x^{i-1}\right)$$

specifies the variable (check) node degree distribution. More precisely λ_i (γ_i) represents the fraction of edges emanating from variable (check) nodes of degree i. The maximum variable degree and check degree is denoted by d_v and d_c, respectively.

The total number of edges E emanating from all variable nodes, is equal to

$$N \sum_{i\geq 2} \frac{\lambda_i / i}{\int_0^1 \lambda(x)dx} = N \frac{1}{\int_0^1 \lambda(x)dx},$$

and, if we consider the M check nodes, the total number of edges E is

$$M \frac{1}{\int_0^1 \gamma(x)dx}.$$

Assuming that all these check equations are linearly independent, we see that the design rate is equal to

$$r(\lambda, \gamma) = \frac{N - M}{N} = 1 - \frac{\int_0^1 \gamma(x)dx}{\int_0^1 \lambda(x)dx}$$

Thus, following we will give a brief description of the channel type, the distributions and their properties and finally we will describe the optimization techniques that we use to obtain the degree distribution pairs with large threshold.

4.2.1 Channel

We consider an order family of binary-input memoryless channels parametrized by a real parameter δ such that if $\delta_1 < \delta_2$ then the channel with parameter δ_2 is a physically degraded version of the channel with parameter δ_1. Furthermore, each channel in this family is output-symmetric, i.e.

$$p(y|x = 1) = p(-y|x = -1). \tag{4.5}$$

4.2.2 Distribution and their properties

For some channels, e.g., the BEC and the BSC, the density of received log-likelihood ratios is discrete. For others, e.g., the BIAWGNC, the density is continuous. In the first case, the message densities will themselves be discrete and in the second case, the message densities will be continuous. In order to be able to treat all these cases in a uniform manner we shall work with a fairly general class of distributions.

Let $\mathcal{F}$ denote the space of right-continuous, non-decreasing functions F defined on $\mathbb{R}$ satisfying $\lim_{x\to-\infty} F(x) = 0$ and $\lim_{x\to+\infty} F(x) \leq 1$. To each $F \in \mathcal{F}$ we associate a random variable z over $(-\infty, +\infty]$. The random variable z has law or distribution F,

i.e.,

$$Pr[z \in (-\infty, x]] = F(x).$$

The reason we allow $\lim_{x\to+\infty} F(x) \leq 1$ rather than $\lim_{x\to+\infty} F(x) = 1$ is to permit z to have some probability mass at $+\infty$, indeed

$$Pr[z = +\infty] = 1 - \lim_{x\to+\infty} F(x).$$

We will work with densities over $(-\infty, +\infty]$ which, formally, can be treated as (Radon-Nikodyn) derivatives of elements of $\mathcal{F}$. The derivative, when it exists, is the density of the associated random variable z over $(-\infty, +\infty)$ although there may be an additional point mass at $+\infty$: recall $Pr[z = +\infty] = 1 - F^{-}(\infty)$. If f is the density corresponding to the distribution F we will write $\int_{\mathbb{R}} h(x)f(x)dx$ as a proxy for $\int_{\mathbb{R}} h(x)dF(x)$.

We are interested in the evolution of the message distributions under the independence assumption. Therefore, we will now discuss how distributions evolve when independent random variables (in either representation) are summed and when the representation of such variables is changed.

Their convolution is defined by

$$(F \otimes G)(x) = \int_{\mathbb{R}} F(x-y)dG(y) = \int_{\mathbb{R}} G(x-y)dF(y).$$

If z_1 and z_2 are independent random variables over $(-\infty, +\infty]$ with distributions F_{z_1} and F_{z_2}, respectively, then the distribution of $z_1 + z_2$ is $F_{z_1} \otimes F_{z_2}$ (as is the case for independent random variables defined over $(-\infty, +\infty)$).

Now, suppose we have a random variable z over $(-\infty, +\infty]$ with distribution F_z and we wish to describe the distribution of the random variable $\gamma(z) = (\gamma_1(z), \gamma_2(z))$, where

$$\begin{aligned} \gamma_1(z) &= \operatorname{sgn}(z) \\ \gamma_2(z) &= -\ln\tanh|\frac{z}{2}|. \end{aligned}$$

We approach this problem by assigning two connected distributions associated to $\gamma_2(z)$ under the conditions $\gamma_1(z) = 0$ and $\gamma_1(z) = 1$, respectively.

Any function G over $\mathrm{GF}(2) \times [0, +\infty)$ can be written as

$$G(s, x) = \chi_{\{s=0\}} G^0(x) + \chi_{\{s=1\}} G^1(x)$$

where $\chi_{\{s=a\}}$ denotes the characteristic function of the set $\{s = a\}$, i.e., $\chi_{\{s=a\}} = 1$ if $s = a$ and $\chi_{\{s=a\}} = 0$ otherwise. Let $\mathcal{G}$ denote the space of functions over $\mathrm{GF}(2) \times [0, +\infty)$ such that $G^0(x)$ and $G^1(x)$ are non-decreasing and right continuous

$$\lim_{x\to+\infty} G^0(x) \geq \lim_{x\to+\infty} G^1(x)$$

and such that $G^0(0) \geq 0$ and $G^1(0) = 0$.

Given a random variable $z \in (-\infty, +\infty]$ with distribution F_z we define the distribution of $\gamma(z)$ as

$$\Gamma(F_z)(s,x) = \chi_{\{s=0\}}\Gamma_0(F_z)(x) + \chi_{\{s=1\}}\Gamma_1(F_z)(x) \tag{4.6}$$

where

$$\Gamma_0(F_z)(x) = 1 - F_z^-\Big(-\ln\tanh\frac{x}{2}\Big) \qquad \text{and} \qquad \Gamma_1(F_z)(x) = F_z\Big(\ln\tanh\frac{x}{2}\Big).$$

Thus

$$\begin{aligned}\Gamma_0(F_z)(x) &= Pr[\gamma_1(z) = 0,\, \gamma_2(z) \leq x] \\ &= Pr[z \geq -\ln\tanh\frac{x}{2}]\end{aligned}$$

and

$$\begin{aligned}\Gamma_1(F_z)(x) &= Pr[\gamma_1(z) = 1,\, \gamma_2(z) \leq x] \\ &= Pr[z \leq \ln\tanh\frac{x}{2}].\end{aligned}$$

Note that $\Gamma(F_z) \in \mathcal{G}$, and, in particular

$$\lim_{x\to+\infty}\Gamma_0(F_z)(x) - \lim_{x\to+\infty}\Gamma_1(F_z)(x) = Pr[z = 0].$$

Let $G = \chi_{\{s=0\}}G^0 + \chi_{\{s=1\}}G^1$ be an element of $\mathcal{G}$. We speak of densities over $\mathrm{GF}(2) \times [0, +\infty]$

$$g(s,x) = \chi_{\{s=0\}}g^0(x) + \chi_{\{s=1\}}g^1(x)$$

by substituting for G^0 and G^1 their associated densities.

The function Γ has a well-defined inverse. Given

$$G = \chi_{\{s=0\}}G^0 + \chi_{\{s=1\}}G^1 \in \mathcal{G}$$

we have

$$\Gamma^{-1}(G)(x) = \chi_{\{s>0\}}G^0\Big(-\ln\tanh\frac{x}{2}\Big) + \chi_{\{s<0\}}G^1\Big(-\ln\tanh\frac{-x}{2}\Big) \tag{4.7}$$

and

$$\Gamma^{-1}(G)(0) = lim_{x\to+\infty}G^0(x).$$

It is easy to check that $\Gamma^{-1} : \mathcal{G} \to \mathcal{F}$ and that $\Gamma^{-1}(\Gamma(F)) = F$ for all $F \in \mathcal{F}$. Further, Γ and Γ^{-1} are additive operators on the spaces $\mathcal{G}$ and $\mathcal{F}$, respectively.

For convenience, although it constitutes an abuse of notation, we will apply Γ and Γ^{-1} to densities.

The space $\mathcal{G}$ has a well-defined convolution. Here, the convolution of two distribu-

tions $\chi_{\{s=0\}}G^0 + \chi_{\{s=1\}}G^1$ and $\chi_{\{s=0\}}H^0 + \chi_{\{s=1\}}H^1$ is the distribution

$$\chi_{\{s=0\}}((G^0 \otimes H^0) + (G^1 \otimes H^1)) + \chi_{\{s=1\}}((G^0 \otimes H^1) + (G^1 \otimes H^0))$$

where, here, $\otimes$ denotes the (one-sided) convolution of standard distributions. In other words, the new convolution is a convolution over the group $\mathrm{GF}(2) \times [0, +\infty)$. By abuse of notation, we denote this new convolution by the same symbol $\otimes$. Again, we shall allow the convolution operator to act on the densities associated to elements of $\mathcal{G}$ with the implicit understanding that the above provides the rigorous definition.

If z_1 and z_2 are independent random variables over $\mathrm{GF}(2) \times [0, +\infty)$ with distributions G_{z_1}, $G_{z_2} \in \mathcal{G}$, respectively, then the distribution of $z_1 + z_2$ is $G_{z_1} \otimes G_{z_2}$.

Density evolution

The symbols P_l and Q_l will be shorthand notations for the densities of the random variables $m_{vc}^{(l)}$ and $m_{cv}^{(l)}$,i.e. message sent from variable node v to check node c and the one sent from check node c to variable node v at the l-th iteration, respectively. We will use the notation $\int P_l$ and $\int Q_l$ to denote the associated distributions.

By

$$m_{cv}^{l} = \gamma^{-1}\Big(\sum_{v' \in V_c\{v\}} \gamma(m_{v'c}^{l-1})\Big),$$

we see that the random variable describing the message passed from check node c to variable node v is the image under γ^{-1} of a sum of random variables from $\mathrm{GF}(2) \times [0, +\infty]$. These random variables are independent by the independence assumption. So, the density of their sum is the convolution of their densities.

Let the graph have degree distribution pair (λ, ρ) where

$$\lambda(x) = \sum_{i \geq 2} \lambda_i x^{i-1} \qquad \text{and} \qquad \rho(x) = \sum_{i \geq 2} \rho_i x^{i-1}.$$

Recall that the fraction of edges connected to a variable node of degree i is λ_i, and the fraction of edges connected to a check node of degree i is ρ_i. Thus, a randomly chosen edge in the graph is connected to a check node of degree i with probability ρ_i. Therefore, with probability ρ_i the sum in (4.2.2) has $(i-1)$ terms, corresponding to the edges connecting c to all its neighbors other than v. We conclude that, in this case, the density of $m_{cv}^{(l)}$ is equal to

$$\Gamma_{-1}(\Gamma(P_{l-1})^{\otimes(i-1)}). \tag{4.8}$$

Summing up over all the possibilities for the degrees of the check node c, we see that the density of the message $m_{vc}^{(l)}$ equals

$$Q_l = \Gamma^{-1}(\rho(\Gamma(P_{l-1}))) = \Gamma^{-1}\Big(\sum_{i \geq 2} \rho_i (\Gamma(P_{l-1}))^{\otimes(i-1)}\Big). \tag{4.9}$$

A recursion for P_l in terms of Q_l is derived similarly and is quite straightforward. The density of the message passed from check node c to variable node v at round l is

equal to Q_l. At v the incoming messages from all check nodes other than c are added to m_0, the received value for v, and the result is sent back to c. Since, by the independence assumption the random variables describing these messages are independent, the density of this message equals

$$P_l = P_0 \otimes \lambda(Q_l) = P_0 \otimes \sum_{i \geq 2} \lambda_i (Q_l)^{\otimes (i-1)} \tag{4.10}$$

where P_0 is the density of the random variable describing the channel. Combining (4.9) and (4.10) we obtain the desired recursion for P_l in terms of P_{l-1}.

In [85], the authors provide that the density function of the messages passed from variable to check nodes during the belief propagation are symmetric, provided that the channel is output-symmetric.

4.2.3 Optimization

In this section, we briefly describe the optimization techniques that we used to obtain degree distribution pairs with large thresholds.

First, we observe that the threshold is defined as the supremum of all channel parameters for which the probability of error under density evolution converges to zero. This is equivalent to requiring that the message distribution converges to Δ_∞. In practice, we can verify at best that the probability of error reaches a value below a prescribed ϵ, so the issue of convergence is not of great concern since we always allow a finite (but small) probability of error.

Secondly, in order to perform the computations we need to quantize the quantities involved. This quantization leads to a quantization error and this error might accumulate over the course of many iterations, rendering the computations useless. This problem can be circumvented in the following way. By carefully performing the quantization one can ensure that the quantized density evolution corresponds to the exact density evolution of a quantized message-passing scheme. Since belief propagation is optimal, such a quantized version is suboptimal and, hence, the reported thresholds can be thought of as lower bounds on the actual thresholds.

Another important result presented in [85] states that density evolution for belief propagation always converges to a fixed point.

Local optimization - hill-climbing approach

Fix a small target error probability ϵ and a maximum number of iterations m. Start with a given degree distribution pair and determine the maximum admissible channel parameter, i.e., the maximum channel parameter such that the error probability after m iterations is below ϵ. Now apply a small change to the degree distribution pair and check if it has either a larger admissible channel parameter or at least a smaller target error probability after m iterations. If so, declare the new degree distribution pair to be the currently best degree distribution pair, otherwise keep the original degree distribution pair. The same basic step is then repeated a large number of times.

The search for good degree distribution pairs can be substantially accelerated by appropriately limiting the search space.We found, for example, that very good degree distribution pairs exist with only a few nonzero terms. In particular, it suffices to allow two or three nonzero check node degrees (and these degrees can be chosen consecutively) and to limit the nonzero variable node degrees to 2, 3, the maximum such degree d_l, and, possibly, a few well-chosen degrees in-between.

Provided that the fixed points are stable, the message distributions at these points are continuous functions of the degree distribution pair. Hence, a small change in the degree distribution pair causes only small changes in the associated fixed-point distributions. Furthermore, if the fixed points are stable, then this affords a certain memorylessness to the density evolution process because they serve as local attractors. Small perturbations to the path will not matter once the domain of convergence of the fixed point is entered and, once the fixed point is found, the path that leads to it is irrelevant.

Assume we determine the critical points (near fixed points, or likely fixed points for a slightly worse initial distribution) for a particular degree distribution pair and we would like to determine the merit of a particular small change of the degree distribution pair. Rather than starting with the initial distribution and then checking if (and how fast) this initial distribution converges to Δ_∞, one can memorize the distributions at the critical points of the original degree distribution pair and then determine how the proposed change affects the speed of convergence locally at these points. Once a promising change has been found, the merit of this change can be verified by starting with the initial degree distribution pair. Typically, only a few iterations are necessary at each critical point to determine if the change of the degree distribution pair improves the convergence or not. This has to be compared to hundreds of iterations or even thousands of iterations which are necessary if one starts with the initial distribution.

In the optimization scheme we just described we made use of the distributions at the "critical points" to find promising changes of the degree distribution pair. The following schemes extend this idea even further; the resulting algorithms are reminiscent of the algorithms used in the BEC case. For simplicity, we will only describe the optimization of the variable node degree distribution. The extension to the check node degree distribution and to joint optimization should be quite apparent.

Assume that we are given a degree distribution pair (λ, ρ), a particular channel parameter σ, and a target probability of error ϵ. Let $\{p_l\}_{l=0}^m$ be the sequence of error probabilities of the belief propagation algorithm. More precisely, p_0 is the initial error probability, p_l is the probability of error after the lth iteration, and $p_m \leq \epsilon < p_{m-1}$. Assume that we want to find a new degree distribution $\tilde{\lambda}$ which achieves the target probability of error in fewer iterations or achieves a lower target in the same number of iterations. Define a matrix $A_{l,j}$, $1 \leq l \leq m$, $2 \leq j \leq d_r$. The entry $A_{l,j}$ is the error probability which results if we run the belief-propagation decoder for $(l-1)$ steps assuming that the variable node degree distribution is λ followed by one step in which we assume that the variable node degree distribution is a singleton with all its mass on the degree j. Note that the actual error probability after the lth iteration, p_l, can be

expressed in terms of $A_{l,j}$ as

$$p_l = \sum_{j=2}^{d_l} A_{l,j}\lambda_j.$$

Let us define a function $p(t)$ for $t \in [0, m]$ by linearly interpolating the p_l, setting $p(l) = p_l$. Define

$$L(\lambda) = \int_{p_l}^{p_0} \Big(- \frac{dp}{dt}(x)\Big)^{-1} dx.$$

We interpret L as the number of iterations required to take the initial probability of error p_0 down to p_m. Using the expression above, we can write down the gradient of $L(\lambda)$ with respect to λ. In particular, for a perturbation h we can compute

$$D_h L(\lambda) = \frac{d}{d\eta} L(\lambda + \eta h)|_{\eta=0}$$

as

$$D_h L(\lambda) = \int_{p_m}^{p_0} \Big(\frac{dp}{dt}(x)\Big)^{-2} D_h\Big(\frac{dp}{dt}(x)\Big) dx.$$

Returning to the discrete representation this is equivalent to

$$D_h L = \sum_{j=2}^{d_l} h_j \Big(\sum_{l=1}^{m} \frac{A_{l,j} - p_l}{p_{l-1} - p_l}\Big).$$

Thus, we observe that the gradient of $L(\lambda)$ is given by

$$\frac{d}{d\lambda_j} L(\lambda) = \sum_{l=1}^{m} \frac{A_{l,j} - p_l}{p_{l-1} - p_l}.$$

There are two ways we can exploit this expression. One is to use the (negative) gradient direction to do hill climbing, and the other is to globally optimize the linearized approximation of L. In either case, we must incorporate the constraints on λ.

Let λ be an alternative degree distribution. Clearly, λ has to be a probability mass function, i.e.

$$\sum_{j=2}^{d_l} \tilde{\lambda}_j = 1 \tag{4.11}$$

and, further, it has to correspond to a code of equal rate, i.e.

$$\sum_{j=2}^{d_l} \frac{\tilde{\lambda}_j}{j} = \sum_{j=2}^{d_l} \frac{\lambda_j}{j}. \tag{4.12}$$

Let be the negative gradient direction of L. If we set $\tilde{\lambda} = \lambda + \eta h$ (for positive η) then the above constraints may not be satisfied. However, among degree distributions satisfying the constraints the one closest to $\lambda + \eta h$ in Euclidean distance can be easily computed by alternating projections. Two projections are required: the first is orthogonal projection of h onto the subspace determined by $\sum_j h_j = 0$ (total probability constraint) and

$\sum_j \frac{1}{j} h_j = 0$ (rate constraint), and the second projection sets $\eta h_j = -\lambda_j$ if, prior to the projection, $\eta h_j + \lambda_j < 0$. Note that an alternative interpretation is to project the gradient direction h onto the convex polytypes of admissible directions. One can then compute the maximum step size η for which the constraints remain satisfied and then recompute the projection at that point. In this way, one can easily walk along the projected gradient direction to look for an improved degree distribution. Let us now consider the second way to exploit the gradient expression for L. Let

$$\tilde{p}_l = \sum_{j=2}^{d_l} A_{l,j} \tilde{\lambda}_j.$$

Then we have

$$L(\tilde{\lambda}) \simeq \sum_{l=1}^{m} \frac{\tilde{p}_l - p_l}{p_{l-1} - p_l}. \tag{4.13}$$

This approximation is valid as long as $\tilde{\lambda}$ does not differ too much from λ, i.e., assuming that the message distributions corresponding to λ and $\tilde{\lambda}$ are not too different, if

$$\max_l \frac{|p_l - \tilde{p}_l|}{p_{l-1} - p_l} < \delta \tag{4.14}$$

where $\delta \ll 1$, and if

$$\tilde{p}_l < p_{l-1}, \qquad 1 \le l \le m \tag{4.15}$$

Recall that we want to minimize $L(\tilde{\lambda})$. The same procedure is then applied repeatedly in an attempt to converge to a good degree distribution. Since both approaches are local optimizations it is appropriate to repeat the optimization with various initial conditions.

Global optimization - Differential Evolution (DE) algorithm

Differential Evolution (DE) is a robust optimizer for multivariate functions. We will not describe the details here, suffice it to say that the algorithm is in part a hill climbing algorithm and in part a genetic algorithm.

Our goal is to maximize the cost function which we define to be the threshold value for the channel. Since such optimizers, and DE in particular, operate best in a continuous parameter space of not too large dimension, and since frequent function evaluations are required in the optimization, we found it convenient to let the parameter space be a continuous space of small dimension. To accomplish this, we introduced fractional phantom distributions. Let the polynomials λ and ρ take on the general form $\sum_i \lambda_i x^{i-1}$ (similarly for ρ), where now both the λ_i and the degree i could take any positive real value. The real degree distribution is obtained from this phantom distribution as $\sum_i (\lambda_{i1} x^{\lfloor i \rfloor - 1} + \lambda_{i2} x^{\lceil i \rceil - 1})$, where λ_{i1} and λ_{i2} are uniquely determined via the equations

$$\lambda_{i1} + \lambda_{i2} = \lambda_i \qquad \text{and} \qquad \frac{\lambda_{i1}}{\lfloor i \rfloor} + \frac{\lambda_{i1}}{\lceil i \rceil} = \frac{\lambda_i}{i}.$$

This way, we are guaranteed to obtain a degree distribution which respects the rate-

constraints for the code.

By allowing fractional degrees we, in effect, force the program to choose (close to) optimal degrees. This results in a significant reduction of the dimensionality of the parameter space, hence the running time, and also in the sparsity of the degree distributions obtained.

4.3 LDPC encoding

For an LDPC code the encoding problem can be divided into two sub-problems: the construction of the parity-check matrix and the encoding operation.

In the following Sections we give a brief description of the proposed solutions for these problems. During these explanations we consider binary codes, so all the operations are GF(2).

4.3.1 Parity-check matrix' construction

As specified above, to define a LDPC code, we need a very sparse random parity check matrix $\boldsymbol{H}$ of size $M \times N$. There are several methods to construct it, following we describe two methods generally used when we attempt to create a regular code and an irregular code, given its degree distribution. For a regular code the process followed is described next:

- select a column weight $p \geq 3$
- create a rectangular matrix $\boldsymbol{H}$ $M \times N$ at random with exactly weight p per column and weight per row as uniform as possible
- if N/M is an integer, then the number of 1's per row can be constrained to be exactly pM/M and so we have a regular LDPC code, because the bipartite graph is regular
- use Gaussian elimination and the reordering of columns to derive an equipment parity-check matrix in a systematic form, i.e. $\boldsymbol{H} = [\boldsymbol{P}|\boldsymbol{I}_M]$.

At this point the matrix $\boldsymbol{H}$ is composed of two very sparse matrices $\boldsymbol{H} = [\boldsymbol{C}_1|\boldsymbol{C}_2]$, where $\boldsymbol{C}_2$ is $M \times M$ very sparse and invertible and $\boldsymbol{C}_1$ is $M \times K$ and still very sparse.

To improve the properties of the code, when generating the matrix $\boldsymbol{H}$, constrain all pairs of columns in the matrix to have an overlap less or equal to 1. Furthermore constrain the matrix $\boldsymbol{H}$ so that the topology of the corresponding bipartite graph does not contain short cycles.

When we construct an irregular code we need its degree distribution. In [85] the authors describe a method to compute it, given the rate of the code, for a code with infinite codeword length. Here we suppose to know it and give a general description for the construction of the matrix [90].

Once we have generated the parity-check matrix $\boldsymbol{H}$ and remembering the relation $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$, we can compute the generator matrix $\boldsymbol{G}^T$:

$$\boldsymbol{G}^T = \begin{bmatrix} \boldsymbol{I}_k \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_k \\ \boldsymbol{C}_2^{-1}\boldsymbol{C}_1 \end{bmatrix}, \tag{4.16}$$

where the identity matrix $\boldsymbol{I}_k$ makes the code systematic.

4.3.2 Encoding operation

One major criticism concerning LDPC codes has been their apparent high encoding complexity. Whereas turbo codes can be encoded in linear time, a straightforward encoder implementation for an LDPC code has complexity quadratic in the block length: given the K-bit binary message $\boldsymbol{s}$, the N encoded bit vector is

$$\boldsymbol{t} = \boldsymbol{G}^T\boldsymbol{s} \mod 2. \tag{4.17}$$

Several authors have addressed this issue and found at least two possible solutions: a cascade construction [91], [92] and a restriction of the shape of the parity-check matrix. Unfortunately both of these result in a performance loss, compared to a standard LDPC code with the same overall length.

A different method encodes the sequence $\boldsymbol{s}$ using the parity-check matrix. Assume we have a systematic code, the codeword can be written as $\boldsymbol{x} = [\boldsymbol{x}_u\, \boldsymbol{x}_p]$, where $\boldsymbol{x}_u$ is the information sequence and $\boldsymbol{x}_p$ is the parity sequence. The same distinction can be done for the parity-check matrix, which can be written as: $\boldsymbol{H} = [\boldsymbol{H}_u|\boldsymbol{H}_p]$. As a consequence we can write: $\boldsymbol{H}_p\boldsymbol{x}_p^T = \boldsymbol{H}_u\boldsymbol{x}_u^T$ and so:

$$\boldsymbol{x}_p^T = \boldsymbol{H}_p^{-1}\boldsymbol{H}_u\boldsymbol{x}_u^T. \tag{4.18}$$

In literature we can find few methods to make this computation efficient. The most important from an efficient point of view is an encoder based on approximate lower triangularization. It uses the sparseness of the parity-check matrix to develop an algorithm for efficient encoders. The authors of [93] suggests placing the matrix in "approximate lower triangular" form, in which the upper right corner is populated with only 0's as shown in Figure. 4.2.

Assume that, by performing row and column permutations only, we can bring the parity-check matrix into the form indicated in Fig. 4.2. We say that $\boldsymbol{H}$ is in approximate lower triangular form. Note that since this transformation was accomplished solely by permutations, the matrix is still sparse. More precisely, assume that we bring the matrix in the form

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{pmatrix} \tag{4.19}$$

where $\boldsymbol{A}$ is $(M-g)\times(N-M)$, $\boldsymbol{B}$ is $(M-g)\times g$, T is $(M-g)\times(M-g)$, $\boldsymbol{C}$ is

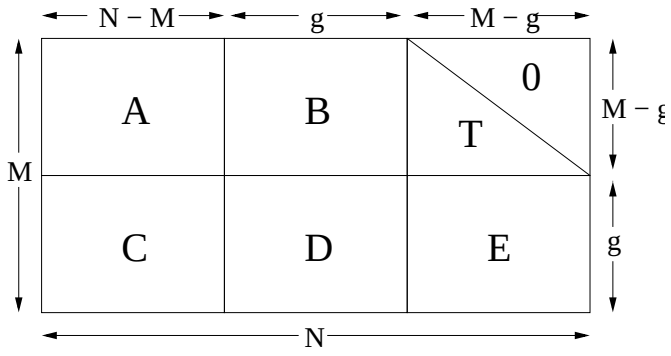

Figure 4.2: Parity-check matrix in approximate lower triangular form.

$g \times (N-M)$, $\boldsymbol{D}$ is $g \times g$, and, finally, $\boldsymbol{E}$ is $g \times (M-g)$[1]. Further, all these matrices are sparse and $\boldsymbol{T}$ is lower triangular with ones along the diagonal. Multiplying this matrix from the left by

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{T}^{-1} & \boldsymbol{I} \end{pmatrix} \tag{4.20}$$

we get

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C} & -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D} & \boldsymbol{0} \end{pmatrix}. \tag{4.21}$$

Let $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{p}_1, \boldsymbol{p}_2)$ where $\boldsymbol{s}$ denotes the systematic part, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ combined denote the parity part, $\boldsymbol{p}_1$ has length g, and $\boldsymbol{p}_2$ has length $(M-g)$. The well know equation $\boldsymbol{H}\boldsymbol{x}^T = 0^T$ splits naturally into two equations, namely

$$\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T + \boldsymbol{T}\boldsymbol{p}_2^T = 0 \tag{4.22}$$

and

$$(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C})\boldsymbol{s}^T + (-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D})\boldsymbol{p}_1^T = 0 \tag{4.23}$$

Define $\boldsymbol{\phi} = -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D}$ and assume for the moment that $\boldsymbol{\phi}$ is non-singular. We will discuss the general case shortly. Then from (4.23) we conclude that

$$\boldsymbol{p}_1^T = -\boldsymbol{\phi}^{-1}(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C})\boldsymbol{s}^T. \tag{4.24}$$

Hence, once the matrix $-\boldsymbol{\phi}^{-1}(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C})$ has been precomputed, the determination of $\boldsymbol{p}_1$ can be accomplished in complexity $O(g \times (N-M))$ simply by performing a multiplication with this (generically dense) matrix. In a similar manner, noting from (4.22) that $\boldsymbol{p}_2^T = -\boldsymbol{T}^{-1}(\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T)$, we can accomplish the determination of $\boldsymbol{p}_2^T$ in complexity $O(N)$.

Table 4.1 contains the complexity of each operation in the computation of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Table 4.2 gives a summary of the proposed algorithm: 2 steps are described: *pre-processing* and *encoding*. In the first one we perform rows and columns permutation to

[1] g is the minimum number of rows we can't use in our triangularization process

Efficient computation of $\boldsymbol{p}_1^T = -\boldsymbol{\phi}^{-1}(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C})\boldsymbol{s}^T$		
Operation	Comment	Complexity
$\boldsymbol{A}\boldsymbol{s}^T$	Multiplication by sparse matrix	$O(n)$
$\boldsymbol{T}^{-1}[\boldsymbol{A}\boldsymbol{s}^T]$	$\boldsymbol{T}^{-1}[\boldsymbol{A}\boldsymbol{s}^T] = \boldsymbol{y}^T \Leftrightarrow \boldsymbol{T}\boldsymbol{y}^T$	$O(n)$
$-\boldsymbol{E}[\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{s}^T]$	Multiplication by sparse matrix	$O(n)$
$\boldsymbol{C}\boldsymbol{s}^T$	Multiplication by sparse matrix	$O(n)$
$[-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{s}^T] + [\boldsymbol{C}\boldsymbol{s}^T]$	Addition	O(n)
$-\phi^{-1}[-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{C}\boldsymbol{s}^T]$	Multiplication by dense $g \times g$ matrix	$O(n)$
Efficient computation of $\boldsymbol{p}_2^T = -\boldsymbol{T}^{-1}(\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T)$		
Operation	Comment	Complexity
$\boldsymbol{A}\boldsymbol{s}^T$	Multiplication by sparse matrix	$O(n)$
$\boldsymbol{B}\boldsymbol{p}_1^T$	Multiplication by sparse matrix	$O(n)$
$[\boldsymbol{A}\boldsymbol{s}^T] + [\boldsymbol{B}\boldsymbol{p}_1^T]$	Addition	$O(n)$
$\boldsymbol{T}^{-1}[\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T]$	$\boldsymbol{T}^{-1}[\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T] = \boldsymbol{y}^T$ $\Updownarrow$ $-[\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^T] = \boldsymbol{T}\boldsymbol{y}^T$	O(n)

Table 4.1: Efficient computation of the parity sequence $\boldsymbol{p}^T = [\boldsymbol{p}_1^T\, \boldsymbol{p}_2^T]$.

bring the parity-check matrix into approximate lower triangular form with a gap g as small as possible. In the second one $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are computed. We point out that the pre-processing step is computed once at the beginning and then only the encoding step in performed.

4.4 The decoding problem

The use of the LDPC codes has two main advantages:

- LDPC codes can be decoded using an iterative decoding algorithm,
- complexity decoding grows only linear with the block length of the code.

The channel adds noise $\boldsymbol{n}$ to the vector $\boldsymbol{t}$ and the received signal is

$$\boldsymbol{r} = [\boldsymbol{G}^T\boldsymbol{s} + \boldsymbol{n}] \mod 2. \tag{4.25}$$

if we assume a binary symmetric channel (BSC).

The decoder's task is to infer $\boldsymbol{s}$ given the received signal $\boldsymbol{r}$ and the noise properties of the channel. The optimal decoder returns the message $\boldsymbol{s}$ that maximizes posterior probability:

$$P(\boldsymbol{s}|\boldsymbol{r}, \boldsymbol{G}) = \frac{P(\boldsymbol{r}|\boldsymbol{s}, G)P(\boldsymbol{s})}{P(\boldsymbol{r}|\boldsymbol{G})}. \tag{4.26}$$

We observe that if the prior probability of $\boldsymbol{s}$ is assumed uniform and the probability of $\boldsymbol{n}$ is assumed to be independent of $\boldsymbol{s}$, than it is convenient to introduce the $M \times N$ parity-check matrix $\boldsymbol{H}$, which is in systematic form $[\boldsymbol{P}|\boldsymbol{I}_M]$. We remember that the parity-check matrix has the property $\boldsymbol{H}\boldsymbol{G}^T = \boldsymbol{0} \mod 2$ so that $\boldsymbol{H}\boldsymbol{n} = \boldsymbol{H}\boldsymbol{r} \mod 2$.

Pre-processing: Input: Non singular parity-check matrix $\boldsymbol{H}$. Output: An equivalent parity-check matrix of the form $\begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix}$ such that $-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B} + \boldsymbol{D}$ is non-singular.

1. *Triangularization*: Perform row and column permutations to bring the parity-check matrix $\boldsymbol{H}$ into approximate lower triangular form

$$\boldsymbol{H} = \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix}$$

with as small a gap g as possible.

2. *Check rank*: Use Gaussian elimination to effectively perform the pre-multiplication

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{E}\boldsymbol{T}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C} & ET^{-1}\boldsymbol{D} & \boldsymbol{0} \end{pmatrix}$$

in order to check that $-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D}$ is non-singular, performing further column permutations if necessary to ensure this property.(Singularity of $\boldsymbol{H}$ can be detected at this point.)

Encoding: Input: Parity-check matrix of the form $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{pmatrix}$ such that $-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B} + \boldsymbol{D}$ is non-singular and a vector $\boldsymbol{s} \in F^{n-m}$. Output: The vector $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{p}_1, \boldsymbol{p}_2)$, $\boldsymbol{p}_1 \in F^g$, $\boldsymbol{p}_2 \in F^{m-g}$, such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{0}^T$.

1. Determine $\boldsymbol{p}_1^T = -\boldsymbol{\phi}^{-1}(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C})\boldsymbol{s}^T$
2. Determine $\boldsymbol{p}_2^T = -\boldsymbol{T}^{-1}(\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\boldsymbol{p}_1^{\boldsymbol{T}})$

Table 4.2: Summary of the proposed encoding procedure. It entails two steps: a pre-processing step and the actual encoding step.

The decoding problem, thus, reduces to the task of finding the most probable noise vector $\boldsymbol{n}$ such that

$$\boldsymbol{H}\boldsymbol{n} \mod 2 = \boldsymbol{z} \tag{4.27}$$

where the syndrome vector $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{r} \mod 2$.

Observation 2 (Decoding problem's formalization)**.** *Assume that $\boldsymbol{n}$ is a sparse random vector with independent and identically distributed bits of density f_n. We want to recover $\boldsymbol{t}$ by finding the most probable $\hat{\boldsymbol{n}}$ that satisfies*

$$\boldsymbol{H}\hat{\boldsymbol{n}} = \boldsymbol{z} \mod 2 \tag{4.28}$$

and $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{r} \mod 2$.

From $\hat{\boldsymbol{n}}$ we obtain out guess for the transmitted signal:

$$\hat{\boldsymbol{t}} = (\boldsymbol{r} + \hat{\boldsymbol{n}}) \mod 2 \tag{4.29}$$

and the first K bits of this are the guess $\hat{\mathbf{s}}$, since we are considering a systematic code.

Observation 3. *Theoretical effectiveness of LDPC codes as error correcting codes depends on the properties of very sparse matrices $\boldsymbol{H}$; the practical effectiveness, instead, depends on our finding the practical algorithm for solving $\boldsymbol{Hn} = \boldsymbol{z} \mod 2$ that is close enough to the optimal decoder.*

Observation 4. *There exist LDPC codes for which optimal decoders would achieve information rates arbitrarily close to the Shannon limit for a wide variety of channel.*

In the following Chapters the LDPC decoding algorithms present in literature are described.

4.5 LDPC decoding's algorithms

Following we describe the proposed LDPC decoding's algorithms, starting from the first proposed by Gallager[3, 4] and then analyzing the one proposed by MacKay [5].

4.5.1 The decoding algorithm by Gallager

In 1962 Gallager proposed two decoding schemes that appear to achieve a reasonable balance between complexity and probability of decoding [3]. The first is particularly simple, but it is applicable only to the binary symmetric channel (BSC) at rates far below channel capacity. The second scheme, which decodes directly from the a posteriori probabilities at the channel output, is more promising.

Algorithm for decoding LDPC codes on BSC channel

The decoder computes all the parity checks and then changes any digit that is contained in more than some fixed number of unsatisfied parity-check equations. Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.

If the parity-check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors. Thus when most of the parity-check equations checking on a digit are unsatisfied, there is a strong indication that that digit is in error.

In this algorithm, when the parity check sets contain one error the digit d is corrected, but it can be corrected even if its parity-check sets contain more than one transmission error. To see how this can be done consider the tree structure in Figure 4.3.

The arbitrary digit d is represented by the node at the base of the tree and each line rising from this node represents one of the parity check sets containing digit d. The other digits in these parity-check sets are represented by the nodes on the first tier of

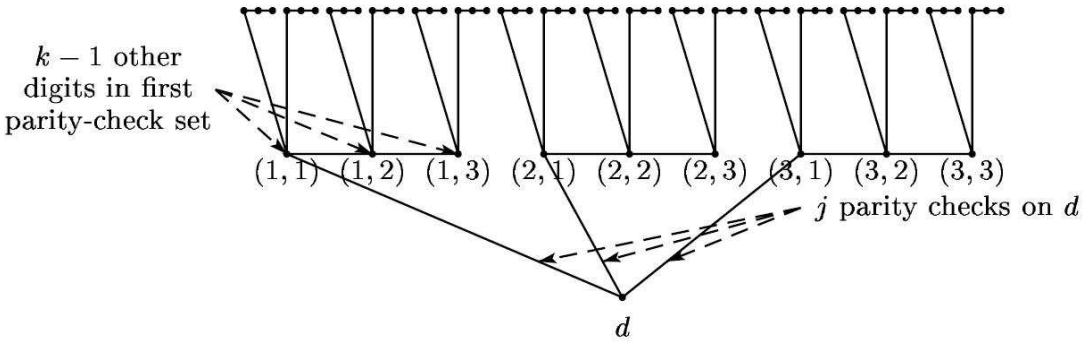

Figure 4.3: Parity-check set tree.

the tree. The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the digits on tier 1, and the nodes on tier 2 represent the other digits in those parity-check sets. Notice that if such a tree is extended to many tiers, the same digit will appear in more than one place.

Assume that both digit d and several of the digits in the first tier are transmission errors. Then, on the first decoding attempt, the error free digits in the second tier and their parity-check constraints will allow correction of the errors in the first tier. This in turn will allow correction of digit d on the second decoding attempt. Thus digit and parity-check equations can aid in decoding a digit seemingly unconnected with them.

If we define $V_{j\backslash i}$ the set of the variable nodes which connect to check node j, excluding variable node i and $C_{i\backslash j}$ the set of check nodes which connect to variable node i, excluding check node j and call r_i the received bit, we can synthesize the described algorithm as follows:

Gallager's LDPC decoding algorithm A for BSC channels

Step 1: Initialize $d_i = r_i$ for each variable node

Step 2: Variable nodes send $\mu_{i\to j} = d_i$ to each check node $j \in C_i$

Step 3: Check nodes connected to variable node i send $\beta_{j\to i} \prod_{l\in V_{j\backslash i}} \mu_{l\to j}$ to i; i.e. the check node sends back to i the value that would make the parity check consistent

Step 4: At the variable node i if $\lceil p/2 \rceil$ or more of the incoming parity checks $\beta_{j\to i}$ disagree with d_i, change the value of variable node i to its opposite value: $d_i = d_i \bigoplus 1$

Step 5: Stop when no more variable nodes are changing or after a fixed number of iterations have been executed. Otherwise go back to step 2.

Probabilistic decoding scheme

The probabilistic decoding scheme utilizes extra digits and extra parity check equations more systematically.

Assume that the codewords from an (N, p, q) code are used with equal probability on an arbitrary binary-input channel. For any digit d an iteration process will be derived that on the m-th iteration computes the probability that the transmitted digit in position d is a 1 conditional on the received symbols out to and including the m-th tier. For the first iteration, we can consider digit d and the digits in the first tier to form a subcode in which all sets of these digits that satisfy the p parity-check equations in the tree have equal probability of transmission.

Consider the ensemble of events in which the transmitted digits in the positions of d and the first tier are independent equiprobable binary digits, and the probabilities of the received symbols in these positions are determined by the channel transition probabilities $P_x(y)$. In this ensemble the probability of any event conditional on the event that the transmitted digits satisfy the p parity-check equations is the same as the probability of an event in the subcode described above. Thus, within this ensemble we want to find the probability that the transmitted digit in position d is a 1 conditional on the set of received symbols $\{y\}$ and on the event S that the transmitted digits satisfy the p parity-check equations on digit d. We write this as

$$P[x_d = 1|\{y\}, S]. \tag{4.30}$$

In [3] Gallager gave the proof of one lemma and one theorem, their propositions are reported here:

Lemma 1. *Consider a sequence of m independent binary digits in which the l-th digit is 1 with probability P_l. Then the probability that an even number of digits are 1 is*

$$\frac{1 + \prod_{l=1}^{m}(1 - 2P_l)}{2} \tag{4.31}$$

Theorem 1. *Let P_d, be the probability that the transmitted digit in position d is a 1 conditional on the received digit in position d, and let P_{il}, be the same probability for the l-th digit in the i-th parity-check set of the first tier. Let the digits be statistically independent of each other, and let S be the event that the transmitted digits satisfy the p parity-check constraints on digit d. Then*

$$\frac{P[x_d = 0|\{y\}, S]}{P[x_d = 1|\{y\}, S]} = \frac{1 - P_d}{P_d} \prod_{i=1}^{p} \Big[\frac{1 + \prod_{l=1}^{q-1}(1 - 2P_{il})}{1 - \prod_{l=1}^{q-1}(1 - 2P_{il})}\Big]. \tag{4.32}$$

Judging from the complexity of this result, it would appear difficult to compute the probability that the transmitted digit in position d is a 1 conditional on the received digits in two or more tiers of the tree. Fortunately, however, the many-tier case can be solved from the 1-tier case by a simple iterative technique.

As an example we consider the 2-tier case. We can use Theorem 1 to find the probability that each of the transmitted digits in the first tier of the tree is a 1 conditional on the received digits in the second tier. The only modification of the tier is that the first product is taken over only $p-1$ terms, since the parity-check set containing digit d is not included. Now these probabilities can be used in Equation (4.32) to find the probability that the transmitted digit in position d is 1. The validity of the procedure follows immediately from the independence of the new values of P_{il} in the ensemble used in Theorem 1. By induction, this iteration process can be used to find the probability that the transmitted digit d is 1, given any number of tiers of distinct digits in the tree.

The general decoding procedure for the entire code may now be stated. For each digit and each combination of $p-1$ parity-check sets containing that digit, use Equation (4.32) to calculate the probability of a transmitted 1 conditional on the received symbols in the $p-1$ parity-check sets. Thus there are p different probabilities associated with each digit, each one omitting 1 parity-check set. Next these probabilities are used in Equation (4.32) to compute a second order set of probabilities. The probability to be associated with one digit in the computation of another digit d is the probability found in the first iteration, omitting the parity-check set containing d. If the decoding is successful, then the probabilities associated with each digit approach 0 or 1 as the number of iterations is increased. The procedure is valid only for as many iterations as meet the independence assumption in Theorem 1.

This assumption break down when the tree closes upon itself. Since each tier of the tree contains $(p-1)(q-1)$ times more nodes than the previous tier, the independence assumption must break down while m is quite small for any code of reasonable block length. This lack of independence can be ignored, however, on the reasonable assumption that the dependencies have a relatively minor effect and tend to cancel each other out somewhat. Also, even if dependencies occur at the m-th iteration, the first $m-1$ iterations have reduced the equivocation in each digit. Then we consider the probabilities after the $m-1$ iterations to be a new received sequence that should be easier to decode than the original received sequence.

The most significant feature of this decoding scheme is that the computation per digit per iteration is independent of the block length. Furthermore, it can be shown that the average number of iterations required to decode is bounded by a quantity proportional to the log of the log of the block length.

For the actual computation of the probabilities in Theorem 1, it appears to be more convenient to use Equation (4.32) in terms of log-likelihood ratios. Let

$$\begin{aligned} \ln \frac{1-P_d}{P_d} &= \alpha_d \beta_d \\ \ln \frac{1-P_{il}}{P_{il}} &= \alpha_{il} \beta_{il} \\ \ln \frac{P[x_d = 0 | \{y\}, S]}{P[x_d = 1 | \{y\}, S]} &= \alpha'_d \beta'_d \end{aligned} \tag{4.33}$$

where α is the sign and β is the magnitude of the log-likelihood ratio. After some

manipulation, Equation (4.32) becomes

$$\alpha'_d\beta'_d = \alpha_d\beta_d + \sum_{i=1}^{j}\Big(\prod_{l=1}^{k-1}\alpha_{il}\Big) f\Big[\sum_{l=1}^{k-1} f(\beta_{il})\Big] \tag{4.34}$$

where $f(\beta) = \ln\frac{e^{\beta}+1}{e^{\beta}-1}$.

The calculation of the log-likelihood ratios in Equation (4.34) for each digit can be performed either serially in time or by parallel computations.

Example

Through the matrix (4.35) we define a $(20, 3, 4)$ code. It is an example of low-density code matrix where $N = 20$, $p = 3$ and $q = 4$.

$$\begin{array}{|c|}\hline 11110000000000000000\\ 00001111000000000000\\ 00000000111100000000\\ 00000000000011110000\\ 00000000000000001111\\ \hline 10001000100010000000\\ 01000100010000001000\\ 00100010000001000100\\ 00010000001000100010\\ 00000010001000100001\\ \hline 10000100000100000100\\ 01000010001000010000\\ 00100001000010000010\\ 00010000100001001000\\ 00001000010000100001\\ \hline\end{array} \tag{4.35}$$

Note that the matrix (4.35) is divided into p submatrices, each containing a single 1 in each column. The first of these submatrices contains all its 1's in descending order, the other submatrices are merely column permutations of the first.

We define an ensemble of (N, p, q) codes as the ensemble resulting from random permutation of the columns of each of the bottom $p-1$ submatrices of a matrix such as (4.35), with equal probability assigned to each permutation.

In Figure 4.4 we represent the Tanner's graph for the code defined with the matrix (4.35) and in Figure 4.5 we represent the tree used with the probabilistic decoder; we can observe that we must consider only the digit $d = 1$ and the first tier, because the second introduce a loop.

Remembering that P_d is the probability that the transmitted digit in position d is a 1 conditional on the received digit in position d, and, let P_{il} be the same probability

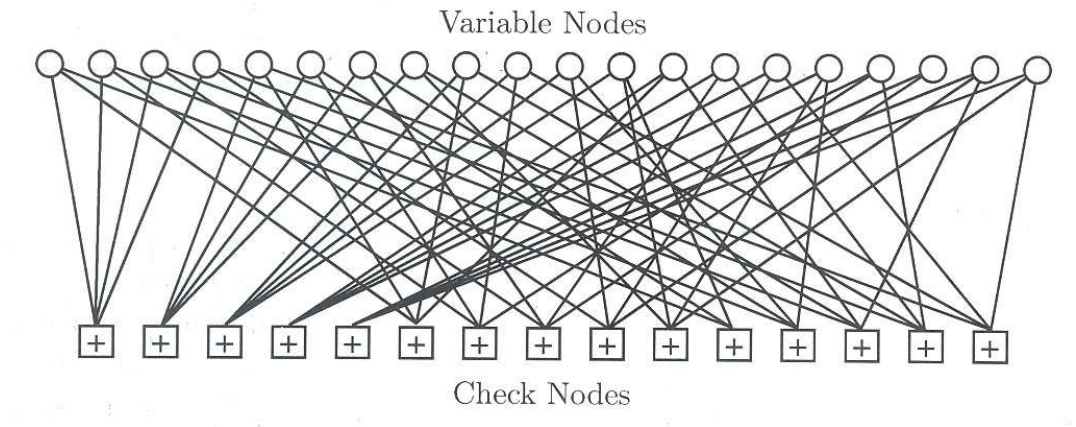

Figure 4.4: Tanner's graph for the code defined with the matrix (4.35).

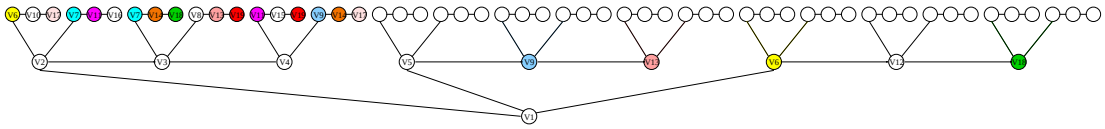

Figure 4.5: Tree used with the probabilistic decoder when the code defined by 4.35 is considered. The same colors are referred to the same variable nodes and through them the loops are shown.

for the l-th digit in the i-th parity check set, the Equation (4.32) becomes:

$$\frac{P[x_d=0|\{y\},S]}{P[x_d=1|\{y\},S]} = \frac{1-P_d}{P_d}\prod_{1=1}^{p}\left[\frac{1+\prod_{l=1}^{q-1}(1-2P_{il})}{1-\prod_{l=1}^{q-1}(1-2P_{il})}\right]$$

$$\frac{P[x_d=0|\{y\},S]}{P[x_d=1|\{y\},S]} = \frac{1-P_d}{P_d}\prod_{1=1}^{3}\left[\frac{1+\prod_{l=1}^{3}(1-2P_{il})}{1-\prod_{l=1}^{3}(1-2P_{il})}\right]$$

since $q=4$ and $p=3$.

4.5.2 The decoding algorithm by MacKay

First of all we recall the definition of the decoding problem:

$$\boldsymbol{Hn} = \boldsymbol{z} \mod 2 \tag{4.36}$$

where the unknown vector $\boldsymbol{n}$ is sparse, the vector $\boldsymbol{z} = \boldsymbol{Hr} \mod 2$ and $\boldsymbol{H}$ is very sparse and we must remember that its properties also depend on the assumed noise model.

In [5], MacKay has developed a "sum-product decoder", also known as a "belief propagation decoder". Before starting with the description of the proposed algorithm, we give some definitions:

Definition 8. ***Checks*** *are the z_j elements corresponding to each row of $\boldsymbol{H}$, where $j = 1, 2, \ldots, M$ of $\boldsymbol{H}$.*

Definition 9. ***Belief network*** *or Bayesan network or Causal network or Influence diagram is the set of bits* $\boldsymbol{x}$ *and checks* $\boldsymbol{z}$*: every bit* x_l *is the parent of* p *checks* z_j*, and each check* z_j *is the child of* q_r *bits, thus the network of checks and bits form a bipartite graph.*

Observation 5. *We say that each check* z_j *is the child of* q_r *bits and not* q*, because we are considering the case in which the number of* 1*'s in each row of* $\boldsymbol{H}$ *is as uniform as possible.*

We aim, given the observed checks, to compute the marginal posterior probabilities $P(x_l = 1|\boldsymbol{z}, \boldsymbol{A})$ for each l. These computations are expected to be intractable for the belief network corresponding to our problem because its topology contains many cycles. However, it is interesting to implement the decoding algorithm that would be appropriate if there were no cycles, on the assumption that the errors introduced might be relatively small. This approach of ignoring cycles has been used in the artificial intelligence literature [94] but is now frowned upon because it produces inaccurate probabilities. However, for our problem the end product is a decoding; the marginal probabilities are not required if the decoding is correct. Also, the posterior probability, in the case of a good code communicating at an achievable rate, is expected typically to be hugely concentrated on the most probable decoding. And as the size $M \times N$ of the code's matrix $\boldsymbol{H}$ is increased, it becomes increasingly easy to produce matrices in which there are no cycles of any given length, so we expect that, asymptotically, this algorithm will be a good algorithm. MacKay, in [5] has obtained excellent results with N equal to 1000 and 10000. The algorithm often gives useful results after a number of iterations much greater than the number at which it could be affected by the presence of cycles.

The algorithm

Also this algorithm, presented by MacKay in [5], is appropriate for binary channel model in which the noise bits are independent.

We denote the set of bits l that participate in check m by $\mathcal{L}(m) = \{l : A_{ml} = 1\}$. Similarly, we define the set of checks in which bit l participates, $\mathcal{M}(l) = \{m : A_{ml} = 1\}$. We denote a set $\mathcal{L}(m)$ with bit l excluded by $\mathcal{L}(m)\backslash l$. The algorithm has two alternating parts, in which quantities q_{ml} and r_{ml} associated with each nonzero element in the $\boldsymbol{H}$ matrix are iteratively updated.

Definition 10. *The quantity* q^x_{ml} *is meant to be the probability that bit* l *of* $\boldsymbol{x}$ *has the value* x*, given the information obtained via checks other than check* m*.*

Definition 11. *The quantity* r^x_{ml} *is meant to be the probability of check* m *being satisfied if bit* l *of* $\boldsymbol{x}$ *is considered fixed at* x *and the other bits have a separable distribution given by the probabilities* $\{q_{ml'} : l' \in \mathcal{L}(m) \setminus l\}$*.*

The algorithm would produce the exact posterior probabilities of all the bits after a fixed number of iterations if the bipartite graph defined by the matrix $\boldsymbol{H}$ contains no cycles.

Practical decoding by the sum-product algorithm - MacKay

Initialization: Let $p_l^0 = P(x_l = 0)$ and $p_l^1 = P(x_l = 1) = 1 - p_l^0$ the prior probability that bit x_l is 0 or 1. For every (m, l) such that $H_{ml} = 1$ the variables q_{ml}^0 and q_{ml}^1 are initialized to the vales p_l^0 and p_l^1, respectively

Horizontal step: Compute the difference $\delta q_{ml} = q_{ml}^0 - q_{ml}^1$ and

$$\delta r_{ml} = \prod_{l' \in \mathcal{L}(m) \setminus l} \delta q_{ml'}.$$

Then, since $r_{ml}^0 + r_{ml}^1 = 1$ define $r_{ml}^0 = (1 + \delta r_{ml})/2$ and $r_{ml}^1 = (1 - \delta r_{ml})/2$

Vertical step: For each l compute

$$q_{ml}^0 = \alpha_{ml} p_l^0 \prod_{m' \in \mathcal{M}(l) \setminus m} r_{m'l}^0$$

$$q_{ml}^1 = \alpha_{ml} p_l^1 \prod_{m' \in \mathcal{M}(l) \setminus m} r_{m'l}^1$$

where α_{ml} is chosen such that $q_{ml}^0 + q_{ml}^1 = 1$

In the description of the algorithm, the horizontal step the probability for z_m having its observed value given either $x_l = 0$ and $x_l = 1$ is founded efficiently by the use of the forward-backward algorithm. Before this formulation of the problem, the probability r_{ml}^0 and r_{ml}^1 were computed as follows:

$$r_{ml}^0 = \sum_{\{x_{l'}, l' \in \mathcal{L}(m) \setminus l\}} P(z_m | x_l = 0, \{x_{l'} : l' \in \mathcal{L}(m) \setminus l\}) \times \prod_{l' \in \mathcal{L}(m) \setminus l} q_{ml'}^{x_{l'}} \tag{4.37}$$

$$r_{ml}^1 = \sum_{\{x_{l'}, l' \in \mathcal{L}(m) \setminus l\}} P(z_m | x_l = 1, \{x_{l'} : l' \in \mathcal{L}(m) \setminus l\}) \times \prod_{l' \in \mathcal{L}(m) \setminus l} q_{ml'}^{x_{l'}}. \tag{4.38}$$

In the vertical step, we can also compute the pseudoposterior probabilities q_l^0 and q_l^1:

$$q_l^0 = \alpha_l p_l^0 \prod_{m \in \mathcal{M}(l)} r_{ml}^0 \tag{4.39}$$

$$q_l^1 = \alpha_l p_l^1 \prod_{m \in \mathcal{M}(l)} r_{ml}^1 \tag{4.40}$$

and these quantities are used to create a tentative decoding $\hat{\boldsymbol{n}}$, the consistency of which is used to decide whether the decoding algorithm can halt.

Soft input to the decoder

There are many ways to measure the uncertainty over a bit at the receiver. We can simply use the received symbol or computed its marginal a posteriori probability, but, what is generally used is the Log-Likelihood ratio (LLR) defined as:

$$\Lambda(r_i) \triangleq \log\Big(\frac{P(x_i = 1|r_i)}{P(x_i = 0|r_i)}\Big)^2. \tag{4.41}$$

Soft decoding algorithm

The algorithm presented above accepts as inputs the received symbols, but to make it more efficient we can have at the demodulator output soft LLRs and so we can adjust the decoding algorithm to work with these. We call this adjustment *soft decoding algorithm* because it has as input the log-likelihood ratios for the bits of each codeword c_i, $i = 1, 2, \ldots, N$ which is defined as:

$$L(c_i) = \log\Big(\frac{P[c_i = 0|\text{channel output for } c_i]}{P[c_i = 1|\text{channel output for } c_i]}\Big) \qquad i = 1, 2, \ldots, N \tag{4.42}$$

where N is the length of the codewords.

Before explaining the soft algorithm, we define:

- C_i the set of check nodes connected to the variable node i;
- $C_{i\backslash j}$ the set of check nodes connected to the variable node i excluded the check node j;
- V_j the set of variable nodes connected to the check node j;
- $V_{j\backslash i}$ the set of variable nodes connected to the check node j excluded the variable node i.

In the algorithm, described in Table 4.3, there are three key variables: $L(q_{ji})$, $L(r_{ji})$ and $L(Q_i)$ with $j = 1, 2, \ldots, M$ where M represents the number of the parity check nodes and $i = 1, 2, \ldots, N$. The $L(\cdot)$ means that we are considering the logarithm values. The former is the logarithm of the probability that the i-th bit of the codeword has the value x, given the information obtained via the check nodes other than check node j (Definition 10); the second is the logarithm of the probability that a check node j is satisfied, i.e. $z_j = 0$, when bit i is fixed to a value x and the other bits are independent with probabilities $q_{j,i'}$, $i' \in V_{j\backslash i}$ (Definition 11). Finally, the last one, expressed in (4.46) is the logarithm of the pseudo-a-posteriori probability and it is the new log-likelihood ratio, used to determine the transmitted bit in the hard decision of the bit c_i.

Decoding step

We observe that if the belief network really is a tree without cycles the values of the pseudoposterior probabilities q_l^0 and q_l^1 at each iteration would correspond to the

[2] Sometimes the definition of the LLR is given as $\log\left(\frac{P(x_i=0|r_i)}{P(x_i=1|r_i)}\right)$, but this doesn't change anything, because it is equal to $-\Lambda(r_i)$

- **Initialization**

$$L(q_{ji}) = L(c_i) \qquad j = 1, 2, \ldots, M \text{ and } i = 1, 2, \ldots, N \tag{4.43}$$

- **Horizontal step**

$$L(r_{ji}) = 2\tanh^{-1}\Big(\prod_{i' \in V_{j\setminus i}} \tanh\Big(\frac{1}{2}L(q_{ji'})\Big)\Big) \tag{4.44}$$

- **Vertical step**

$$L(q_{ji}) = L(c_i) + \sum_{j' \in C_{i\setminus j}} L(r_{j'i}) \tag{4.45}$$

In this step we also compute $L(Q_i)$, the quantity correspondent to the pseudo-posterior probability:

$$L(Q_i) = L(c_i) + \sum_{j' \in C_i} L(r_{j'i}) \tag{4.46}$$

Table 4.3: Soft decoding algorithm

posterior probabilities of bit l given the states of all the checks in a truncated belief network centered on bit l and extending out to a radius equal to twice the number of iterations.

Decoding Procedure

- Set $x_l = 1$ if $q_l^1 > 0.5$
- See if the checks $\boldsymbol{H}\hat{\boldsymbol{n}} = \boldsymbol{z} \mod 2$ are all satisfied
  - halting when they are
  - declaring a failure if some maximum number of iterations occurs without successful decoding

When there is a failure, the partial decoding $\hat{\boldsymbol{n}}$ may serve as a useful starting point for another decoding algorithm.

To conclude our analysis, we note that undetected errors would only occur if the decoder found an $\hat{\boldsymbol{n}}$ satisfying $\boldsymbol{H}\hat{\boldsymbol{n}} = \boldsymbol{z} \mod 2$ which is not equal to the true $\boldsymbol{n}$ and detected errors occur if the algorithm runs for the maximum number of iterations without finding a valid decoding.

Observation 6. *Soft decoding algorithm is very important, because it gives better performance since it uses soft values as input, but also because, considering the logarithm*

values it is efficient from a computational point of view: all the multiplications required in the vertical step and to computed the pseudo-a-posteriori probabilities are transformed into additions and it is well known that these are very simple if compared to multiplications.

Analysis of the decoding algorithm

The algorithm have correction effects which are independent of the block length N, for large N. This consideration brings us to make a conjecture:

Observation 7. *Given a binary symmetric channel with noise density f, there exist practical decoders for LDPC codes with rate R close to capacity, that can achieve negligible probability of error for sufficiently large N.*

Finally we can conclude saying that there is a correction effect if the Hamming distance between $\hat{\boldsymbol{n}}$ and the true vector $\boldsymbol{n}$ decreases.

Empirical performance of Low-Density Parity-Check codes on Gaussian channels

In [84] D. J. C. Mackay and R. M. Neal showed that performance substantially better than that of standard convolutional and concatenated codes con can be achieved; indeed the performance is almost as close to the Shannon limit as that of turbo codes.

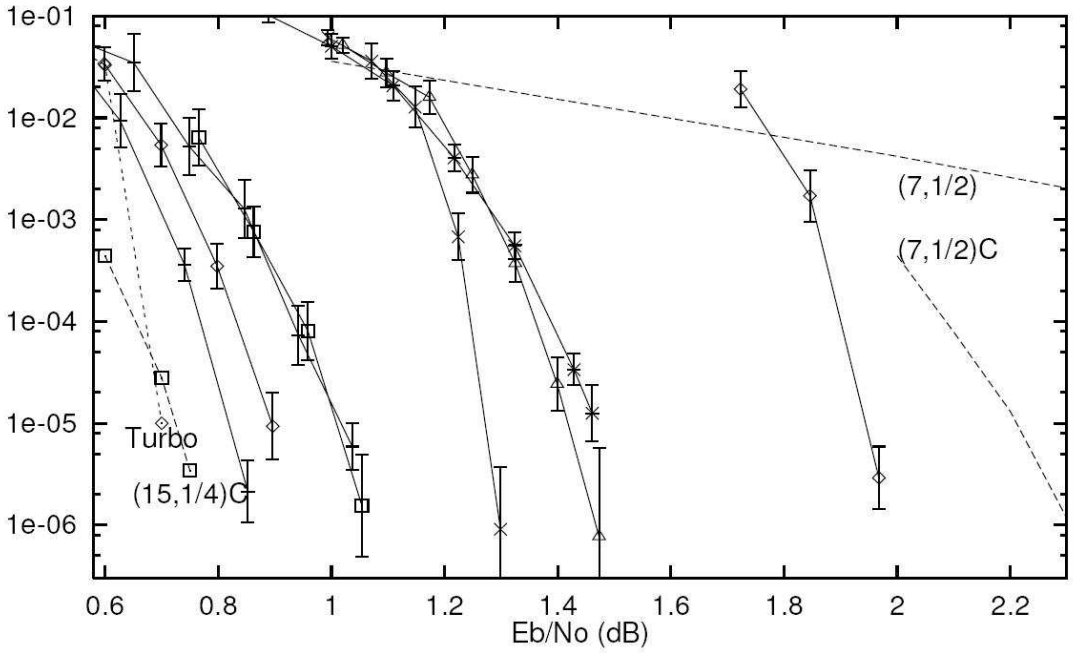

Figure 4.6: LDPC codes' performance over Gaussian channel (solid curves) compared with state of the art codes (dotted curves).

Figure 4.6 compares the performance of LDPC codes with state of the art codes.

The vertical axis shows the empirical bit error probability (BER). Following we describe the mean of the curves represented in Figure 4.6:

- The curve labeled $(7, 1/2)$ shows the performance of a rate 1/2 convolutional code with constraint length 7, known as the factor standard for satellite communications [95].
- The curve $(7, 1/2)C$ shows the performance of the concatenated code composed of the same convolutional code and a Reed-Solomon code.
- The curve $(15, 1/4)C$ shows the performance of an extremely expensive and computer intensive concatenated code developed at JPL based on a constraint length 15, rate 1/4 convolutional code.
- The curve labeled *Turbo* shows the performance of the rate 1/2 turbo code described in [19].

All the other curves represent LDPC with the parameters (N, K, R) with the following values (29507, 9507, 0.322), (15000, 5000, 0.333), (14971, 4971, 0.332), (65389, 32621, 0.499), (19839, 9839, 0.496), (13298, 3296, 0.248), (29331, 19331, 0.659) from left to right.

It should be emphasized that all the errors made by the LDPC codes were detected errors: the decoding algorithm reported the fact that it had failed.

Results show that performance substantially better than that of standard convolutional and concatenated codes can be achieved; indeed the performance is almost as close to the Shannon limit as that of turbo codes. It seems that the best results are obtained by making the weight per column as small as possible. Unsurprisingly, codes with larger block length are better. In terms of the value of E_b/N_0, the best codes are the ones with rates between 1/2 and 1/3.

Chapter 5

Alternative algorithms for LDPC decoding

5.1 Message passing algorithm: complexity reduction

As described in Chapter 4 the Message Passing Algorithm (MPA) is used to decode the LDPC codes, because, even if its computational complexity is very high, it is yet practically implementable and performs very well. Here we propose a modification of the MPA in order to reduce the number of multiplications in each iteration. In fact, MPA requires real-number addition, subtraction, multiplication, division, exponential and logarithm operations and, as it is well known, the last four operation are more complex than addition and subtraction. In [90] the number of multiplications in this algorithm is defined as a linear function of the number of 1's in the parity check matrix.

What we are going to propose is the reduction of the number of multiplications, so, in the following we will use the terms reduction of complexity or reduction of the number of multiplications indifferently, even if it's an abuse of notation. In our modified MPA we assume that all the received bits which exceed, in terms of absolute value, a given threshold are reliable. This means that we assume that the value assigned to these bits doesn't change with the evolution of the algorithm. This observation suggests to assign to these bits a value, in term of LLR, equal to $\pm\infty$, where the sign is in accordance with the LLR's one. In Section 5.1.2 we will show how to choose the threshold and how this assumption can be used to reduce the number of multiplications at each iteration. The modified MPA has a reduced complexity because the number of multiplications has been reduced, but, can present an error floor dependent on the threshold value. To verify if the modified proposed algorithm can be competitive with the known ones we will compare the complexity and the error performance.

5.1.1 LDPC decoding techniques

In Chapter 4 the decoding techniques presented for the LDPC codes have been described. Here we briefly describe the algorithms practically used in tests and simulations, except the MPA, exhaustly described above. Here, to make the reading more

comfortable, we report only the brief description.

A LDPC code can be decoded in various ways, namely: majority-logic (MLG) decoding, bit-flipping (BF) decoding, weighted BF decoding, a posteriori probability (APP) decoding, and iterative decoding based on belief propagation (IDBP), commonly known as sum-product algorithm (SPA) or message passing algorithm (MPA).

The first two types are hard decision decoding, the last two are soft decision decoding and the third one is in between. MLG [90] is the simplest one in decoding complexity; BF requires a little more decoding complexity, but gives better error performance than MLG. APP and MPA decoding provide much better error performance but require much large complexity than the former two. MPA decoding gives the second best error performance among the five types of decoding algorithms and yet is practically implementable, conversely the APP provides the best error performance, but it is computationally intractable. A simplified version can be considered to reduce the computational complexity, but error performance degrades. The most important algorithm for a comparison with our proposed one is the weighted-BF algorithm, because it's said to represent a good trade-off between the computational complexity and the error performance.

Let define a codeword $\boldsymbol{v} = (v_o, v_1, \ldots, v_{n-1})$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ the soft decision received sequence at the output of the demodulator and $\boldsymbol{z} = (z_o, z_1, \ldots, z_{n-1})$ the corresponding hard decision. Let $\boldsymbol{H}$ be the parity-check matrix of an LDPC code C with m rows and n columns, then we can define the syndrome of the received sequence $\boldsymbol{z}$ as

$$\boldsymbol{s} = \boldsymbol{z}\boldsymbol{H}^T. \tag{5.1}$$

The received vector $\boldsymbol{z}$ is a codeword if and only if $\boldsymbol{s} = \boldsymbol{0}$, otherwise errors in $\boldsymbol{z}$ are detected, in fact nonzero syndrome component s_j indicates a *parity failure*. We can define the error pattern $\boldsymbol{e}$ given by $\boldsymbol{e} = \boldsymbol{v} + \boldsymbol{z}$ which must satisfies $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^T$.

Once given these few definitions we can briefly describe the decoding algorithms.

Majority-Logic Decoding

From the structural properties of the parity-check matrix $\boldsymbol{H}$, we see that for every bit position l, there's a set

$$A_l = \{\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, \ldots, \boldsymbol{h}_{\gamma_l}^{(l)}\} \tag{5.2}$$

of γ_l rows in $\boldsymbol{H}$ that are orthogonal on this bit position. Now we define the set of γ_l check-sums orthogonal on the error digit e_l

$$S_l = \{\boldsymbol{e}\boldsymbol{h}_j^{(l)} : \boldsymbol{h}_j^{(l)} \in A_l, \text{ for } 1 \leq j \leq \gamma_l\} \tag{5.3}$$

These check-sums can be used for estimating the error digit e_l based on the one step MLG decoding rule [90], which guarantees the correction only if there are $\frac{\gamma_l}{2}$ or fewer errors in the pattern error $\boldsymbol{e}$.

Bit-Flipping Decoding Algorithm

This algorithm, presented for the first time by Gallager in [4], is based on the change (or flip) of the number of parity failures in $\boldsymbol{s}$ when a bit in the received sequence $\boldsymbol{z}$ is changed or flipped.

First, the decoder computes all the parity-check sums based on (5.1) and than changes any bit in the received sequence $\boldsymbol{z}$ that is contained in more than some fixed number δ of failed parity-check equations. Using the modified received sequence $\boldsymbol{z}'$, the decoders recompute the syndrome and the process is repeated until all parity-check sums are satisfied or a preset maximum number of iterations is reached. The design parameter δ is called *threshold* and should be chosen to optimize the error performance while minimizing the number of computation of parity-check sums. The value of δ depends on the code parameters: ρ, γ, $d_{min}(C)$ and SNR.

Owing to the nature of low-density parity-check codes, BF decoding algorithm corrects many error patterns whose number of errors exceeds the error correcting capability of the code.

Weighted Bit-Flipping (WBF) Algorithm

To improve the performance of the BF algorithm some kind of reliability information can be considered. So now we consider the soft-decision received sequence $\boldsymbol{y}$ and for $0 \leq l \leq n-1$ and $1 \leq j \leq m$ define:

$$|y_j|_{min}^{(l)} \triangleq \{\min\{|y_i|\} : 0 \leq i \leq n-1, h_{i,j} = 1\} \tag{5.4}$$

and

$$E_l \triangleq \sum_{s_j^{(l)} \in S_l} (2s_j^{(l)} - 1)|y_j|_{min}^{(l)}. \tag{5.5}$$

where S_l is the set of check-sums orthogonal on bit-position l.

Now we observe that E_l is simply a weighted check sum that is orthogonal on the code bit position l and can be used in the bit-flipping algorithm.

Message Passing Algorithm

Before explaining the decoding algorithm in Table 5.1, we define:

- C_i the set of check nodes connected to the variable node i;
- $C_{i\backslash j}$ the set of check nodes connected to the variable node i excluded the check node j;
- V_j the set of variable nodes connected to the check node j;
- $V_{j\backslash i}$ the set of variable nodes connected to the check node j excluded the variable node i.

- **Initialization**

$$L(q_{ji}) = L(c_i) \qquad j = 1, 2, \ldots, m;\ i = 1, 2, \ldots, n \tag{5.6}$$

- **Horizontal step**

$$L(r_{ji}) = 2\tanh^{-1}\Big(\prod_{i' \in V_{j\setminus i}} \tanh\Big(\frac{1}{2}L(q_{ji'})\Big)\Big) \tag{5.7}$$

- **Vertical step**

$$L(q_{ji}) = L(c_i) + \sum_{j' \in C_{i\setminus j}} L(r_{j'i}) \tag{5.8}$$

In this step we also compute $L(Q_i)$, the quantity corresponding to the pseudo-a-posteriori probability:

$$L(Q_i) = L(c_i) + \sum_{j' \in C_i} L(r_{j'i}) \tag{5.9}$$

Table 5.1: Description of the message passing algorithm.

5.1.2 Proposed algorithm

As said in the Introduction we ignore the additions and the subtractions in this analysis of complexity of MPA. Observing the algorithm description in Table 5.1, we find that the number of multiplications needed in each iteration of the decoding process is of the order $O(2m\bar{\rho} + 4n\bar{\gamma})$ where $\bar{\rho}$ and $\bar{\gamma}$ represent the mean number of 1's per rows and columns respectively and the number of logarithm operations needed is of the order $O(n)$ [90]. From these values and observing the evolution of the algorithm we had the idea for a modification of the algorithm which should drive to a reduction of the number of multiplications required.

MPA variation

Let's try to analyze the algorithm step by step to find all the possibilities to reduce the algorithm complexity.

Starting from the initialization step defined in equation (5.6), the $L(q_{j,i})$'s equal the LLRs defined as:

$$L(c_i) = \log\Big(\frac{P[c_i = 0|\text{channel output for } c_i]}{P[c_i = 1|\text{channel output for } c_i]}\Big) \qquad i = 1, 2, \ldots, N, \tag{5.10}$$

so if some absolute value of these quantities are greater than a given threshold we can say that the corresponding bits are reliable and change their LLR values to $\pm\infty$ according with the known signs.

When we compute the horizontal step as defined in equation (5.7), we observe that

for every $L(q_{j,i'})$ with $\pm\infty$ value the $tanh(\cdot)$ is equal to ± 1 and so all what we need is to keep trace of the number of -1, because this can change the sign of the product result, but we don't need to compute the multiplications.

Note that in the vertical step (equation (5.8)) we sum the contributions of $L(r_{ji})$'s. In case two or more values of $L(r_{ji})$ are equal to $\pm\infty$ we could sum terms infinitely large with opposite sign to obtain an undetermined result. In order to avoid this case we put the constraint that, in each $V_{j\setminus i}$ (equation (5.7)), at least one variable node is different from $\pm\infty$, which means that attention must be paid in the choice of the threshold, because its value can't be too small. Adding this constraint means that the $L(r_{ji})$ can assume $\pm\infty$ value when, with high probability, the bit is enough reliable and so it's hard to obtain undetermined results. Unfortunately, this is not enough to handle all the NaN cases. In the update of $L(q_{j,i})$ (equation (5.8)) we don't need to compute the update for the values fixed at the former steps, because we expect they don't change. However, in this case there are not multiplications, so at this step we don't save operations of the type we are focusing on.

The most important topic of our modified algorithm is the determination of the threshold: if it is chosen too small we fix some values that are not enough reliable and, as a consequence, in equation (5.8), we try to sum $+\infty$ and $-\infty$, without reaching a result. On the other side, we don't want to use a threshold too high, because this doesn't allow us to improve the efficiency of the algorithm.

5.1.3 System description

The system we consider here is represented in Figure 5.1. The source bits are encoded through an LDPC code of rate 1/2 and codeword length $n = 576$, then they are modulated through a QPSK modulator and transmitted on an AWGN channel. At the receiver we have a soft demodulator whose log-likelihood ratios are the input to the decoder. Obviously, the decoding is performed with our modified MPA and to have some terms of comparison we perform the decoding also through the standard MPA and the weighted BF algorithm [96].

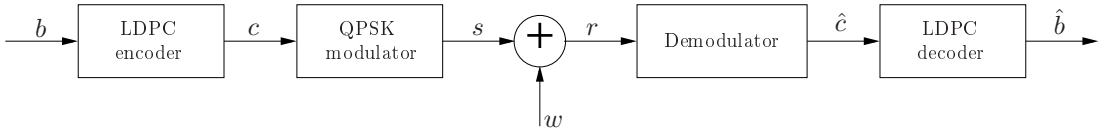

Figure 5.1: Representation of the considered transmission scheme, where the LDPC decoder can be implemented with the standard or modified MPA or with the WBF.

Since we want to have reliable performance results, we consider the concept of confidence, i.e. we have a probability equal to 95% that the measured value lies in the computed confidence interval [97].

We suppose that our system transmit 100 words of 288 information bits every time. These bits are encoded, modulated and transmitted on the channel. At the receiver side, the received signal (i.e. the signal at the output of the channel) is demodulated,

decoded and the bit error rate (BER) is computed. At this point we compute the confidence interval and verify if it's equal or smaller than the 15% of the estimated BER value; if it's not we run the simulation again for other 100 words and recomputed BER and confidence interval. The process is iterated until the constraint on the confidence interval is satisfied or a maximum number of iterations, equal to $100\dot{0}00$, is reached.

When we consider our modified MPA we need to handle the not-a-number (NaN) cases due to the attempt of summing together some $+\infty$ and $-\infty$. This can happen when we use a threshold which is too low and so we consider reliable some bits which aren't. What we do in these situations is considering a failure of the decoder on the word with the NaN bit and compute the BER on the other words.

5.1.4 Simulation results

To test our algorithm we consider the LDPC code proposed for the WiMAX system (IEEE 802.16e [1], [2]) with a codeword of length $n = 576$.

Figure 5.2 represents the performance in terms of bit error rate for the above LDPC code when it's decoded through different algorithms, namely the standard and our modified MPA, with different thresholds, and the WBF algorithm. To have other terms of comparison, the BER for the hard decision and for a '3G' turbo code is plotted. The reason why we choose the '3G' turbo code and not the one defined for the WiMAX system is because we want to compare the performance of our algorithm with the ones of a general turbo code, i.e. a turbo code composed by two identical convolutional code and an interleaver, and in the standard 802.16e there are no codes of this type. Another term of comparison is given by the WBF decoding algorithm, represented through the solid line with $\circ$ marker. As we can see from Figure 5.2, the performance of this algorithm is not so good, but this can be explained considering that it is proposed for decoding finite geometry LDPC codes [96] and the code we are considering is not of this type.

In Figure 5.2 the solid line with $\square$ represents the hard decision performance, the solid line with $\diamond$ represents the performance of the standard MPA algorithm and the solid line with $\star$ represents the turbo code performance. The dotted lines with $\circ$, $\square$, $\diamond$ and $\star$ represent the performance of our modified algorithm for different threshold values, which are defined by the maximum value of the received LLRs (max) or through the equation:

$$th = \min_{llr} + \frac{p}{100}(\max_{llr} - \min_{llr}). \tag{5.11}$$

where $p = 40$, 50, 90, the values reported in Figures 5.2 and 5.3.

Finally, in Figure 5.2 we can see a solid line with hexagon marker which represents the performance we can obtain choosing the best threshold at each SNR value. In this Figure, only the thresholds that lead to a significant gain on the performance of the standard algorithm are represented, but, obviously, we simulated more thresholds, to be sure that our choice is the best one.

To reach the best performance we can use our modified algorithm with different threshold until 4.5 dB and then we must use the standard algorithm.

What we want to underline is that all the results represented in this Figure are computed on the bit error rate of the words decoded through our modified algorithm; this means that we are not handling the not-a-number (NaN) cases. The same policy is adopted for the computation of the interval of confidence.

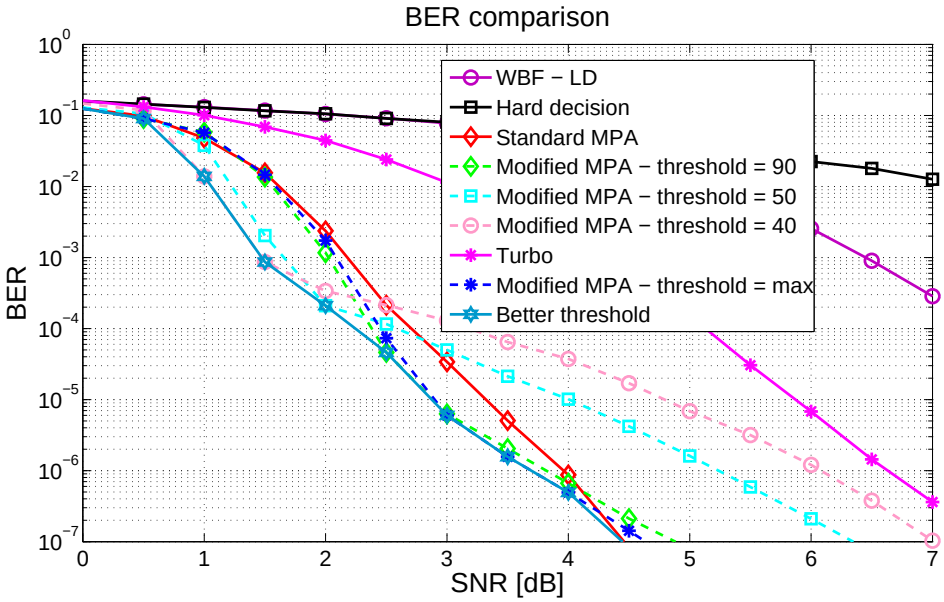

Figure 5.2: Bit error rate comparison when an LDPC code is decoded through different algorithms: MPA standard and modified, and WBF. Comparison of the performance of the LDPC code with a turbo code of the same rate.

Figure 5.3 represents the percentage of saved products and decoded words when the modified MPA algorithm is used with the four different thresholds. Figure 5.3.a) shows an expected result: the percentage of saved products is higher when the threshold is lower and, for example, at SNR= 2 dB and for $p = 90$ proposed algorithm saves the 35% of multiplication with respect to the standard MPA. What we can conclude is that, if we know the type of application we are using and so the reliability we need, we can reach a trade-off between the computational complexity and the system reliability, just choosing the correct threshold.

From Figure 5.3.b) we can observe that the more products we save the less words we can decode and so what is important, at this point, is to study a way to handle the NaN. Looking the Figure, we observe also a strange behavior of these percentages: for all the thresholds there's a minimum around $1 - 1.5$ dB. To understand this strange behavior will be one of the first topics of our future research.

Our results show that, choosing a good threshold allows a reduction of the BER, but this performance is computed only on the words the algorithm is able to decoding.

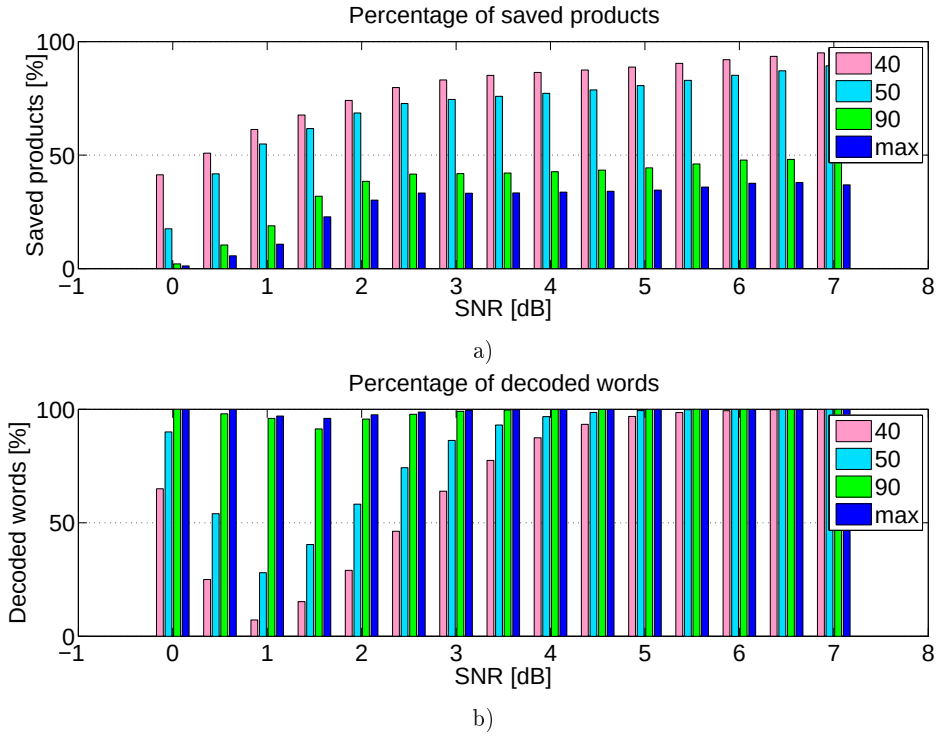

Figure 5.3: Percentage of a) saved products b) decoded words when we consider the modified MPA with different threshold.

5.2 Message passing algorithm: quantized version

In the standard LDPC decoder the input is a real vector which represents the log-likelihood ratios for the bits of the codeword.

5.2.1 Representation of real numbers in fixed-point data type

In digital hardware, numbers are stored in binary words. A binary word is a fixed-length sequence of bits (1's and 0's). How hardware components or software functions interpret this sequence of 1's and 0's is defined by the data type.

Binary numbers are represented as either fixed-point or floating-point data types.

A fixed-point data type is characterized by the word length in bits, the position of the binary point, and whether it is signed or unsigned. The position of the binary point is the means by which fixed-point values are scaled and interpreted.

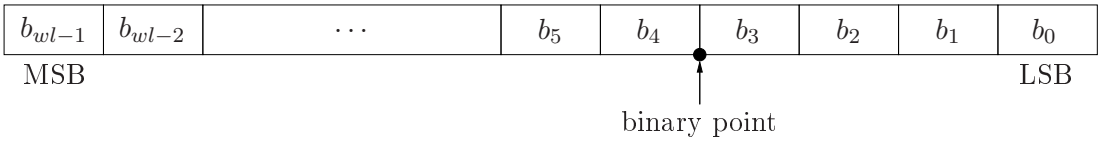

Figure 5.4: Binary representation of a generalized fixed-point number (either signed or unsigned).

In Figure 5.4 we see a binary representation of a generalized fixed-point number

(either signed or unsigned), where

- b_i is the ith binary digit
- wl is the word length in bits
- b_{wl-1} is the most significant bit (MSB)
- b_0 is the less significant bit (LSB)
- The binary point is shown four places to the left of the LSB. In this example, therefore, the number is said to have four fractional bits, or a fraction length of four.

Fixed-point data types can be either signed or unsigned. Signed binary fixed-point numbers are typically represented in one of three ways:

- Sign/magnitude
- One's complement
- Two's complement

5.2.2 Implementation of a function with fixed-point data type

When we implement a function with fixed-point data type we suppose that:

1. input and output are represented with n bits
2. internal states are represented with $2n$ bits.

The second assumption comes from the observation that the sums have still n bits, but this is not true for the products, so we use words of $2n$ bits to have a small loss in the precision of the fixed-point representation. These results are then re-quantized at the output.

For working with fixed-point data type, step by step, you should proceed as described following:

- Fix the number n of bits that you want to use
- Fix a range of representable values and so determine the precision of your system, i.e. fraction length.

  Example

  You want to represent the numbers in the intervals $[-128, 127]$ with $n = 16$ bits. Since $128 = 2^7$, you need 7 bits for the magnitude of the entire part and 1 bit for the sign, then, considering all, you need 8 bits for the entire part and you have $n - 8 = 8$ fractional bits. This means that the interval $[0, 1)$ is divided into 2^8 parts and the precision of your representation is 2^{-8}.

- Now you are using $n = 16$ bits integers in the range $[-2^{15}, 2^{15} - 1]$ to represent your values. This is a numerable and finite set, so a single map between these integers and the real values exists. The scale factor that you must use is 2^8, where the exponent is the fraction length.

5.2.3 Implementation of the soft decoding algorithm using fixed-point data type

We consider input and output at 16 bits and we suppose that the values of the log-likelihood ratios are in $[-64, 64)$. Since $64 = 2^6$ we need 6 bit for the magnitude plus 1 for the sign, for the entire part. Then ,the fraction length is of 9 bits; this means that we have a precision of $2^{-9} \simeq 2 \cdot 10^{-3}$.

Figures 5.5 and 5.6 represent the performance, in terms of codeword error probability and bit error probability, respectively, when we consider a LDPC code of rate $R = 1/2$ and codeword length $N = 576$. The decoding algorithm used is the same, but the two curves are obtained considering two different types of input to the decoder:

- the real-valued log-likelihood ratio;
- a 16 bit-quantized version of the same LLR.

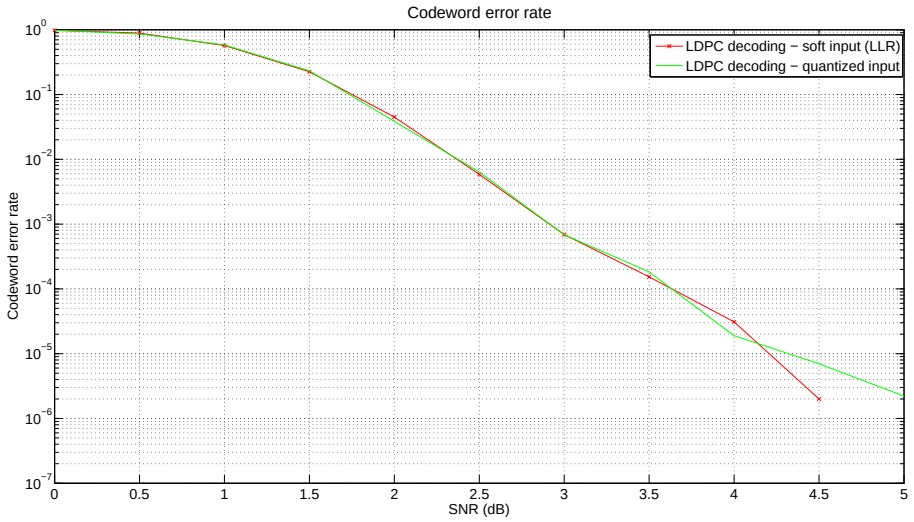

Figure 5.5: Considering a LDPC code of rate 1/2 and codeword length 576, we compare the performance, in terms of codeword error rate, of the message passing algorithm considering its input is either a real-valued log-likelihood ratio (LLR) or a quantized version of it.

From these Figures we can see that the performance are almost the same, so we can try to use a small number of bits for our quantization. So, now we need to determine the number of bit that we must use in our quantization and the fraction length. To determine these quantities we need to know which is the small value in the log-likelihood that can be considered reliable in the proceeding of the message passing algorithm.

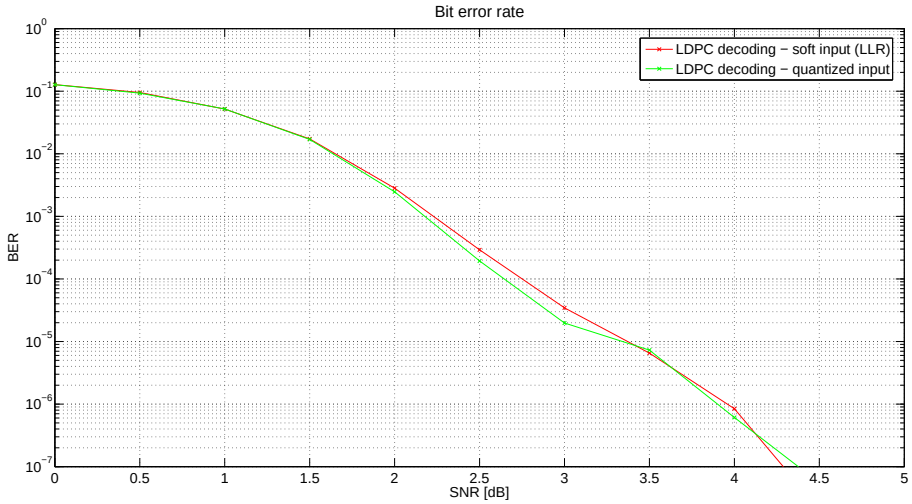

Figure 5.6: Considering a LDPC code of rate 1/2 and codeword length 576, we compare the performance, in terms of bit error rate, of the message passing algorithm considering its input is either a real-valued log-likelihood ratio (LLR) or a quantized version of it.

Finally, Figure 5.7 represents the performance, in terms of bit error rate, of the message passing algorithm when the input is quantized. We suppose to use 4 bits for the integer part and the total number of bits is 8 (blue line), 12 (magenta line), 16 (green line) and 32 (cyan line).The red line, instead represents the performance when the input is real valued.

We can observe that the best performance are obtained with 12 bits, so it is important to determine the number of bits we want to use and the fraction of it that are used to represent the integer part.

5.3 Other decoding algorithm for LDPC codes

As it is well known, computation complexity of message passing algorithm is high, so, in the following sections, alternative algorithms are analyzed especially from a computation point of view.

5.3.1 Priority first search algorithm

This algorithm was designed for decoding convolutional codes and more in general all the codes which can be represented with a trellis. Since we can design a trellis for a linear block code starting from its parity-check matrix, the algorithm can be used also for this kind of codes. Considering, then, that low-density parity-check codes are linear block codes, we have thought to use it instead of the message passing algorithm.

Aim of this algorithm is the transformation of the decoding problem into a search problem. For the linear block codes the algorithm proposed as solution for the decoding problem is the *maximum-likelihood soft-decision decoding*. By *maximum-likelihood*

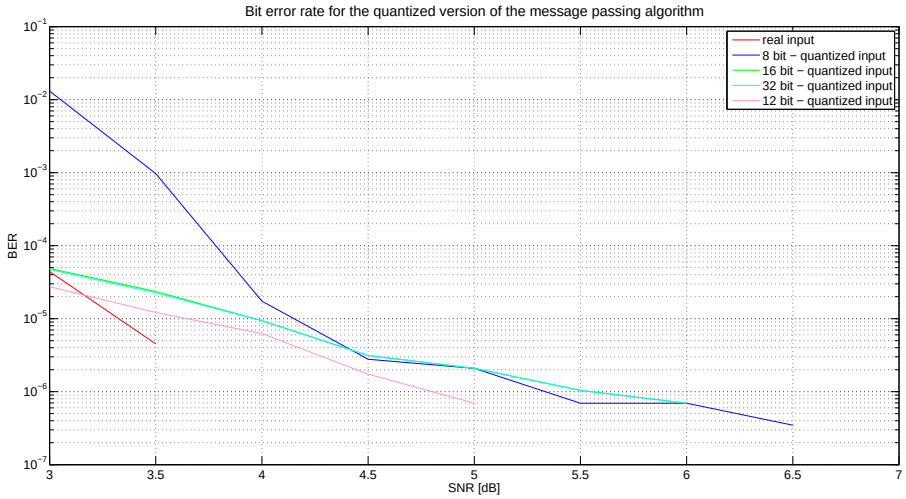

Figure 5.7: Considering a LDPC code of rate 1/2 *and codeword length* 576*, we compare the performance, in terms of bit error rate, of the message passing algorithm considering its input is either a real-valued log-likelihood ratio (LLR) or a quantized version of it. We suppose to use* 8*,* 12*,* 16 *and* 32 *bits and the number of bits used for the integer part is fixed to* 4*.*

decoding (MLD) we mean the minimization of the probability of decoding to an incorrect codeword when all codewords have equal probability of been transmitted. By *soft-decision* we mean the use of real number associated with every component of the codeword in the decoding procedure.

The idea presented for the first time in [6] is based on the Dijkstra's algorithm [98] to search for the trellis for a code equivalent to the transmitted code. The use of this priority-first search strategy for decoding drastically reduces the search space and results in an efficient optimal soft-decision decoding algorithm for linear block codes. Furthermore, the decoding efforts of our decoding algorithm are adaptable to the noise level.

In this algorithm, every branch in the tree graph is assigned a cost based on a maximum-likelihood decoding (MLD) rule. The purpose of the search is to find a desired path (codeword) which satisfied the MLD rule. This search is guided by an evaluation function f defined for every node in the graph. The algorithm maintains a stack of nodes of the graph that are candidates to be expanded. The node in the stack with minimum value of the function f is selected to expand. If the algorithm selects a goal node for the expansion it means that the desired path has been found.

The algorithm

The main advantage of the presented algorithm is given by the uncompleted visit of the nodes in the graph. In fact a successor, when applied to a node b gives all the immediate successors of the node and, for every immediate successor of node b, stores the partial path cost of the path ending at it. Then we assign the cost found to the

corresponding successor b_j and call it the cost (metric) of b_j, $g(b_j)$.

The process explained above is called *expanding* the node.

In PFSA the next node to be expanding is the one with the smallest cost on the list of all leaf nodes of the subtree constructed so far by the algorithm. If we suppose to memorize the nodes to be expanded in a stack ,we must keep it ordered according to the cost associated to each node. When the algorithm chooses to expand a goal node, it outputs the path associated with the cost of the goal node.

PFSA requires that for all nodes b_i and b_j such that node b_j is an immediate successor of node b_i, must be verified the equation

$$g(b_i) \leq g(b_j) \tag{5.12}$$

because it guarantees that the algorithm will always find an optimal path.

Owing the above requirement (equation (5.12)), the cost of any node is non-decreasing along any path in a code tree and is defined as follows:

$$g(b_j) = g(b_i) + (\phi - (-1)^c)^2 \tag{5.13}$$

where ϕ is the log-likelihood at the output of the demodulator for the symbol we are considering and $c \in \{0, 1\}$.

We observe that when a goal node is chosen for expansion, all costs of nodes in the stack are greater then or equal to the cost of the goal node and since all successor of any of these nodes will have costs no less than that of the node, one will not find any path with smaller cost than that of the goal node.

Considering a block code with a relative small codeword length, we can suppose the knowledge of the trellis. When we expand a node, we check if the state associated with each successor is in the trellis. If it belongs to the trellis, we insert the node into the stack, otherwise we don't consider it.

In figure 5.8 we can see the performance for a $(15, 11)$ Hamming code. The red line represents the performance when the decoding process is performed with the Hamming decoding process, the blue one represents the performance obtained with the PFSA described above, but where we suppose to know the trellis of the code and use this knowledge when we expand a node. In this case, when we find out the node with the minimum cost function, we control if it is in the trellis, if it is not we don't put it into the stack, because with it we don't find a codeword. Finally the green one represents the performance when we apply the PFSA only to the k information bits and then compute the parity-check bits and chose the codeword with the minimum distance from the received vector.

As we can see from Figure 5.8 the first variation of the PFSA, the one represented with the blue line, which is based on the knowledge of the trellis, outperforms the performance of the standard decoding algorithm. The problem arises when we consider a code with longer codewords, because the trellis grows enormous and we need too much memory. Unfortunately a LDPC trellis is very big and it is impossible to use the PFSA

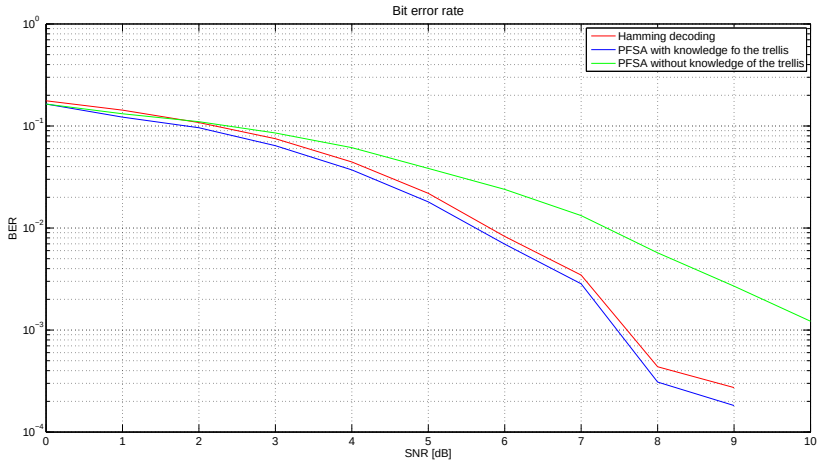

Figure 5.8: Considering a $(11, 15)$ *Hamming, we compare the performance, in terms of bit error rate, of the standard decoding algorithm with the one of two variation of the PFSA algorithm.*

with the trellis knowledge to decode a LDPC code.

Figure 5.9 represents the performance of the PFSA applied only to the k information bits and then compute the parity-check bits and choose the codeword with the minimum distance from the received vector. Blue and green lines represent the performance for this algorithm considering the acknowledge and the knowledge of the trellis, respectively. The red line, instead, represents the performance of the standard decoding algorithm for the Hamming codes.

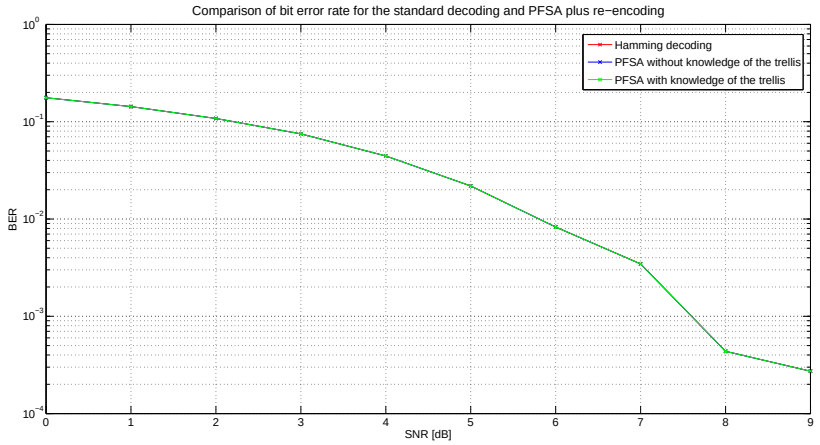

Figure 5.9: Considering a $(11, 15)$ *Hamming, we compare the performance, in terms of bit error rate, of the standard decoding algorithm with the one of two variation of the PFSA algorithm.*

As we can see from Figure 5.9 the performance is the same for the tree algorithms. As said above we can't assume the knowledge of the trellis for a LDPC codes and

from the observation of Figure 5.9, we had the idea of using the PFSA on the k bits without the knowledge of the trellis. The problem is that for a LDPC code we expand a lot of nodes and so the time needed to decode a codeword is very long. Thus, the MPA is better than the PFSA for LDPC decoding.

Just because the performance of the MPA is better than the one of the PFSA for an AWGN channel, we decided to not perform the simulation with a frequency selective channel, such as SCME, because the MPA will certainly perform better.

5.3.2 The genetic algorithm (GA)

Genetic algorithms (GA) were introduced by Goldberg in 1989. They are search algorithms that apply operations from natural genetics to guide the trek through a search space. GAs has theoretically and empirically proves to provide robust search capability in complex spaces, offering a valid approach to a problem requiring efficient and effective search.

GAs are general purpose search algorithms whose principles lie on natural genetics and can be applied to solve problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly non linear.

A genetic algorithm maintains a population of individuals that evolve according to rules of selection and genetic operators, such as reproduction, crossover and mutation. GA begins with a population that consists in randomly created individuals (possible solutions) and rapidly modifies this population "evolving" towards an optimal solution.

Each individual in the population is assigned a measure of its fitness in the environment. Reproduction focuses its attention on high fitness individuals, thus exploiting the available fitness information. Crossover and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist's viewpoint, these algorithms are complex enough to provide robust (good performance across a variety of problem types) and powerful adaptive search mechanisms. The adaptive behavior of the GA depends on this feedback to drive the population towards better overall performance [99] [100].

Therefore, considering a particular problem, an ad-hoc evaluation or fitness function must be devised.

As already known, GAs' performance is a function of parameter settings [101] [102]. The number of possible parameter assignments rules out a factorial design to fix the best parameter setting.

The algorithm for low-density parity-check codes

In [10] the authors presented an algorithm for the decoding of low-density parity-check codes based on GA. They suppose to use an additive white Gaussian noise (AWGN) channel, and called $\boldsymbol{x}$ the transmitted vector and $\boldsymbol{y}$ the received one, which is nothing more than the transmitted one corrupted by AWGN.

The algorithm can be implemented in three steps:

- syndrome calculation step

- GA application step

- meta decision step.

Syndrome calculation step

In this first step the proposed algorithm constructs a modified received vector $\boldsymbol{y}_{hard}$, which is basically a hard decision vector of the received vector y. In this step, the components y_i of the received vector, essentially real numbers, are converted into binary values (taken from the discrete alphabet 0,1) using a fixed threshold. Then, the decoding algorithm verifies if this modified vector satisfies the syndrome condition

$$\boldsymbol{H}\boldsymbol{y}_{hard} = \boldsymbol{0}. \tag{5.14}$$

If the modified received vector $\boldsymbol{y}_{hard}$ meets this condition, then a valid code vector $\boldsymbol{d}$ is obtained: $\boldsymbol{d} = \boldsymbol{y}_{hard}$. Otherwise, the decoder makes the following two steps.

Genetic algorithm step

The algorithm begins creating an initial population of V candidates: a set of individual vector v with real components $v_i \in [0, 1]$. To avoid an a priori reduction of the searching space, an initial random population is generated. A new 500 individual generation (children) is created through the following steps:

- Selects individuals (called parents) based on their fitness value 5.17 through the selection function.

- The two individuals with the best fitness values survive for the next generation (elite children= 2).

- The crossover fraction ($P_c = 0.95$) specifies the fraction of the population, other than elite children, that are made up of crossover children.

- To complete the new generation, mutation children are created by introducing random changes with a given probability rate ($P_m = 0.01$) to a single parent. The algorithm stops when the limit of 25 generations is reached.

The GA parameters were heuristically selected to optimize its performance.

The solution provided by the GA algorithm is a $\boldsymbol{z}$ vector obtained as follows:

$$z_i = \begin{cases} 1 & \text{if } \tilde{y}_i > v_i \\ 0 & \text{if } \tilde{y}_i \leq v_i \end{cases} \tag{5.15}$$

where

$$\tilde{y}_i = \frac{1}{1 + e^{-2y_i}} \tag{5.16}$$

expression that maps the components of the received vector $\boldsymbol{y}$ into [0, 1]. Hence the received vector $\boldsymbol{y}$ format agrees with the candidate vectors $\boldsymbol{v}$ format.

The process involves the following fitness function:

$$Fitness = \sum_{j=1}^{m} b_j + \sum_{j=1}^{n} |z_i - \tilde{y}_i| \tag{5.17}$$

where b_j are the components of the vector $\boldsymbol{b}$ defined as

$$\boldsymbol{Hz} = \boldsymbol{b}. \tag{5.18}$$

In equation (5.17), m is the number of rows of the parity check matrix $\boldsymbol{H}$, i.e. the number of parity-check nodes, and n the code vector length.

The fitness function (equation (5.17)) measures both a component wise distance between the candidate vector and the received one, and also how close the candidate vector satisfies the syndrome condition (equation (5.14)).

A set of q decoded vectors $\boldsymbol{z}$ is obtained applying GA algorithm q times, where q is an arbitrary integer value heuristically optimized (partial solutions). These q vectors are candidates for the following step of the decoding process, which consists in applying the meta-decision process.

Meta decision step

The Meta-decision process reduces the results scattering of the GA, which comes from the randomness of the initial population.

The $\boldsymbol{z}$ vectors are a set of possible solutions obtained at q GA runs, next a meta-decision stage generates the final solution, i.e, a decoded vector $\boldsymbol{d}$.

This process applies the majority logic, a well-known procedure utilized in the error correction decoding theory. This procedure performs a component wise decision over the z candidate vectors, setting each final component d_i as the bit state of higher frequency.

Still open point and conclusions

The results presented in [10] are quite promising: in their example, as we can see in [10], the GA based decoder performance is better than the traditional sum-product decoder for signal-to-noise ratio (SNR) greater than 3 dB.

We also observe that the main advantage of the proposed GAMD decoder is that noise level transmission channel information needs not be known, an essential condition for sum product algorithm.

Neverless it is important to understand how the authors compute the crossover and the mutation and how they determine

- the number of elements in the population,
- the number of iteration at the GA step,
- the number of decoded vectors used in the meta decision step,
- the crossover fraction P_c,
- the probability rate of changes P_m.

It is of the same importance to understand how to determine the above parameters when we consider different codes, especially with growing sizes.

Since the results presented in [10] were quite promising, we ran the genetic algorithm, with those parameters, on our quasi-compliant WiMAX standard simulator, but, unfortunately, our results were a complete disaster. We try to define different values for the parameters, but we couldn't find any theoretical rule to determine as good parameters for our system as those for the code in [10]. We try to define the parameters somehow, but we weren't lucky in our research: all the set of simulation performed, presented very bad results and this is the reason why, here, we don't present any kind of result.

Chapter 6

Concatenation of short LDPC and turbo codes for improved performance

It is well known that both Low-Density Parity-Check (LDPC) and Turbo codes are good codes if we consider long codewords, which means that the information part is greater than a thousand of bits. In our days, in many application, such as voice, delay is an important issue and large block size commonly used in LDPC or turbo codes cannot be applied. We propose the use of a concatenation of LDPC, as the outer code, and turbo, as the inner code, when we consider short frame, i.e. the information part of a codeword is around a hundred bits. The objective of the work is to come up with well-performing short code structures through the concatenation of LDPC and turbo codes: a synergy of short LDPC and short turbo codes vulnerable to different error patterns. The overall code rate is of concern: we consider 1/3, 5/16 and 1/4 rates. Other particular topics about this work are the choice of the decoding algorithms. For turbo codes the one with less computational complexity: the max-log-MAP algorithm is used and, for LDPC, the one used performs very well even if it has very high complexity: it is the message passing algorithm (MPA).

The work consists in analyzing the critical sequences of errors for turbo decoding and try to design the LDPC code with the purpose of decoding those kind of sequences. This would allow us to reduce the error floor presented by turbo codes at high SNR. Another important aspect is the determination of when a decoder should continue to iterate and when it would be better to forward the frame to the other decoder, which means when would the LDPC decoder have better luck decoding the frame than if the turbo decoder kept iterating.

6.1 Introduction

Both Low-Density Parity-Check (LDPC) and Turbo codes, when we consider long codewords, have near-capacity performance on a large variety of data transmission and

storage channels. This is the reason why they hold the attention of coding theorists and practitioners.

Everything about LDPC codes has been said in Chapters 4 and 5. Here we remember that they were invented by Gallager in 1960's [3] and [4], reinterpreted by Tanner in 1981 [103] and rediscovered by MacKay, Luby and others in the mid-1990's [104], [105], [106] and [5].

On the other side, Turbo codes were discovered by Berrou, Glavieux and Thitijashima [19] in 1993. In this work they demonstrated that these codes, with very large interleavers, can approach the Shannon bound on the capacity of a communication channel. Then, Berrou and Glavieux, further elaborated the concept in [20] and [21].

6.1.1 Concatenation: why and expected results

Serial concatenation of LDPC and turbo codes is not a new idea. It was first proposed in [22] where the authors observed that while turbo codes performs very well for BER's above 10^{-4} (waterfall performance), however they have a significant weakened performance at BER's below 10^{-5}: this phenomenon is known as error floor. Another important observation, we need to keep in our mind, is that, in turbo decoding, only information bits are decoded and they cannot be used for error detection. The poor minimum distance and lack of error detection capability make these codes perform badly in terms of block error probability. Poor block error performance also makes these codes not suitable for many communication applications. On the contrary, finite geometry LDPC codes do not have all the above disadvantages of turbo codes, except that they may not perform as well as turbo codes for BER's above 10^{-4}. Other reasons, which could justify the choice of a LDPC code as outer code are given by a very low complexity decoding, compared to the complexity if the MAP algorithm used for the turbo codes, and the fact that this algorithm can receive the reliable decoded bits directly from turbo decoders. Finally, LDPC has an inherent interleaving effect and so, if it's concatenated with another code, interleaver is not required [107]. Straight conclusion from the above observation is that the advantages of extremely good error performance of turbo codes for BER's above 10^{-4} and the advantages of finite geometry LDPC codes such as no error floor, possessing error detection capability after decoding and good block error performance, can be combined to form a coding system that performs well for all ranges of SNR's. Obviously, the first system of this type we can think about is the concatenation of an inner turbo code and a finite geometric outer LDPC code. In 2005 Lee and others [23], [24] analyzed the concatenation of LDPC and turbo codes, because they thought it would be a good idea to obtain a good trade-off between decoding complexity and performance. They observed that turbo code is the standard error correcting code for the third generation mobile communication systems, but even if it shows very good performance at low SNR's, its floor for higher SNR's is not as good as the high quality and high speed multimedia services required in the next generation mobile communication systems. Since LDPC code doesn't present the same floor for high SNR's, the concatenated scheme can be an alternative to solve the

problem of the new requirements. In [23] it is shown that the concatenated scheme does not suffer error floor and outperforms LDPC code of the same code rate when outer LDPC code rate is selected properly, hence the suitable LDPC code rate remains a still open point. In [24], the authors are still focused on the same problem, but since both the code presents iterative decoding they proposed to decode the concatenated scheme with a global iterative decoding process, which means that each iteration of the decoding algorithm performs turbo and LDPC decode. Results similar to the ones presented in [23] are shown. Finally, in 2006, the same authors in [25], observing that concentrated errors may occur in some codewords in the output of a turbo code, proposed the use of a block interleaver in the concatenation, between the two codes. Results show that the block interleaver improves the performance of the concatenation spreading out the concentrated errors at the output of the turbo decoder. What we can underline here is that all these works about the concatenation of LDPC and turbo codes consider long codewords, which is necessary to obtain good performance for both the considered codes. This is not too good if we must respect some delay constraints as in speech communications. This observation is the starting point of this work, where we want to analyze the performance of the concatenation scheme when the word we need to encode is a hundred bits. Since the codeword is relatively short, we analyze the output at the turbo decoding and try to construct an "ad hoc" LDPC decoder which is suitable for decoding what the turbo code cannot correct. This idea is useful also to avoid the block interleaver usage, which is not suitable, because it introduces more delay.

6.2 Error pattern at the output of the turbo code and LDPC construction

6.2.1 Turbo code: '3G 'type

The scheme of Turbo coder is a Parallel Concatenated Convolutional Code (PCCC) with two 8−state constituent encoders and one Turbo code internal interleaver. The coding rate of Turbo coder is 1/3. The structure of Turbo coder is illustrated in Figure 6.1.

The transfer function of the 8−state constituent code for PCCC is:

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)}\right], \tag{6.1}$$

where

$$g_0(D) = 1 + D^2 + D^3 \tag{6.2}$$

$$g_1(D) = 1 + D + D^3 \tag{6.3}$$

The initial value of the shift registers of the 8−state constituent encoders shall be

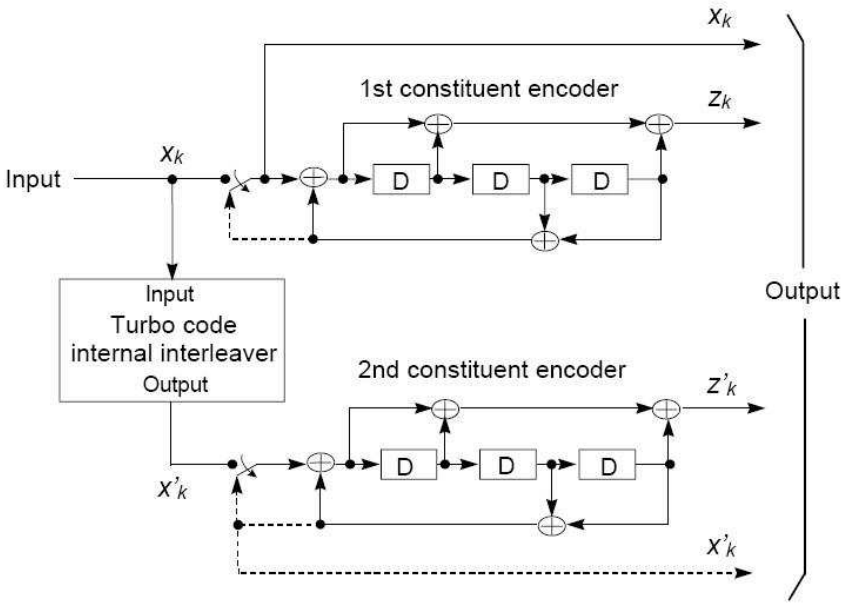

Figure 6.1: Structure of rate 1/3 Turbo coder.

all zeros when starting to encode the input bits. Output from the Turbo coder is:

$$x_1,\, z_1,\, z'_1,\, x_2,\, z_2,\, z'_2,\, \ldots,\, x_K, z_K, z'_K,$$

where $x_1,\, x_2,\, \ldots,$‘,x_K are the bits input to the Turbo coder i.e. both first 8−state constituent encoder and Turbo code internal interleaver, and K is the number of bits, and $z_1,\, z_2,\, \ldots.,\, z_K$ and $z'_1,\, z'_2,\, \ldots,\, z'_K$ are the bits output from first and second 8−state constituent encoders, respectively. The bits output from Turbo code internal interleaver are denoted by $x'_1,\, x'_2,\, \ldots,\, x'_K$, and these bits are to be input to the second 8−state constituent encoder.

Also the turbo interleaver is standard compliant [49].

For the decoding two algorithms are implemented: the log-MAP decoding and the MAX-log-MAP decoding, which is an approximation of the former. In this work we use the second one, since it is more efficient from a computational point of view.

Code's properties

We are looking for those double error pattern in the CC sequences which cannot be decoded through the CC decoding.

We observe that the polynomials in Equations (6.2) and (6.3) can be written in other two different forms (binary and octagonal) as follows:

$$g_0 = [1\,0\,1\,1] = 13$$

$$g_1 = [1\,1\,0\,1] = 15,$$

hence the encoder needs 3 memory elements and we can define the *constraint length* equal to 4.

The code is a *systematic feedback code* and the encoder can be represented as in Figure 6.2, where we represented the content of the shift register at the beginning of the coding.

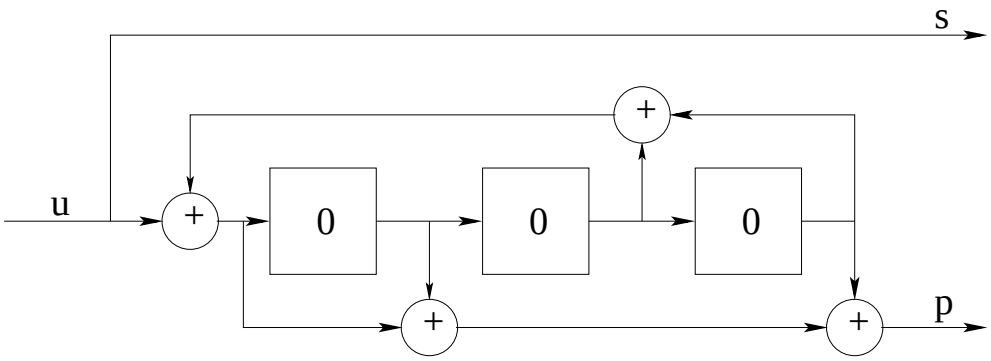

Figure 6.2: Representation of the encoder for the (4, [13 15], 13) *convolutional code.*

In Figure 6.2, u represents the information sequence, s the systematic output of the encoder and p the computed parity-check bits.

Figure 6.3 represents the trellis used for the decoding of the received codeword.

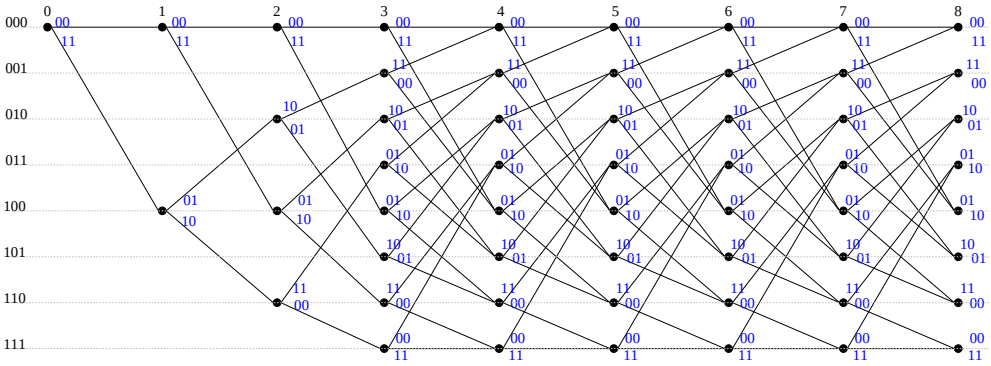

Figure 6.3: Representation of the trellis for the (4, [13 15], 13) *convolutional code.*

Considering Figure 6.2 and the input sequence $\boldsymbol{u} = (1\,0\,0\,0\,0\,0\,0\,1)$, we can show that the feedback polynomial is *primitive* [90]. Table 6.1 contains the state evolution and the output of the convolutional encoder.

We observe that the shift register's contents rotate through the all $2^3 - 1 = 7$ possible states before returning to the original all-zero state and if the last input bit would be 0 instead of 1 the new state would be $1\,0\,0$ and so another cycle is repeated. In Figure 6.4 we can see the path for this input sequence on the trellis.

Observation 8. *The behavior described above makes us think that the error pattern of our interest are the ones with this characteristic: considering the source bits, if we have*

u	state	s	p
	000		
1	100	1	1
0	010	0	1
0	101	0	1
0	110	0	1
0	111	0	0
0	011	0	0
0	001	0	1
1	000	1	1

Table 6.1: Encoder states and output, when the sequence $\boldsymbol{u} = (1\,0\,0\,0\,0\,0\,0\,1)$ *is encoded.*

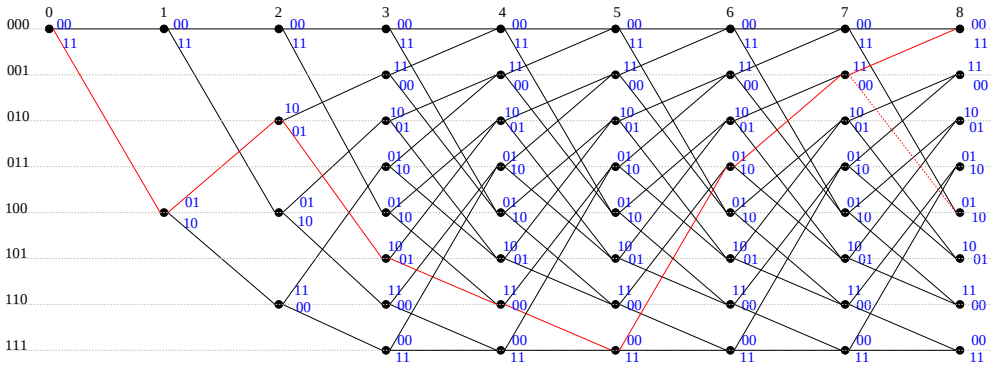

Figure 6.4: Representation of the path on the trellis for the coding of sequence $\boldsymbol{u} = (1\,0\,0\,0\,0\,0\,0\,1)$ *through the* $(4, [13\,15], 13)$ *convolutional code. The red path is the one of our interest and the dashed line represents the beginning of a new cycle if the last input is a* 0 *instead of a* 1.

an error at the instant k *we would not be able to decode the pattern if the second error would happen at the instant* $k + n \cdot 7$*, with* $n = 1, 2, \ldots$.

From Figure 6.2 we can see that the same sequence is encoded through a CC encoder twice, the first time directly, the second time after an interleaver operation.

At the receiver side (see Figure 6.5) the received systematic part is the input for both the max-log-MAP decoder and to the second after the interleaver operation. From the Observation 8 and the Figure 6.5 we understand that the sequence that are not corrected after the turbo decoders are those whose maintain the property defined in Observation 8 also after the interleaver operation.

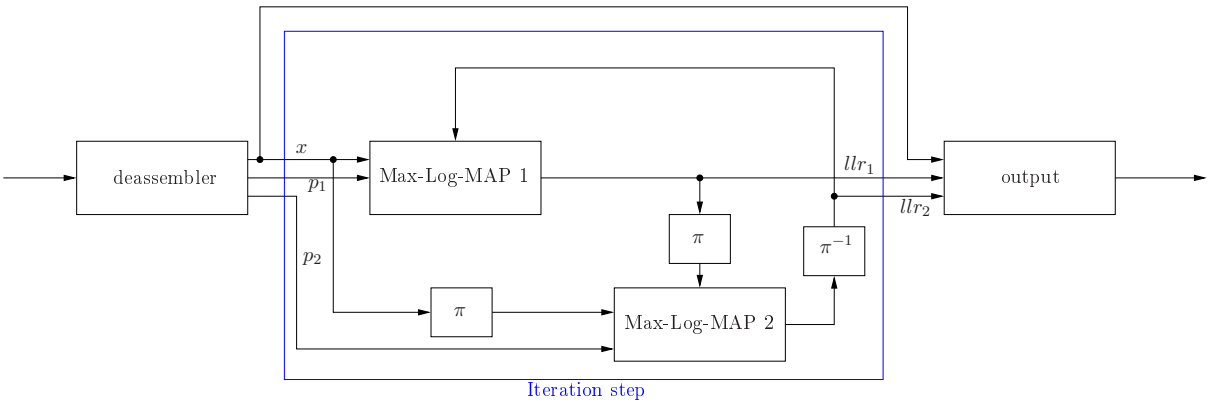

Figure 6.5: Representation of the turbo decoder. The block deassembler subdivides the received signal in three parts: the systematic bits, the parity-checks for the first encoder and the parity-checks for the second one; the π represents the interleaver operation and the π^{-1} the deinterleaver one. Finally the block output computes the final log-likelihood ratio.

We call *critical sequences* all the possible sequences with 2 errors of this kind, because they cannot be corrected after the interleaver, since the distance between the 2 errors is still equal to the minimum distance or a multiple of it.

The interesting results of this study is given by the fact that these sequences can be grouped, because different groups of sequences affect different and separate groups of bits at the output of the decoder. Figures 6.6 and 6.7 represent the critical sequences when we consider a concatenation with overall rate equal to $R = 1/3$, with turbo code's rate $R_T = 1/2$ and LDPC code's rate $R_L = 2/3$. Since the turbo code is the inner one, its sourceword length is equal to 150, because the words we want to encode have length 100 bits.

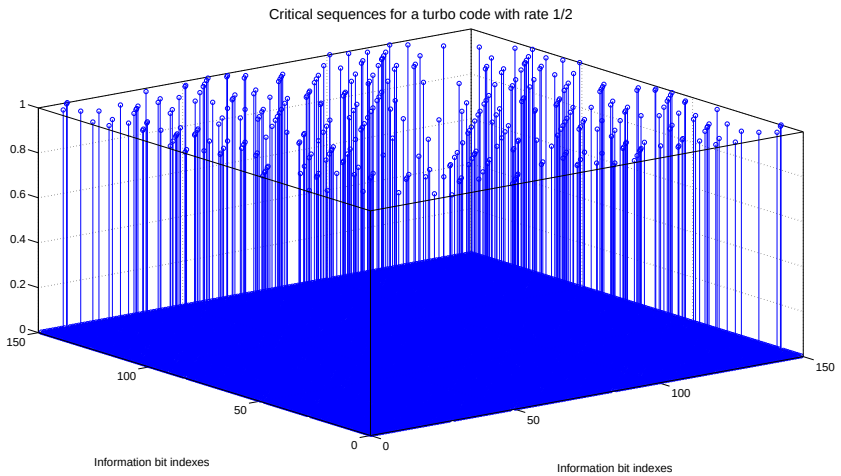

Figure 6.6: Representation of the critical sequences for a turbo code with rate 1/2 when the sourceword length is 150. This Figure refers to the results reported in Section 6.4.1

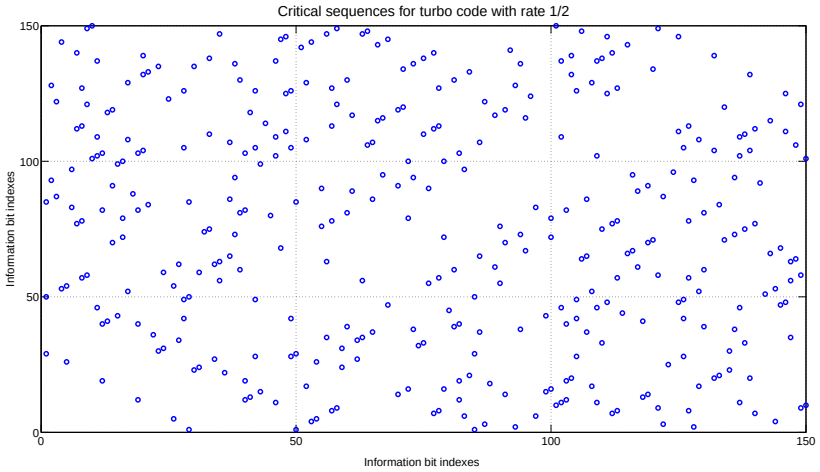

Figure 6.7: Two-dimensional representation of Figure 6.6.

This observation gave us an idea: when we construct the LDPC matrix for the outer code we add a constraint; we want that the columns correspondent to the bits of the same group with errors at the output of the turbo decoders have no ones in common. This means that those bits are checked by different check nodes and so the probability of correcting the errors through the LDPC code is higher.

6.2.2 LDPC code construction

We can choose if we want to use a regular or an irregular LDPC code and if we choose the second one we must specify a degree distribution. In literature it is shown that the irregular codes have better performance especially if the codewords are very long. In [108], the authors studied the probability densities at the message nodes of the graph and show that they always converge as the number of iterations tends to infinity. Using this result they found the optimal degree distribution when a codeword is of infinite length. However we consider short frame, so, we can find the optimal distribution for a code with the specified rate and infinite codeword length and then we can adapt the computed parameter to our case [90]. Unfortunately, this operation introduces cycles in the Tanner graph and so the performance experiences error floor for high SNR and bad results for low SNR because of a big number of degree−2 variable node.

To solve this problem the variable node distribution must be modified to have a number of variable nodes of degree−2 less than or equal to the number of parity-check nodes and when the graph is constructed the degree−2 variable nodes must be made correspond to the check nodes. It is important also that the degree−2 variable nodes are cycle-free. Once the variable node degree distribution is modified also the check node degree distribution must be changed, because it must be compatible with the other one.

On the web site `http://lthcwww.epfl.ch/research/ldpcopt/index.php` there is an implemented algorithm to compute the optimal degree distributions for infinite codeword and from `http://www.cs.toronto.edu/ radford/ftp/LDPC-2006-02-08/`

`index.html` we can download some c-functions to construct the parity-check matrix once we have the degree distributions.

After appropriate modifications, these functions are used to construct the parity-check matrix which needs to be modified to satisfy the new constraint.

6.3 System description

The system considered is represented in Figure 6.8.

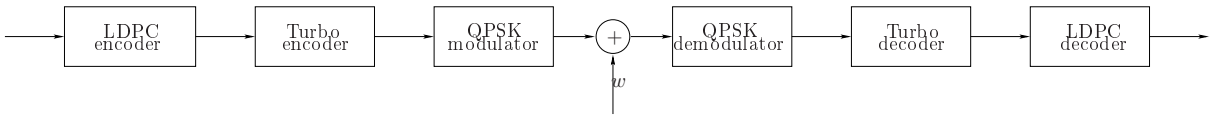

Figure 6.8: Representation of the considered transmission scheme.

Data are encoded first through a LDPC encoder and then through a turbo encoder. After the coding operation they are modulated through a QPSK modulator and sent over an AWGN channel. At the receiver side, they are first demodulated and then decoded through a turbo and a LDPC decoder.

Since we want to have reliable performance results, we consider the concept of confidence, i.e. we want a probability equal to 95% that the measured value lies in the computed confidence interval [97].

We suppose that our system transmits 100 words of 100 or 105 information bits every time. These bits are encoded, modulated and transmitted on the channel. At the received side, the received signal (i.e. the signal at the output of the channel) is demodulated, decoded and the bit error rate is computed. At this point we compute the confidential interval through the equation:

$$confidence = 2 \cdot 1.96 \cdot \sqrt{(var/total\_number\_words/info\_bits)}/BER, \qquad (6.4)$$

where $var = BER \cdot (1 - BER)$, $total\_number\_words$ is the total number of words simulated until that moment and $info\_bits$ is the information length of a codeword. This confidence interval must be equal or smaller than 15% of the estimated BER value; if it is not we run the simulation again for other 100 words and recomputed BER and confidence interval. The process is iterated until the constraint on the confidence interval is satisfied or a maximum number of iterations, equal to 10000, is reached.

6.4 Results

The results shown in this section are computed through Matlab simulations: functions are implemented in Matlab and C, where the c-functions are transformed in mex-functions and so they can be used into Matlab simulations.

6.4.1 Overall rate 1/3

The turbo rate is $R_T = 1/2$ and it is obtained through the puncturing in the '3G' standard [49]. The LDPC code has rate $R_L = 2/3$ and we compared the performance with different matrices.

Figure 6.9 represents the performance of the concatenation when we use regular and irregular LDPC matrices obtained with the algorithms described in Section 6.2.2 (curves red, cyan and green). The blue and yellow curves represent the performance when the matrices used are modified to satisfy the new constraint. We can observe that for growing SNR the performance is better and this is reasonable, because, with this growing, the errors are due more to the distance of the code than to the channel noise.

Unfortunately, the concatenation doesn't present the expected results, because the LDPC code isn't able to recovery the loss do to the turbo puncturing.

Figure 6.9 represents also the performance for an LDPC code and a turbo code with rate 1/3. While the concatenation outperform the turbo for SNR higher than 3.5 dB, the used LDPC code performs very well and is of interest for another kind of study.

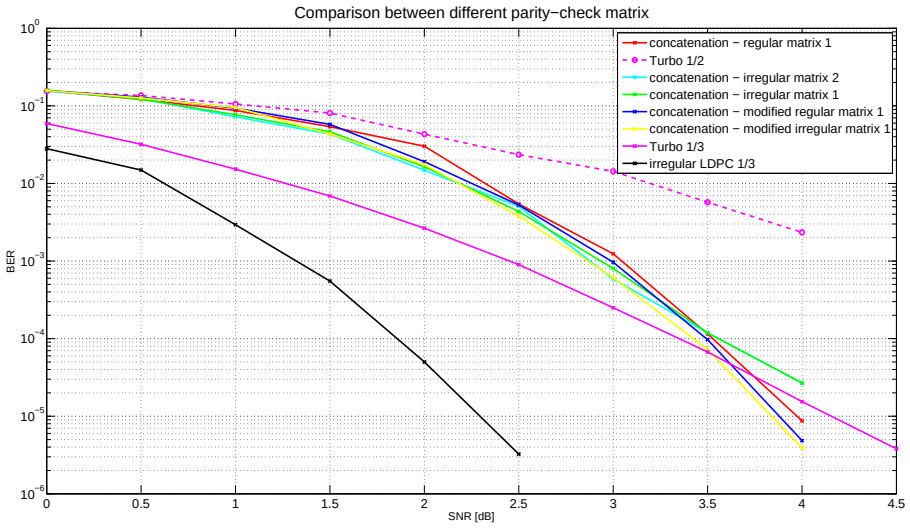

Figure 6.9: Concatenation performance with overall rate equal to 1/3.

6.4.2 Overall rate 5/16

The turbo rate is $R_T = 1/3$ and the LDPC code's one is $R_L = 15/16$. In this case the performance doesn't experience the loss due to the puncturing of the turbo code, but the problem is that the LDPC parity check matrix is very small (7×112) and there is not enough space to modify the matrix with the aim of satisfying the new constraint.

In Figure 6.10 magenta and black curves represent performance for turbo code with rate 1/3 and LDPC code with rate 5/16, respectively. On the other hand blue and green lines represent performance for the concatenation system when we use the random

matrix and its modified version. Here we can observe that the concatenation has some advantages in terms of bit error rate, but it is impossible to satisfy the new constraint and so the results are almost the same (see blue and green lines).

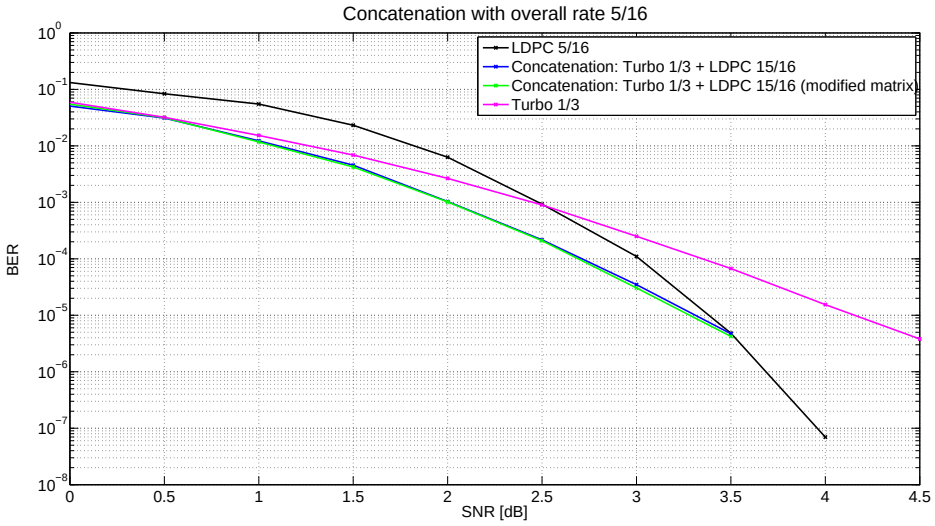

Figure 6.10: Concatenation performance with overall rate equal to 5/16.

6.4.3 Overall rate 1/4

The turbo rate is $R_T = 1/3$ and the LDPC one is $R_L = 3/4$. In this case the performance doesn't experience the loss due to the puncturing of the turbo code, but the problem is that the overall rate is very low and this is not good in our system, because we are trying to satisfy some delay constraints.

In Figure 6.11 magenta and cyan curves represent performance for turbo code with rate 1/3 and LDPC code with rate 1/4, respectively. On the other hand, blue and green lines represent performance for the concatenation system when we use the random matrix and its modified version. Here we can observe that the concatenation has some advantages in terms of bit error rate: for example, at BER equal to 10^{-6} the concatenation scheme gains 1.1 dB. From Figure 6.11 we observe also that the concatenation with the modified matrix (green line) presents an enhancement with respect to the non-modified one (blue line). This has a theoretical explanation: at low SNR most of the errors are due to the noise of the channel, but, as the SNR grows, the errors are due to the inability of decoding some sequences which can be corrected, with high probability, by the modified LDPC.

Unfortunately, the BER's values are very small, and, even if we ran the maximum number of iteration for every simulated point, we can't say those results are completely reliable.

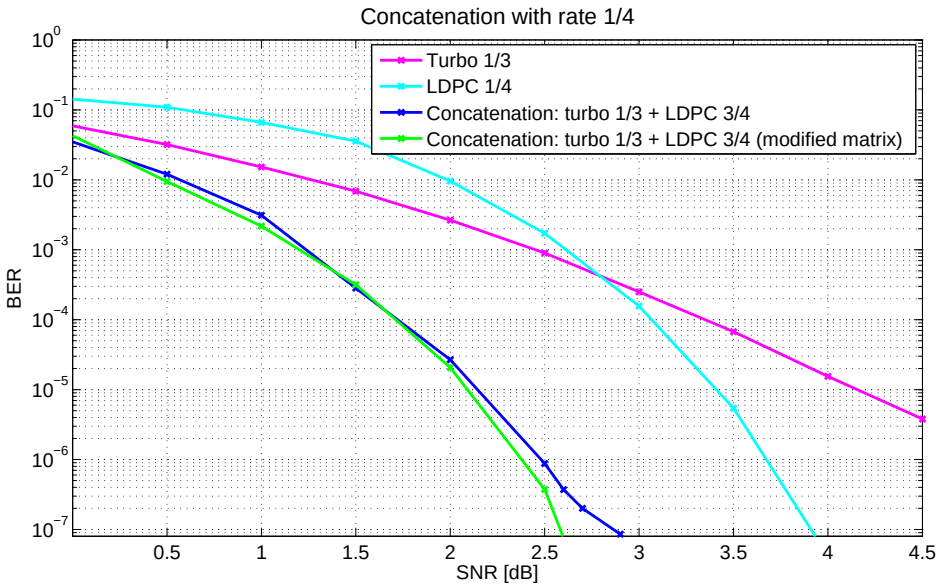

Figure 6.11: Concatenation performance with overall rate equal to 1/4.

6.4.4 Conclusion

Observing Figures 6.9, 6.10 and 6.11 we can conclude that concatenation presents performance gain in terms of bit error rate for codes with very low rates. This is far from our purpose: we presented the concatenation scheme for short frames to respect some delay constraints.

Observing LDPC performance in Figures 6.9 and 6.11, another idea comes to our mind: compare the BER for the LDPC 1/3 (Figure 6.9) and LDPC 1/4 (Figure 6.11). From a theoretical point of view we expect the LDPC 1/4 outperforms the LDPC 1/3, but actually it is not true, as we can see from Figure 6.12, where the black line represents per BER of the LDPC code with rate 1/3, the cyan one the performance of LDPC code with rate 1/4 and the green one the performance of the concatenation scheme with overall rate equal to 1/4. For example, at $2 \cdot 10^{-5}$ BER the LDPC 1/3 has a gain of 1.1 dB and this is due to the strong dependence of the performance from the parity-check matrix.

For both the LDPC codes, the parity-check matrices were chosen by using the same design criterion. The choice has been limited, for both codes, to a small number of parity-check matrices. The conclusion we tried is that for LDPC code with rate 1/3 we found a very good solution while for the LDPC code with rate 1/4 the best solution we found is far away from the acceptability and the search must be continued.

We can compare also the performance of the LDPC code with rate 1/3 and the concatenation with overall rate 1/4. In this case results confirm the expectation: at $2 \cdot 10^{-5}$ of BER the concatenation scheme gains 0.2 dB over LDPC code and at 10^{-7} 0.4 dB.

To conclude this section we observe also that all the parity check matrices have been

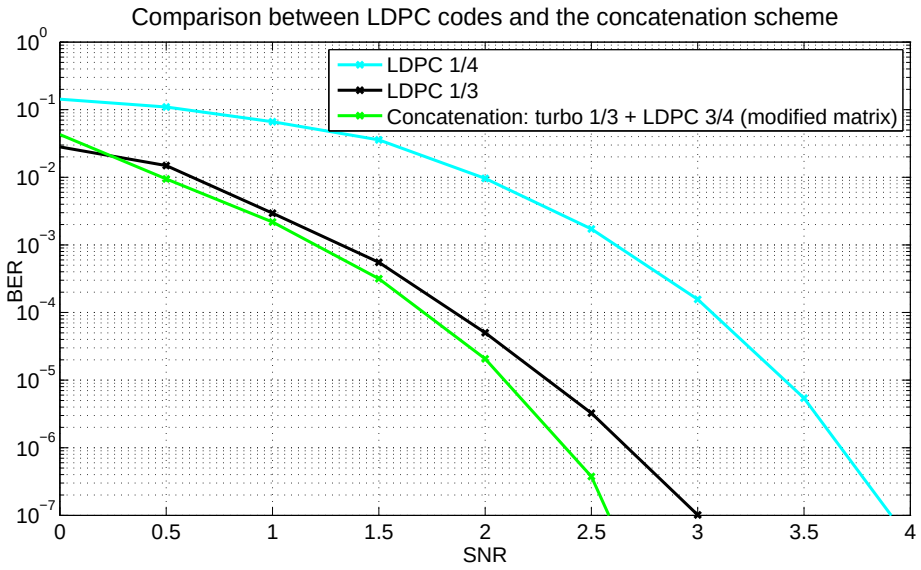

Figure 6.12: Comparison of LDPC codes with rate $1/3$ *and* $1/4$ *and concatenation scheme with rate* $1/4$

chosen with the design criterion described above, thus the remark made for LDPC code with rate $1/4$ can be made also for the LDPC code of rate $5/16$ and all the LDPC codes involved in the concatenation schemes.

Chapter 7

WiMAX standard

This Chapter has the aim of briefly describe those parts of the standard which have been of interest for the implementation of a downlink simulator standard compliant.

What I want to underline is that it doesn't give a complete description of the standard, so for more details and completeness I recommend the readers to reefer to [1] and [2].

7.1 Aim and frequency allocation

The WirelessMAN- OFDMA PHY, based on the OFDM modulation is designed for non-line-of-sight (NOS) operation in the frequency bands below 11 HZ, as seen from the table 7.1.

Designation	**Applicability**	**PHY**	**Options**	**Duplexing alternative**
WirelessMAN−SC^{TM}	$10-66$ HZ	8.1		TDD FDD
WirelessMAN−SCa^{TM}	Below 11 HZ licensed bands	8.2	AAS, ARQ, STC	TDD FDD
WirelessMAN −OFDM^{TM}	Below 11 HZ licensed bands	8.3	AAS, ARQ, Mesh, STC	TDD FDD
WirelessMAN−OFDMA	Below 11 HZ licensed bands	8.4		AAS, ARQ, STC TDD FDD
WirelessMAN −HUMAN^{TM}	Below 11 HZ license-exempt bands	[8.2, 8.3 or 8.4 and 8.5	AAS, ARQ, Mesh, STC	TDD

Table 7.1: Air interface nomenclature.

For licensed bands, channel bandwidth shall be limited to the regulatory provisioned

bandwidth divided by any power of two, and must be no less than 1 MHz.

The OFDMA PHY mode based on at least one of the FFT sizes 2048 (backward compatible to IEEE Std 802.16-2004), 1024, 512 and 128 shall be supported. This facilitates support of the various channel bandwidths.

The Mobile Station (MS), called also User Equipment (UE) or Subscriber Station (SS), may implement a scanning and search mechanism to detect the DL signal when performing initial network entry and this may include dynamic detection of the FFT size and the channel bandwidth employed by the Base Station (BS).

We have five frequency range that will support WiMAX, and to everyone of this we associate one or more combinations of channel bandwidth, FFT size, channel raster and duplexing mode. In [109], whose scope is to provide the OFDMA system profile specification of mobile network, primarily for the purpose of certification of conformant MS and BS, we find all the possible combinations of the above parameters. They are resumed in Table 7.2.

Frequency range (HZ)	**Channel frequency step (kHz)**	**Channel bandwidth (MHz)**	**FFT size**	**Duplexing mode**
$2.3 - 2.4$	250	5	512	TDD
		10	1024	TDD
		8.75	1024	TDD
$2.305 - 2.32$	250	5	512	TDD
$2.345 - 2.36$		10	1024	TDD
$2.496 - 2.69$	250 (200 in Europe)	5	512	TDD
		10	1024	TDD
$3.3 - 3.4$	250	5	512	TDD
		10	1024	TDD
$3.4 - 3.8$	250	5	512	TDD
		7	1024	TDD
		10	1024	TDD

Table 7.2: Combinations of channel bandwidth, FFT size, channel raster and duplexing mode.

As we can see from the Table 7.2, even if the standard allows the two duplexing modes TDD and FDD, the WiMAX Forum requires only the TDD mode to have a certified product.

7.2 OFDMA symbol description, symbol parameters and transmitted signal

7.2.1 Time domain description

The OFDMA waveform is created through the Inverse Discrete Fourier Transform (IDFT) and its time duration is named *useful symbol time* T_b. To collect multipath, while maintaining the orthogonality of the tones, a Cyclic Prefix (CP) is used, which consists in coping the last T_g seconds of the useful symbol period at the beginning of the

symbol. This method allows the receiver to take the samples for performing the Fast Fourier Transform (FFT) anywhere over the length of the extended symbol, T_s and so multipath immunity and tolerance for symbol time synchronization errors are achieved.

The CP values proposed in the standard are 1/4, 1/8, 1/16 and 1/32, but in [109] it is said that the value we must use is 1/8.

The transmission energy increases with the length of the CP, while the receiver energy remains the same, so there is a loss in E_b/N_0, that can be quantified through:

$$(P_{loss})_{dB} = \frac{10\log(1-\frac{T_g}{T_s})}{\log(10)}. \tag{7.1}$$

where T_g and T_s are defined in 7.2.3 When CP is equal to 1/8, in (7.1) we found:

$$(P_{loss})_{dB} = 0.5 \text{ dB}$$

On initialization, a MS should search all possible values of CP, until it finds the one being used by the BS. The SS must use the same CP on the uplink, because it can be changed by the BS, but this operation requires the resynchronization of all the SSs to the BS.

7.2.2 Frequency domain description

An OFDMA symbol is made up of subcarriers, whose number determines the used FFT size. There are three subcarrier types:

- **Data subcarriers**: for data transmission;

- **Pilot subcarriers**: for various estimation purposes;

- **Null carriers**: no transmission at all, for guard bands and DC carrier.

In the OFDMA mode, the active subcarriers (pilot and data subcarriers) are divided into subsets of subcarriers, which may, be adjacent, and each subset is termed *subchannel*. In downlink, one subchannel can be intended for groups of receivers; in uplink, instead, one subchannel may be assigned to more transmitters, so that SS may transmit simultaneously.

This technique supports scalability, multiple access and advanced antenna array processing capabilities.

7.2.3 Parameters

Primitive parameters

Four parameters characterize the OFDMA symbol:

1. **BW:** nominal channel bandwidth

2. $\mathbf{N}_{used}$**:** number of used subcarriers (included the DC subcarrier)

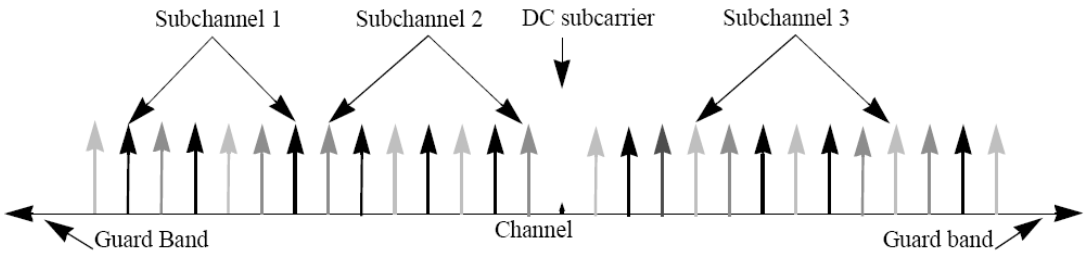

Figure 7.1: OFDMA frequency description: an example of three channel.

3. **n:** sampling factor; in conjunction with BW and N_{used}, determines the subcarrier spacing and the useful symbol time. Its values is 28/25 if the channel bandwidth is a multiple of any of 1.25, 1.5, 2 and 2.75 MHz and 8/7 in the other cases.

4. **G:** ratio of CP time to useful time ($G = T_g/T_b$)

Derived parameters

These parameters are defined in terms of the primitive ones.

1. $\mathbf{N}_{FFT}$**:** FFT size is the smallest power of two greater than N_{used}

2. $\mathbf{F_s}$**:** sampling frequency $F_s = \left\lfloor n \cdot \frac{BW}{8000} \right\rfloor \times 8000$

3. $\boldsymbol{\Delta f}$**:** subcarrier spacing $\Delta f = \frac{F_s}{N_{FFT}}$

4. $\mathbf{T_b}$**:** useful symbol time $T_b = \frac{1}{\Delta f}$

5. $\mathbf{T_g}$**:** CP time $T_g = G \cdot T_b$

6. $\mathbf{T_s}$**:** OFDMA symbol time $T_s = T_g + T_b$

7. $\mathbf{T_{sam}}$**:** sampling time $T_{sam} = \frac{T_b}{N_{FFT}}$

Channel BW	**Scaling factor**	$\mathbf{F_s}[dB]$
1.25	28/25	1.4
1.5	28/25	1.68
2	28/25	2.24
2.75	28/25	3.08
5	28/25	5.6
8.75	28/25	9.8
10	28/25	11.2
7	8/7	8

Table 7.3: Sampling frequency F_s.

7.3 Frame duration

In the standard [2] the frame length can be equal to 20, 12.5, 10, 8, 5, 4, 2.5 and 2 ms, but we saw in [109] that the length chosen is equal to 5 ms.

This length indicates the periodicity of the downlink frame start preamble in both TDD and FDD cases. Therefore we can say that in every second we transmit 200 frames.

On this point three observations are necessary:

- the frame duration is not an integer multiple of the OFDMA symbol duration. Therefore sometimes padding may be necessary between the last useful OFDMA symbol of the frame and the beginning of the next frame;
- in TDD case (which is the only one proposed in [109], as we can see from Table 7.2) Receive/transmit Transition Gap (RTG) and Transmit/receive Transition Gap (TTG) guard intervals must be included in the frame, and both of them shall be no less than 5 μs in duration;
- a simple scheme of the frame structure is shown in Figure 7.2, where we can see that it is composed by a downlink transmission period (which includes also the preamble), a TTG, an uplink transmission period and a RTG.

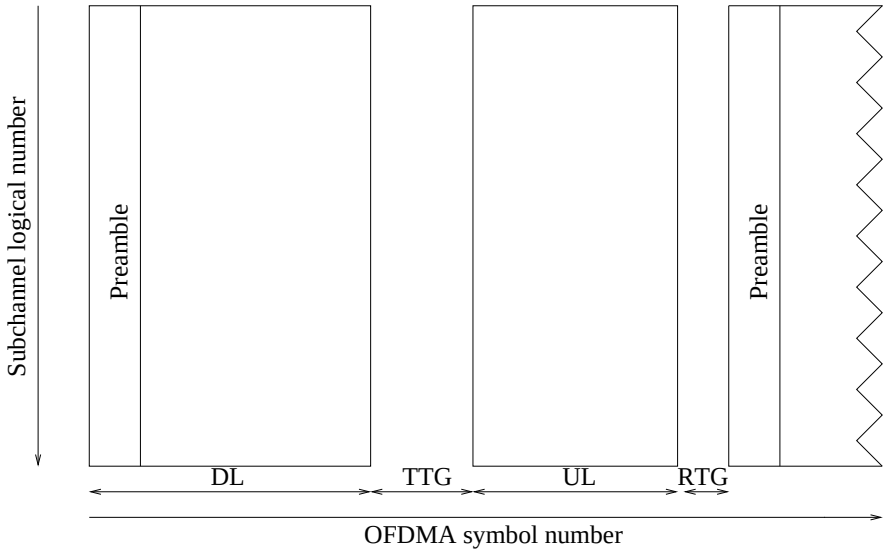

Figure 7.2: Simple scheme of an OFDMA frame in TDD mode.

Before characterizing RTG and TTG it is necessary to define the Physical Slot (PS) as a time unit, dependent on the PHY specification, for allocating bandwidth and in WirelessMAN-OFDMA it is given by:

$$PS = \frac{4}{F_s}.$$

RTG is a gap between the uplink burst and the subsequent downlink burst in a TDD transceiver. This gap allows time for the BS to switch from receive to transmit mode and MS to switch from transmit to receive mode. During this gap the BS and MS are not transmitting modulated data, but simply allowing the BS transmitter carrier to rump up the transmit/receive antenna switch to actuate and MS receiver sections to activate.

TTG is a gap between the downlink burst and the subsequent uplink burst in a TDD transceiver. This gap allows time for the BS to switch from transmit to receive mode and MS to switch from receive to transmit mode. Also in this case, during this gap the BS and MS are not transmitting modulated data, but simply allowing the BS transmitter carrier to rump down, the transmit/receive antenna switch to actuate and MS transmitter section to activate.

Both these gaps are included in the frame by the BS and not from the MS. Their minimum durations are equal to 5 μs, and we can say that the frame in the time is composed by the uplink, the downlink and the gaps RTG and TTG and the sum of their duration must be equal to 5 μs. In Table 7.4 we find the durations of RTG and TTG in PS and the number of OFDM symbols that compose the frame at the varying of the channel bandwidth.

Channel bandwidth (MHz)	**Number of OFDM symbols**	**TTG (PS)**	**RTG (PS)**
10	47	296	168
8.75	42	218	186
7	33	376	120
5	47	148	84

Table 7.4: Durations of the several parts of the frame.

Example

We have a channel bandwidth equal to 10 MHz and we want to compute the frame duration.

1. We compute the sampling frequency F_s

$$\begin{aligned} F_s &= \left\lfloor n \cdot \frac{BW}{8000} \right\rfloor \times 8000 \\ &= \left\lfloor \frac{28}{25} \cdot \frac{10^7}{8000} \right\rfloor \times 8000 \\ &= \left\lfloor \frac{7}{5} \cdot 10^3 \right\rfloor \cdot 8000 \\ &= 11.2 \text{ MHz} \end{aligned} \tag{7.2}$$

2. We compute PS:

$$\begin{aligned} PS &= \frac{4}{FS} \\ &= 0.357\ \mu\text{s} \end{aligned} \tag{7.3}$$

3. We compute TTG:

$$\begin{aligned} TTG &= 296 \cdot PS \\ &= 105.7\ \mu\text{s} \end{aligned} \tag{7.4}$$

4. We compute RTG:

$$\begin{aligned} TTG &= 168 \cdot PS \\ &= 60\ \mu\text{s} \end{aligned} \tag{7.5}$$

5. We compute the OFDM symbol duration:

$$\begin{aligned} T_b &= \frac{1}{\Delta f} \\ &= \frac{1}{\frac{F_s}{N_{FFT}}} \\ &= \frac{1024}{11.2 \cdot 10^6} = 91.428\ \mu\text{s} \end{aligned} \tag{7.6}$$

6. We add the cyclic prefix:

$$\begin{aligned} T_s &= T_b \cdot \frac{9}{8} \\ &= 102.86\ \mu\text{s} \end{aligned} \tag{7.7}$$

7. We compute the frame length:

$$\begin{aligned} T_F &= \text{TTG} + \text{RTG} + \text{Number of OFDM symbols} \cdot T_s \\ &= 105.7 + 60 + 47 \cdot 102.86\ \mu\text{s} \\ &\simeq 5\ \text{ms}, \end{aligned} \tag{7.8}$$

7.4 Downlink Subcarriers Allocation

The downlink can be divided into three segment structures and includes a preamble which begins the transmission.

In Figure 7.3 the downlink transmission basic structure is shown.

There are several ways to perform the subcarrier allocation, but only one of them, PUSC one, is mandatory, the remaining ones are optional.

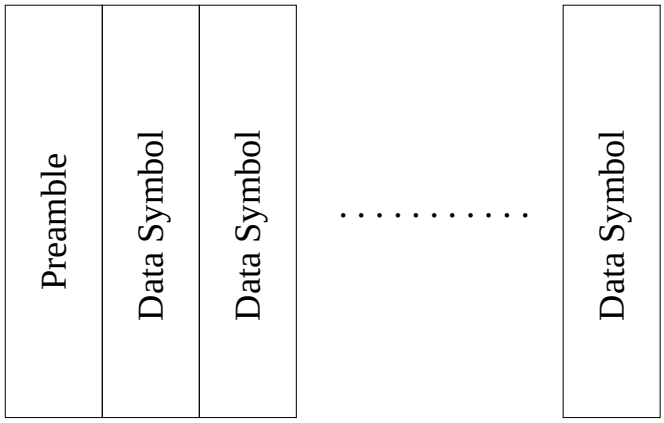

Figure 7.3: Downlink transmission basic structure.

As we can see from [109], the remaining allocations are distinguished into two types: the first one is defined potentially optional, because their implementation is not explicitly mentioned as mandatory, but from the standard [2] we may conclude it is, through not really required for the system to perform basic operations, so at the end we can say they are optional. The second type is explicitly mentioned as optional in the standard and so it may or may not be implemented.

7.4.1 Preamble

All preambles are structured as either one or two OFDM symbol. The OFDM symbols are defined by the values of the composing subcarriers. Each of those OFDM symbols contains a cyclic prefix, which length is the same as the CP for data OFDM symbols.

The first symbol of the downlink transmission is the preamble. There are three types of preamble carrier-sets, those are defined by allocation of different subcarriers for each one of them and those subcarriers are modulated using a boosted BPSK modulation with specific Pseudo-Noise (PN) code.

The preamble carrier-sets are defined using the following equation:

$$PreambleCarriesSet_n = n + 3 \cdot k, \qquad n = 0,\ 1,\ 2 \text{ and } k = 0,\ \ldots,\ 567 \tag{7.9}$$

where $PreambleCarrierSet_n$ specifies all subcarriers allocated to the specific preamble.

Each segment[1] uses one type of preamble out of the three sets in the following manner:

- Segment 0 uses preamble carrier-set 0;
- Segment 1 uses preamble carrier-set 1;
- Segment 2 uses preamble carrier-set 2.

[1] A segment is a subdivision of the OFDMA available subchannels. One segment is used for deploying a single instance of the MAC.

In the case of segment 0, the DC carrier will not be modulated at all, therefore it shall always be zeroed. For the preamble symbol there will be 172 guard band subcarrier on the left side and on the right side of the spectrum.

7.4.2 Partial Usage Of Subchannel (PUSC)

Here we give only the description of the mandatory allocation, whose name is *Partial Usage of Subchannel* (PUSC). For the other type of allocation see Appendix C

Symbol structure

We use three types of subcarriers:

- pilot subcarriers,
- data subcarriers,
- zero subcarriers.

In this case the procedure used is the following:

- the symbol is divided into basic clusters;
- zero carriers are allocated;
- pilots and data carriers are allocated into each cluster.

Each symbol is characterized by the following parameters:

1. **Number of DC subcarriers**; its index is given by $N_{FFT}/2$, counting from 0)
2. **Number of guard subcarriers, Left**, i.e. on the left side of the spectrum
3. **Number of guard subcarriers, Right**, i.e. on the right side of the spectrum
4. **Number of subcarriers per cluster**
5. **Number of clusters**
6. **Renumbering sequence**, used to renumber clusters before allocation to subchannels
7. **Number of data subcarriers in each symbol per subchannel**
8. **Number of subchannels**
9. **Basic permutation sequence**, also called *PermBase*

which depend on the FFT size.

The values of these parameters, depending on the FFT size, are enumerated in Table 7.5.

Now we make some observations:

Parameter	FFT size			
	2048	1024	512	128
Number of DC subcarriers	1	1	1	1
Number of guard subcarriers, Left	184	92	46	22
Number of guard subcarriers, Right	183	91	45	21
Number of subcarriers per cluster	14	14	14	14
Number of cluster	120	60	30	6
Number of data subcarriers in each symbol per subchannel	24	24	24	24
Number of subchannels	60	30	15	3
PermBase	$12-8$	$6-4$	5	/

Table 7.5: Symbol parameters for the available FFT size (i.e. 2048, 1024, 512 and 128).

Observation 9. *The number of pilot subcarriers in each symbol per subchannel is* 4 *for the* 2048, 1024, 512 *and* 128 *FFT size.*

In fact to compute the number of pilot subcarriers in each symbol per subchannel we follow these computations:

1. we compute the number of data subcarriers in each symbol, in all subchannels as the product of the number of data subcarriers in each symbol per subchannel times the number of subchannels

2. we compute the number of subcarriers in the all clusters as the product of the number of subcarriers per cluster and the number of clusters

3. we compute the number of all pilot subcarriers as the difference between the results found at the second and first points

4. finally, we compute the number of pilot subcarriers in each symbol per subchannel dividing the above result for the number of subchannels.

Now, we make an example to explain what is written above.

Example

We consider the case of 128 and 1024 subcarriers and use the parameters defined in Table 7.5.

1. Total number of data subcarriers in each symbol N_{td}:

$$\begin{aligned} N_{td_{128}} &= N_d \cdot N_{sch} \\ &= 24 \cdot 3 \\ &= 72 \end{aligned} \qquad \begin{aligned} N_{td_{1024}} &= N_d \cdot N_{sch} \\ &= 24 \cdot 30 \\ &= 720 \end{aligned}$$

where N_d is the number of data subcarriers in each symbol per subchannel and N_{sch} is the number of subchannels.

$$N_{tsub_{clusters_{128}}} = N_{clusters} \cdot N_{sub_{cluster}} = 6 \cdot 14 = 84 \qquad N_{tsub_{clusters_{1024}}} = N_{clusters} \cdot N_{sub_{cluster}} = 60 \cdot 14 = 840$$

2. Number of subcarriers in all clusters $N_{tsub_{clusters}}$:

   where $N_{clusters}$ is the number of clusters and $N_{sub_{cluster}}$ is the number of subcarriers per cluster.

3. Total number of pilots in a symbol N_{tp}:

$$N_{tp_{128}} = N_{tsub_{clusters}} - N_{td} = 84 - 72 = 12 \qquad N_{tp_{1024}} = N_{tsub_{clusters}} - N_{td} = 840 - 720 = 120$$

4. Number of pilots in a symbol per subchannel N_p:

$$N_{p_{128}} = \frac{N_{tp}}{N_{sch}} = \frac{12}{3} = 4 \qquad N_{p_{1024}} = \frac{N_{tp}}{N_{sch}} = \frac{120}{30} = 4$$

   where N_{sch} is the number of subchannels.

If we compute these values using the numerical values of Table 7.5 we find that the number of pilots in each symbol per subchannel is equal to 4 for all the FFT size considered.

Observation 10. *In every subchannel we have* 28 *subcarriers.*

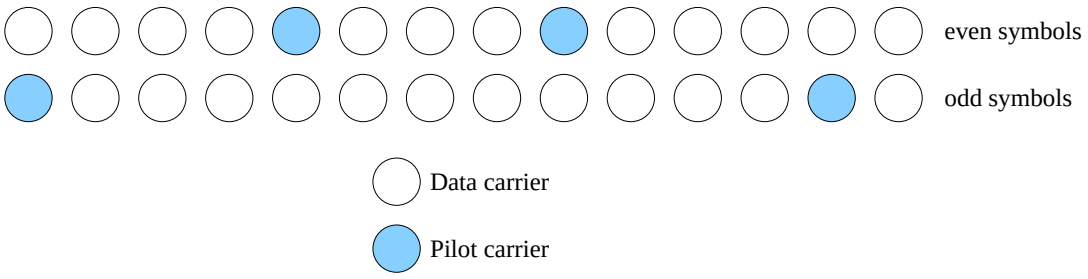

Figure 7.4: Cluster structure.

Figure 7.4 shows subcarriers from left to right in order of increasing subcarrier index. In order to determine the PUSC pilot position, symbols are counted from the beginning of the current zone and the first one is always even. Obviously, the preamble is not counted as a part of the zone.

Subchannels subcarrier allocation

The procedure is the following:

1. Subcarriers are divided into $N_{cluster}$ physical clusters with 14 adjacent subcarriers.

2. The physical clusters are renumbering into logical clusters:

$$LogicalCluster = \begin{cases} RenumberingSeguence(PhysicalCluster) \\ \quad \text{First DL zone of "All SC indicator=0"} \\ \quad \text{in STC\_DL\_Zone\_IE"} \\ RenumberingSeguence((PhysicalCluster)+ \\ \quad 13DL\_PermBase) \mod N_{cluster} \\ \quad \text{otherwise} \end{cases} \tag{7.10}$$

   and the renumbering sequence and $PermBase$ are specified in the standard (8.4.6.1.2.1 in [2]). The renumbering sequences are reported in Appendix D, the sequences of *Permutation Base*, instead,are reported in Table 7.7.

3. Logical cluster are allocated to six groups as it is shown in Table 7.6. These groups

FFT size	**Group** 0	**Group** 1	**Group** 2	**Group** 3	**Group** 4	**Group** 5
2048	0 – 23	24 – 39	40 – 63	64 – 79	80 – 103	104 – 119
1024	0 – 11	12 – 19	20 – 31	32 – 39	40 – 51	52 – 59
512	0 – 9	10 – 19	20 – 29	/	/	/
128	0 – 1	2 – 3	4 – 5	/	/	/

Table 7.6: Cluster subdivision into six major groups.

   may be allocated to segments, if a segment is being used, then at least one group shall be allocated to it. By default group 0 is allocated to segment 0, group 2 to segment 1 and group 4 to segment 2.

4. Allocating subcarriers to subchannel in each major group is performed separately for each OFDMA symbol by first allocating the pilot subcarriers within each cluster and then taking all remaining data carriers within the symbol and using the following procedure:

   - the remaining subcarrier within each major group are partitioned into groups of contiguous subcarriers
   - each subchannel consists of one subcarrier from each group, so the number of group must be equal to the number of subcarriers per subchannel: $N_{subcarriers}$, instead the number of subchannels $N_{subchannels}$ is equal to the number of subcarriers in each group, so $N_{subchannels} \cdot N_{subcarriers}$ is equal to the number of data subcarriers

- the partition into subchannels is given by the *permutation formula* (7.11):

$$\begin{aligned} subcarrier(k,s) &= N_{subchannels} \cdot n_k + \{p_s[n_k \mod N_{subchannels}] + \\ &+ DL\_PermBase\} \mod N_{subchannels} \end{aligned} \tag{7.11}$$

where

- $subcarrier(n,s)$ is the index of the subcarrier n into the subchannel s; its values are 0, 1, ..., $N_{subcarriers} \cdot N_{subchannels} - 1$ ($n = 0, 1, \ldots, Nsubcarriers - 1$ and $s = 0, 1, \ldots, N_{subchannels} - 1$)
- $n_k = (k + 13s) \mod N_{subcarriers}$
- $p_s[j]$ is the series obtained by rotating basic permutation sequence cyclically to the left s times
- $DL\_PermBase$ is an integer ranging from 0 to 31, which is set to preamble $IDcell$ in the first zone and determined by the DL_MAP for other zones.

The basic permutation sequence depends on the FFT size and on the number of the major group in which we find the subcarrier to be allocated and is reported in Table 7.7.

FFT size	**Even group**	**Sequence**	**Odd group**	**Sequence**
2048	12 subchannels	6, 9, 4, 8, 10, 11, 5, 2, 7, 3, 1, 0	8 subchannels	7, 4, 0, 2, 1, 5, 3, 6
1024	6 subchannels	3, 2, 0, 4, 5, 1	4 subchannels	3, 0, 2, 1
512	5 subchannels	4, 2, 3, 1, 0	/	-
128	/	-	/	-

Table 7.7: DL_PermBase.

Example - 128 **FFT size**

We consider a 128 FFT size.

We have 6 clusters of 14 adjacent subcarriers, hence we allocate 84 subcarriers, because the remaining are guard subcarriers (22 on the left side and 21 on the right side) and DC subcarriers (1).

In Figure 7.5 the subdivision in clusters is represented. The blue numbers represent the physical numeration, instead the black ones are the number of subcarriers in each groups.

Through the renumbering sequence (see Appendix D) the physical clusters are renumbering into logical clusters, as it is shown in Figure 7.6.

As we can see from the legend in the Figure 7.6, the blue numbers still represent the physical numeration and the green ones are the new logical numeration.

The renumbered clusters are subdivided into three groups (see Table 7.6).

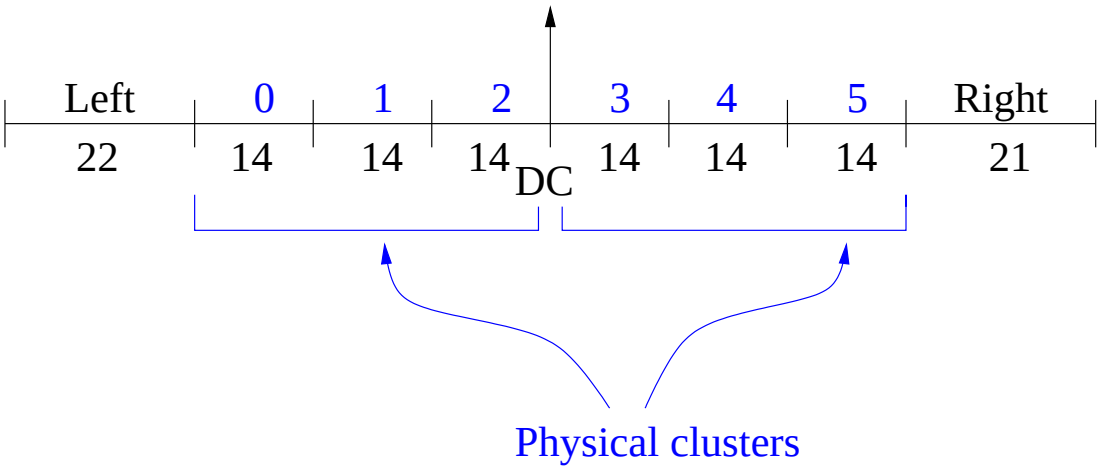

Figure 7.5: Physical clusters.

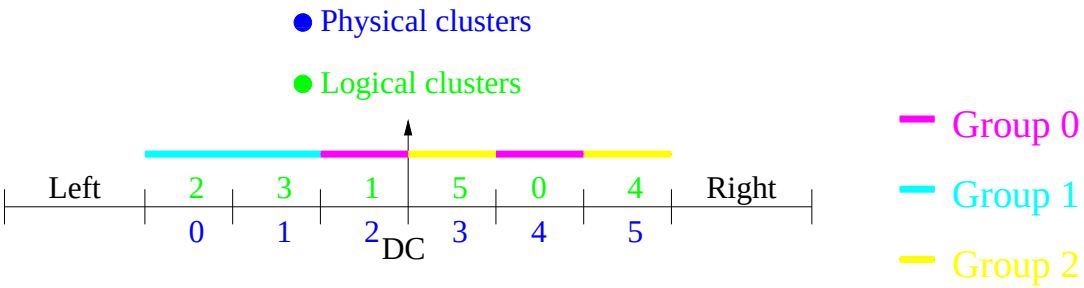

Figure 7.6: Logical clusters.

At this point, we need to allocate the pilot subcarriers, which are 4 in each subchannel, as we pointed out in the above example. The pilots are allocated to fixed positions (see Figure 7.4):

- for the even symbols the pilots are the 5^{th} and 9^{th} subcarriers
- for the odd symbols the pilots are the 1^{st} and 13^{th} subcarriers.

The structure of the six cluster is represented in Figure 7.7 and in this one the subdivision in groups of adjacent subcarriers is also represented. The remaining subcarriers are numbered from 0 to 71.

We point out that, in this Figure we assume that the cluster are in logical order, so, looking Table 7.6, we can show also the major groups.

In 128 FFT size we have 3 channels of 24 subcarriers (see Table 7.5).

Now, in each major group, we shall clustering the subcarriers into groups of dimension equal to the number of channels defined in it and then, through equation (7.11), we assign the subcarriers to subchannels.

We observe that 128 FFT size is a particular case, because we have one subchannels per major group, hence the group of adjacent subcarriers are made of only once. Then,

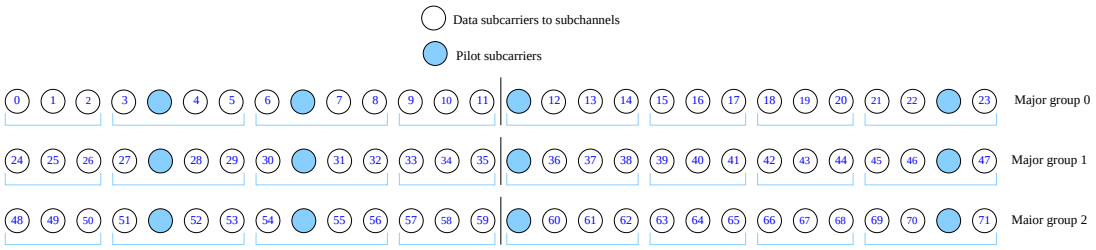

Figure 7.7: Pilots allocation.

from Table 7.7, we see that we haven't basic permutation sequences and so equation (7.11) is reduced to:

$$subcarrier(k, s) = N_{subchannels} \cdot n_k \tag{7.12}$$

where:

- $k = 0, 1, \ldots, 23$
- $s = 0$
- $N_{subchannels} = 1$
- $n_k = (k + 13 \cdot s) \mod N_{subcarriers} = k \mod 24 = 0, 1, \ldots, 23.$

As it is obvious from the above observation and as we can see in Figure 7.8, the subcarriers of a major group, in this case, form a subchannel.

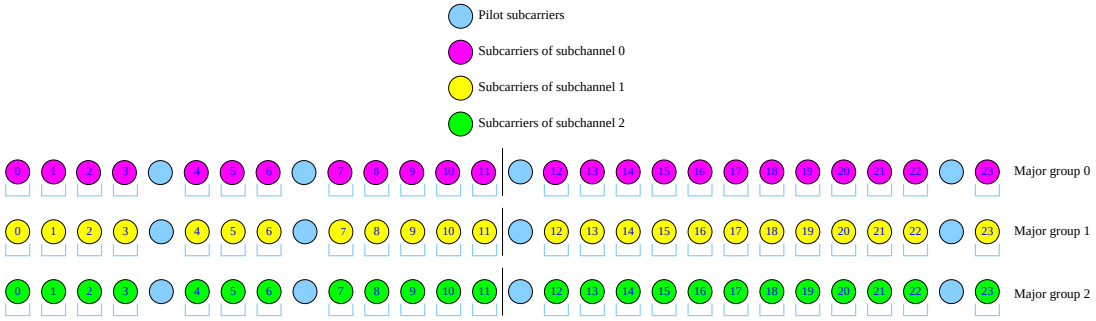

Figure 7.8: Subcarriers to subchannel assignment.

In Figure 7.8 we use four colors to represent:

- the pilots (light blue color)
- the subcarriers allocated to subchannel 0 (magenta color)
- the subcarriers allocated to subchannel 1 (yellow color)
- the subcarriers allocated to subchannel 2 (green color)

Example - 1024 FFT size

To show the use and results of the equation (7.11) we consider an example with 1024 subcarriers.

First we show the subdivision in physical clusters (Figure 7.9) and then the renumbering into logical clusters (Figure 7.10).

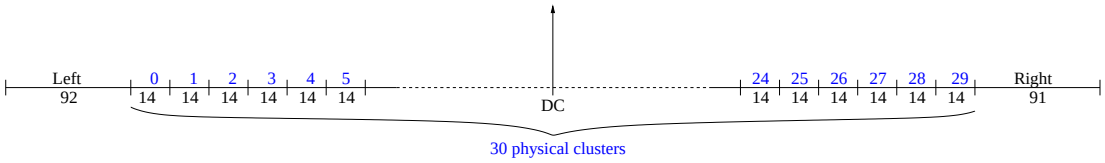

Figure 7.9: Physical clusters.

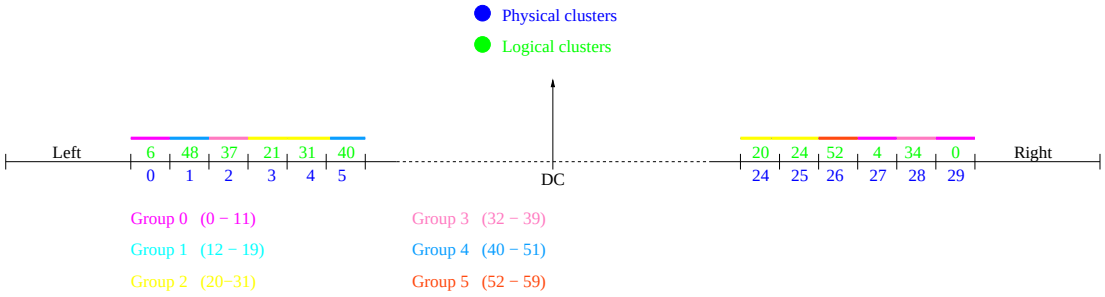

Figure 7.10: Physical clusters.

In Figure 7.11 we insert the pilot subcarriers into each cluster. As we can see the positions are the same seen in the above example.

After the insertion of the pilots we subdivide the remaining subcarriers within each of the six major group in groups of adjacent subcarriers of the dimension of the number of 6 and 4 if we are considering even or odd groups respectively.

Then, a subcarrier from each group, chosen through equation (7.11), forms the subchannels.

In Figures 7.12 and 7.13 we show the allocation of the subcarriers to subchannels in the major groups 0 and 1 considering a $DL\_PermBase$ equal to 0. Considering, then, that in the last two major groups the allocation would be the same, we represent the allocation if we assume $DL\_PermBase = 15$.

To distinguish the subchannels we use two sets of different color for the two major groups. In the first major group we have:

- pilot subcarriers (light blue color)
- subcarrier allocated to subchannel 0 (magenta color)
- subcarrier allocated to subchannel 1 (yellow color)
- subcarrier allocated to subchannel 2 (green color)

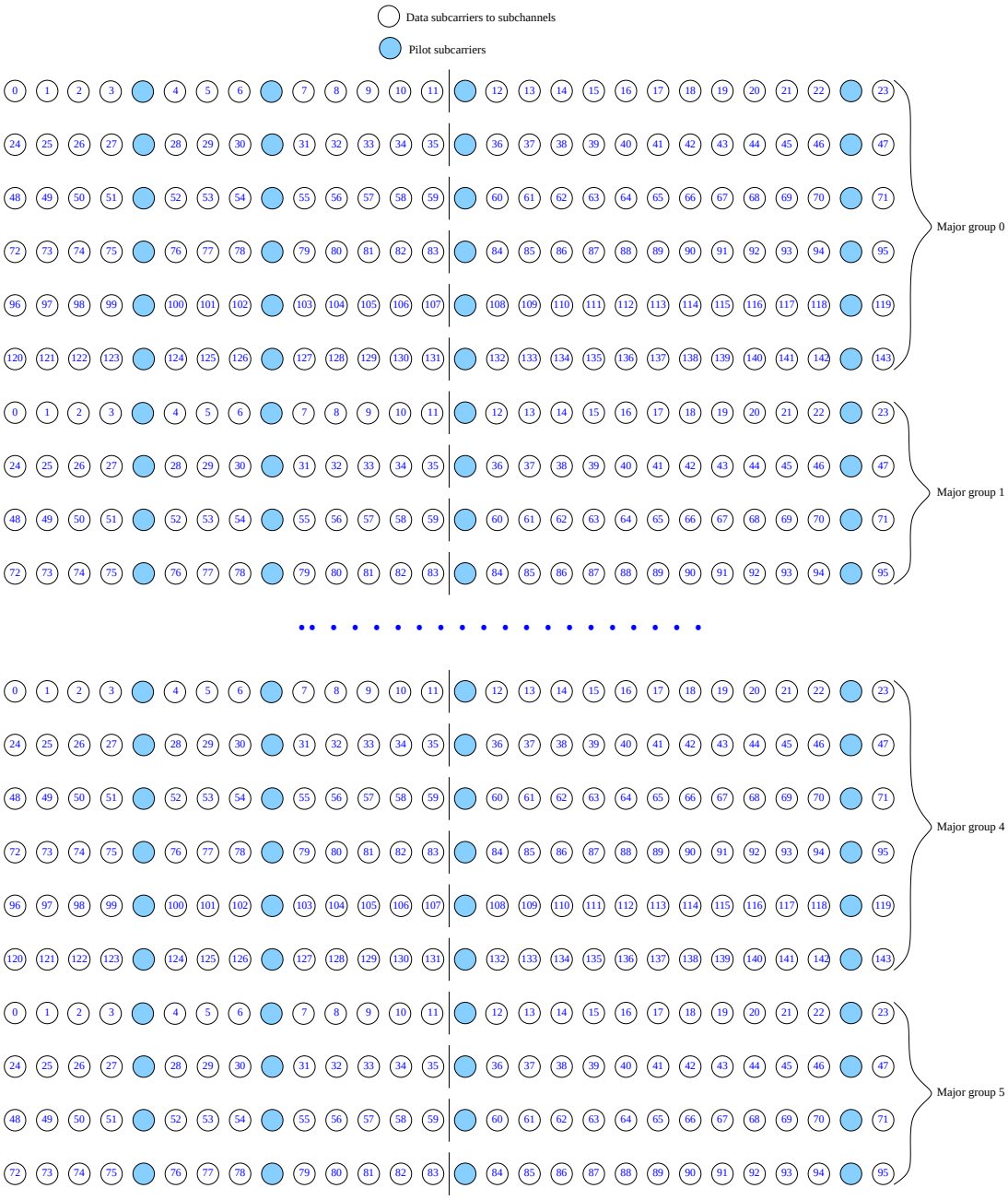

Figure 7.11: Pilot subcarriers in the clusters.

- subcarrier allocated to subchannel 3 (cyan color)
- subcarrier allocated to subchannel 4 (red color)
- subcarrier allocated to subchannel 5 (pink color)

and in the second major group:

- subcarrier allocated to subchannel 6 (dark cyan color)
- subcarrier allocated to subchannel 7 (light pink color)

- subcarrier allocated to subchannel 8 (dark green color)
- subcarrier allocated to subchannel 9 (dark magenta color)

Figure 7.12: Subcarrier allocation into subchannels, for the first two major group, considering a DL_PermBase equal to 0.

7.5 Channel coding

Channel coding procedures include:

- **randomization**
- **FEC coding**
- **bit interleaving**
- **repetition**
- **modulation**.

In Figure 7.14 the channel coding process is shown and we point out that repetition shall only be applied to QPSK modulation.

Figure 7.13: Subcarrier allocation into subchannels, for the last two major group, considering a DL_PermBase equal to 15.

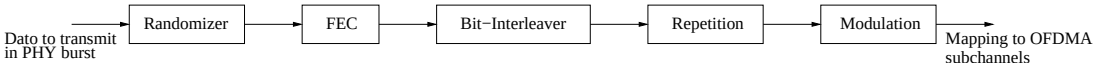

Figure 7.14: Channel coding process for regular and repetition coding transmission.

7.5.1 Randomization

This operation is mandatory in [2].

Data randomization is performed on all data transmitted in uplink and downlink, except the FCH. It is initialized on each FEC block, through the vector

$$[\text{LSB}]\ \ 011011100010101\ \ [\text{MSB}]$$

and in Figure 7.15 the PRBS generator for data randomization is represented.

From the Figure 7.15 we see that the PRBS generator shall be $1+x^{14}+x^{15}$. Preambles are not randomized and each data byte to be transmitted shall enter sequentially into the randomizer, MSB first. The randomizer sequence is applied only to information bits. If the amount of data to transmit does not fit exactly the amount of data allocated, so padding may be added at the end of the transmission block.

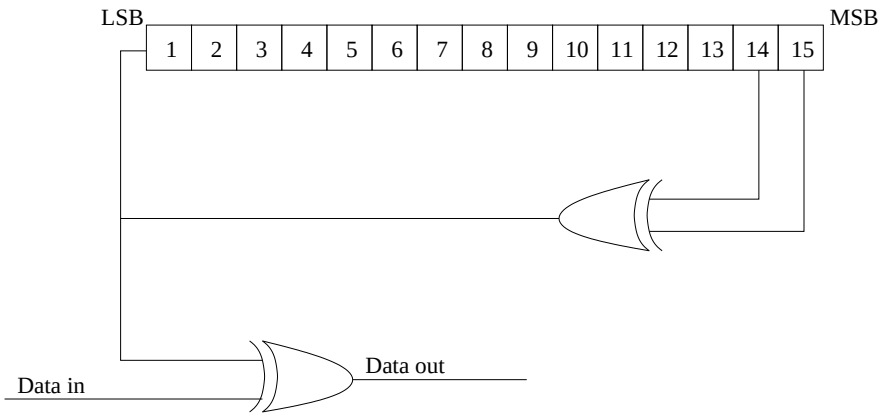

Figure 7.15: PRBS generator for data randomization.

H-ARQ requires the randomizer pattern is identical for each H-ARQ attempt, so, for H-ARQ, it shall be initialized with the vector created as it is shown in Figure 7.16.

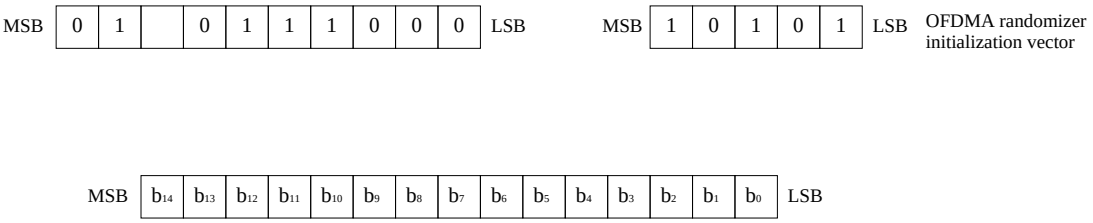

Figure 7.16: Creation of the OFDMA randomizer initialization vector for H-ARQ.

7.5.2 Encoding

The WiMAX standard defines some codes, but only one of them is mandatory: the *tail-biting convolutional encoding*. The other ones are: Block Turbo Code (BTC), Convolutional Turbo Code (CTC) and Low-Density Parity-Check code (LDPC). Here we describe only the mandatory and the LDPC ones, because these are the ones of out interest. The other ones are described in Appendix E.

The encoding block size depends on the number of slots allocated and the modulation specified for the current transmission. Concatenation of a number of slots shall be performed in order to make larger blocks of coding, where it is possible, with the limitation of not exceeding the largest supported block size for the applied modulation and coding.

For any modulation and FEC rate, given an allocation of n slots, the following parameters are defined:

- **j**: parameter dependent on the modulation and FEC rate
- **n**: number of allocated slots/repetition factor
- **k**: $\lfloor n/j \rfloor$

- **m**: $n \mod j$.

We define a rule for the slots concatenation; it is described in Table 7.8.

Number of slots	Slots concatenated
$n \leq j$	1 block of n clusters
$n > j$, $n \mod j = 0$	k blocks of j slots
$n > j$, $n \mod j \neq 0$	$(k-1)$ blocks of j slots 1 block of $2 \times \lceil (m+j)/2 \rceil$ slots 1 block of $2 \times \lfloor (m+j)/2 \rfloor$ slots

Table 7.8: Rule for the slots concatenation.

In Table 7.9 we find the value of j at the varying of the modulation and FEC rate.

Modulation and rate	j
QPSK 1/2	6
QPSK 3/4	4
16 - QAM 1/2	3
16 - QAM 3/4	2
64 - QAM 3/4	1
64 - QAM 3/2	1

Table 7.9: Values of j at the varying of modulation and FEC rate.

There are severely encoding methods, but only one of them is mandatory and so need to be implemented: the others are optional, but in [109] is required to implement one of them.

Convolutional coding (CC) - mandatory

Each FEC block is encoded by the binary convolutional encoder, which shall have native rate 1/2, and a constraint length equal to $k = 7$ and shall use the following generator polynomials codes to derive its two code bits:

$$G_1 = 171_{\text{OCT}} \quad X \tag{7.13}$$

$$G_2 = 133_{\text{OCT}} \quad Y \tag{7.14}$$

In Table 7.10 the puncturing patterns and serialization order that shall be used to realize different code rates are defined; in it "1" means a transmitted bit and "0" means a removed bit.

Rate	$\mathbf{d_{free}}$	X	Y	XY
1/2	10	1	1	X_1Y_1
2/3	6	10	11	$X_1Y_1Y_2$
3/4	5	101	110	$X_1Y_1Y_2X_3$

Table 7.10: Convolutional code with puncturing configuration.

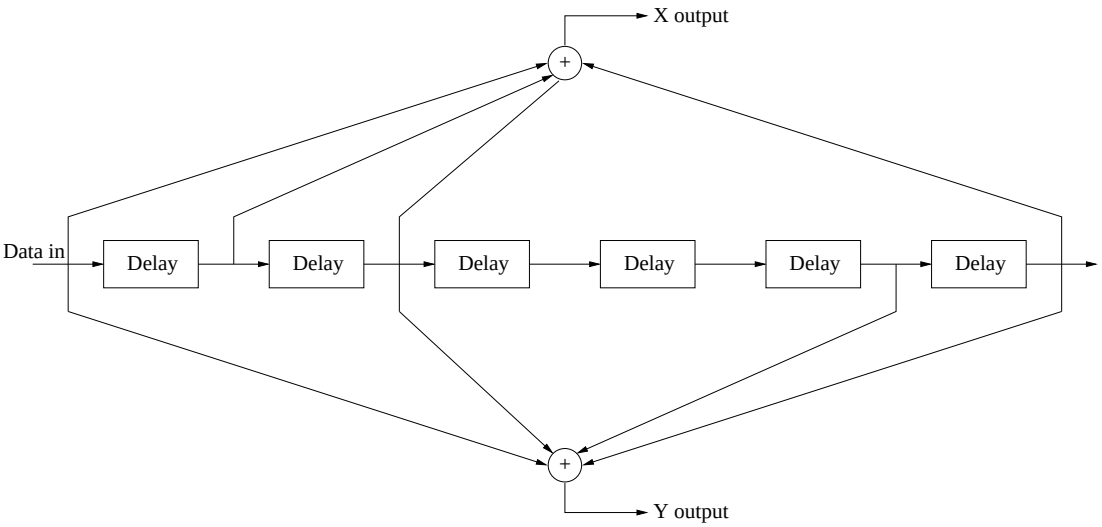

Figure 7.17: Convolutional encoder of rate 1/2.

Each FEC block is encoded by a *tail-biting convolutional encoder*, which is achieved by initializing the encoder's memory with the last data bits of the FEC block being encoded.

Following (in Table 7.11) we define the basic sizes of the useful data payloads to be encoded in relation with the selected modulation type, encoding rate and concatenation rule.

	QPSK		16 - QAM		64 - QAM		
Encoding rate	1/2	3/4	1/2	3/4	1/2	2/3	3/4
Data payload (bytes)	6						
		9					
	12		12				
	18	18		18	18		
	24		24			24	
		27					27
	30						
	36	36	36	36	36		

Table 7.11: Useful data payload for slot.

H-ARQ implementation is optional and an Incremental Redundancy (IR) based on it is taking the puncturing pattern into account and for each retransmission the code block is not the same.

The puncture patterns are predefined or can be easily deducted from the original pattern and can be selected based on SPID (which is the SubPacket Identifier (ID), used to identify the four subpackets generated from an encoder packet).

At the receiver, the received signals are de-punctured according to its specific pattern, decided by the current SPID, then the combination is performed at bit metrics level.

		Code rate			
		1/2	2/3	3/4	5/6
SPID = 0	X	1	10	101	10101
	Y	1	11	110	11010
SPID = 1	X	1	01	011	01011
	Y	1	11	101	10101
SPID = 2	X	1	10	110	10110
	Y	1	11	011	01011
SPID = 3	X	1	01	101	01101
	Y	1	11	110	10110

Table 7.12: Puncture pattern definition for H-ARQ.

About the Table 7.12 there is a specification:

- SPID = 0: puncture pattern equal to the mandatory one
- SPID = 1: puncture pattern left cyclic shift of the one from SPID = 0
- SPID = 2 and SPID = 3: patterns are governed by the same rule.

Low density parity check code - optional

Code description

The LDPC code is based on a set of one or more fundamental LDPC codes. Each of the fundamental codes is a systematic linear block code.

Each LDPC code is defined by a matrix $\boldsymbol{H}$ of size $m \times n$, where n is the size of the code and m is the number of parity-check bits in the code. The number of systematic bits is $k = n - m$.

The matrix $\boldsymbol{H}$ is define as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{P}_{0,0} & \boldsymbol{P}_{0,1} & \boldsymbol{P}_{0,2} & \cdots & \boldsymbol{P}_{0,n_b-2} & \boldsymbol{P}_{0,n_b-1} \\ \boldsymbol{P}_{1,0} & \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} & \cdots & \boldsymbol{P}_{1,n_b-2} & \boldsymbol{P}_{1,n_b-1} \\ \boldsymbol{P}_{2,0} & \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} & \cdots & \boldsymbol{P}_{2,n_b-2} & \boldsymbol{P}_{2,n_b-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{P}_{m_b-1,0} & \boldsymbol{P}_{m_b-1,1} & \boldsymbol{P}_{m_b-1,2} & \cdots & \boldsymbol{P}_{m_b-1,n_b-2} & \boldsymbol{P}_{m_b-1,n_b-1} \end{bmatrix} = \boldsymbol{P}^{H_b} \tag{7.15}$$

where $\boldsymbol{P}_{i,j}$ is one of a set of $z \times z$ permutation matrices or a $z \times z$ zero matrix. The matrix $\boldsymbol{H}$ is expanded from a binary base matrix $\boldsymbol{H}_b$ of size $m_b \times n_b$, where $n_b = 24$, $n = z \cdot n_b$ and $m = z \cdot m_b$, with z an integer. The base matrix is expanded by replacing each 1 in the base matrix with a $z \times z$ permutation matrix, and each 0 with a $z \times z$ zero matrix.

The set of permutation matrices contains the $z \times z$ identity matrix and circular right shifted version of the identity matrix.

H_b is partitioned into two sections, where $\boldsymbol{H}_{b1}$ corresponds to the systematic bits

and $\boldsymbol{H}_{b2}$ corresponds to the parity check bits, such that

$$\boldsymbol{H} = [(\boldsymbol{H}_{b1})_{m_b \times k_b} | (\boldsymbol{H}_{b2})_{m_b \times m_b}]$$

Section $\boldsymbol{H}_{b2}$ is further partitioned into two sections: $\boldsymbol{h}_b$ and $\boldsymbol{H}'_{b2}$, where $\boldsymbol{h}_b$ has odd weight, and $\boldsymbol{H}'_{b2}$ has a dual diagonal structure with matrix elements at row i, column j equal to 1 for $i = j$, for $i = j + 1$ and 0 elsewhere.

The base matrix has $h_b(0) = 1$, $h_b(m_b - 1) = 1$ and a third value $h_b(j)$, $0 < j < m_b - 1$ equal to 1.

So:

$$\begin{aligned} \boldsymbol{H}_{b2} &= [\boldsymbol{h}_b | \boldsymbol{H}'_{b2}] \\ &= \left[\begin{array}{c|cccccc} h_b(0) & 1 & & & & \\ h_b(1) & 1 & 1 & & 0 & \\ \cdot & & 1 & \ddots & & \\ & & & \ddots & 1 & \\ \cdot & & & & & \\ \cdot & & 0 & & 1 & 1 \\ h_b(m_b - 1) & & & & & 1 \end{array}\right] \end{aligned} \tag{7.16}$$

The largest code length of each code rate is $n = 2304$ and, in any case, n must be a multiple of $n_b = 24$.

Code rate and block size adjustment

The LDPC code flexibility supports different block sizes for each code rate through the use of an expansion factor z.

Table 7.13 shows the parameters for each code rates:

- $n_{(bytes)} = n_{(bits)}/8$
- $z = n_{(bits)}/24$
- $k = n_{(bytes)} * R$

Packet encoding

The encoding block size k shall depend on the number of subchannels allocated and the modulation specified for the current transmission. As usual, concatenation of a number of subchannels shall be performed in order to make larger blocks of coding where it is possible, with the limitation of not passing the largest block under the same coding rate. For the concatenation rule see Table 7.15, but remember that for the LDPC the concatenation does not depend on the code rate.

For any modulation and FEC rate, given an allocation of N_{SCH} subchannels, we define the following parameters:

n (bits)	n (bytes)	z factor	k bytes				Number of subchannels		
			$R = 1/2$	$R = 2/3$	$R = 3/4$	$R = 5/6$	QPSK	$16-$QAM	$64-$QAM
576	72	24	36	48	54	60	6	3	2
672	84	28	42	56	63	70	7	–	–
768	96	32	48	64	72	80	8	4	–
864	108	36	54	72	81	90	9	–	3
960	120	40	60	80	90	100	10	5	–
1056	132	44	66	88	99	110	11	–	–
1152	144	48	72	96	108	120	12	6	4
1248	156	52	78	104	117	130	13	–	–
1344	168	56	84	112	126	140	14	7	–
1440	180	60	90	120	135	150	15	–	5
1536	192	64	96	128	144	160	16	8	–
1632	204	68	102	136	153	170	17	–	–
1728	216	72	108	144	162	180	18	9	6
1824	228	76	114	152	171	190	19	–	–
1920	240	80	120	160	180	200	20	10	–
2016	252	84	126	168	189	210	21	–	7
2112	264	88	132	176	198	220	22	11	–
2208	276	92	138	184	207	230	23	–	–
2304	288	96	144	192	216	240	24	12	8

Table 7.13: LDPC block sizes and code rates.

- j_i: parameter dependent on the modulation and number of antennas in case of spatial multiplexing

- N_{SCH}: number of allocated subchannel

- $F = \text{floor}(N_{SCH}/j_i)$

- $M = N_{SCH} \mod j_i$

Table 7.14 specifies the concatenation of subchannels for different allocations and modulations.

Modulation	j_1	j_2	j_3	j_4
QPSK	24	12	8	6
16 - QAM	12	6	4	3
64 - QAM	8	4	2	2

Table 7.14: Parameter j for LDPC.

$N_{SCH} \leq j_i$	1 block of N_{SCH} subchannels
$N_{SCH} > j_i$	If $M == 0$ F blocks of j_i slots else $(F-1)$ blocks of j_i slots 1 blocks of $\lceil((M+j_i)/2))\rceil$ slots 1 blocks of $\lfloor((M+j_i)/2))\rfloor$ slots end

Table 7.15: Subchannels concatenation.

Control information and packets that result in a codeword size n of less than 576 bits are encoded using CC.

7.5.3 Interleaving

All encoded data bits shall be interleaved by a block interleaver with a block size corresponding to the number of coded bits per the encoded block size N_{cbps}. The interleaver is defined by a two-step permutation. The first ensures that adjacent coded bits are mapped onto nonadjacent subcarriers. The second permutation insures that adjacent coded bits are mapped alternately onto less or more significant bits of the constellation, thus avoiding long runs of lowly reliable bits.

Let N_{cpc} be the number of coded bits per subcarrier, i.e. 2, 4 or 6 for QPSK, 16−QAM or 64−QAM, respectively. Let $s = N_{cpc}/2$. Within a block of N_{cbps} bits at transmission, let k be the index of the coded bit before the first permutation, m_k be the index of that coded bit after the first and before the second permutation and let j_k be the index after the second permutation, just prior to modulation mapping, and d be the modulo used for the permutation.

The first permutation is defined by Equation (7.17):

$$m_k = \frac{N_{cbps}}{d} \cdot k_{mod(d)} + \lfloor \frac{k}{d} \rfloor \qquad k = 0,\, 1,\, \ldots,\, N_{cbps} - 1 \text{ and } d = 16 \tag{7.17}$$

The second permutation is defined by Equation (7.18):

$$j_k = s \cdot \lfloor \frac{m_k}{s} \rfloor + (m_k + N_{cbps} - \lfloor \frac{d \cdot m_k}{N_{cbps}} \rfloor)_{mod(2)} \qquad k = 0,\, 1,\, \ldots,\, N_{cbps} - 1 \text{ and } d = 16 \tag{7.18}$$

The de-interleaver, which performs the inverse operation, is also defined by two permutations. Within a received block of N_{cbps} bits, let j be the index of a received bit before the first permutation; m_j be the index of that bit after the first and before the second permutation; let k_j be the index of that bit after the second permutation, just prior to delivering the block to the decoder.

The first permutation is defined by Equation (7.19):

$$m_j = s \cdot \lfloor \frac{j}{s} \rfloor + (j + \lfloor \frac{d \cdot j}{N_{cbps}} \rfloor) \qquad j = 0,\, 1,\, \ldots,\, N_{cbps} - 1 \text{ and } d = 16 \tag{7.19}$$

The second permutation is defined by Equation (7.20):

$$k_j = d \cdot m_j - (N_{cbps} - 1) \cdot \lfloor d \cdot \frac{m_j \cdot m_j}{N_{cbps}} \rfloor \qquad j = 0,\, 1,\, \ldots,\, N_{cbps} - 1 \text{ and } d = 16 \tag{7.20}$$

7.5.4 Modulation

There are different way to modulate the data and pilot subcarriers.

Data modulation

After bit interleaving, the data bits are entered serially to the constellation mapper. Gray-mapped QPSK and 16−QAM are supported , whereas 64−QAM is optional. The

constellations shall be normalized by multiplying the constellation point with the factor $1/\sqrt{2}$, $1/\sqrt{10}$ and $1/\sqrt{42}$ respectively, to achieve equal average power.

Pilot modulation

Pilot sequences w_k are generated through a PRBS generator, whose polynomial is $X^{11} + X^9 + 1$. Than the value of the pilot for OFDM symbol k is derived from w_k. For each pilot the BPSK modulation shall be derived as follows:

$$\begin{aligned} Re\{c_k\} &= \frac{8}{3}\left(\frac{1}{2} - w_k\right) \\ Im\{c_k\} &= 0 \end{aligned} \tag{7.21}$$

where the multiplication by 8/3 realizes a boost of 2.5 dB.

Preamble modulation

Once we have defined the sequence of subcarriers that must be used in the preamble, as described in Section 7.4.1, Let's call them pre_k. Their modulation is a boosted BPSK:

$$\begin{aligned} Re\{p_k\} &= 4 \cdot \sqrt{2}\left(\frac{1}{2} - pre_k\right) \\ Im\{p_k\} &= 0. \end{aligned} \tag{7.22}$$

Chapter 8

Performance evaluation

In the Chapters above channel estimation and channel coding techniques have been described. In this Chapter those techniques will be applied to a WiMAX standard compliant systems. First of all the implemented WiMAX system will be described and than the performance with infinite and finite computational resource will be presented.

8.1 WiMAX simulator

A standard compliant implementation of the PHY level of a downlink 802.16e system was developed in a Matlab/Simulink software environment. The frame and symbol structures are compliant with the standards specifications contained in [1] and [2].

The simulation software is able to evaluate the performance of the PHY layer in different propagation scenarios (e.g. urban/suburban/rural) and in various interference contexts.

All the simulations are carried out in a baseband environment, neglecting the implementation of pass band frequency modulation and of carrier/symbol/frame synchronization.

The transmitter supports all the convolutional codes and QAM modulation schemes (MCS) approved by the previously mentioned standards. The useful payload is contained in a single data region, and fills all the available subcarriers. Also the irregular LDPC code with all the QAM modulation are supported, but only for a code rate equal to 1/2. Finally for the QPSK modulation we consider also a turbo code with rate 1/3, a regular LDPC code with rate 3/4 and a concatenation of the last two with an overall rate equal to 1/4. All the transmitting BSs are assumed to adopt different DL permutation bases and the same MCS of the useful payload. By the way, simulations can be arranged where these parameters are set individually for each interfering BS.

Figure 8.1 gives a block representation of the implemented system and Figures 8.2, 8.3 and 8.4 give a more accurate representation of the block transmitter, frame assembler and receiver.

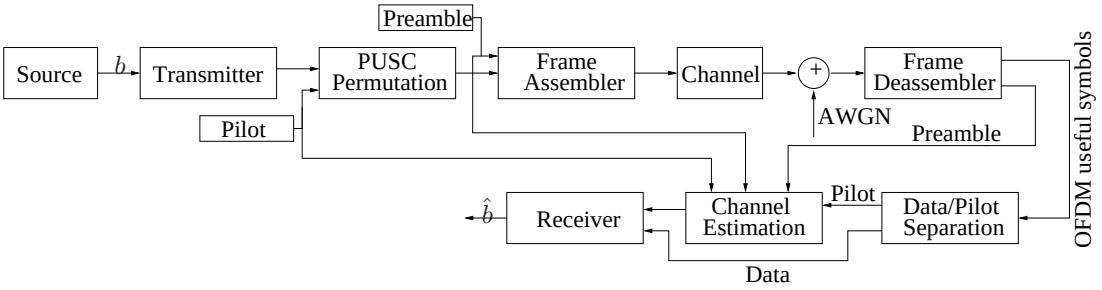

Figure 8.1: Simulation system - simplified scheme.

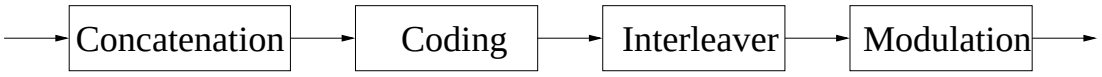

Figure 8.2: Transmitter.

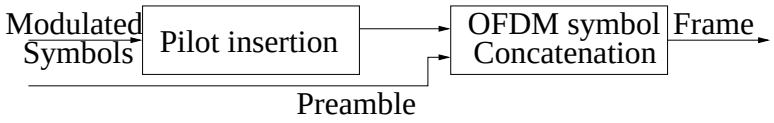

Figure 8.3: Frame assembler.

Figure 8.4: Receiver.

8.1.1 Transmitter

The transmitter implements the concatenation and coding. We can choose between five different type of coding: convolutional code, regular and irregular LDPC codes, turbo code or concatenation. The convolutional code (CC) is implemented exactly as described in Section 7.5.2. For the other codes we consider the same concatenation rule (see Table 7.8) because we want to encode sourceword of the same length. For the irregular LDPC code, the matrix used is the one defined in the standard for sourceword of length 288 and rate 1/2 (see Table 7.13). We don't use the concatenation rule described in Table 7.15, because, for the case we are considering (a full loaded system with 512 subcarriers, QPSK modulation and code rate equal to 1/2) we find the codeword length is 720, but this value is not included in Table 7.13, and for values different from the ones listed in that table the standard suggests to use the CC.

After the coding operation the interleaver one is performed as described in 7.5.3.

The last operation is the modulation one: we can choose between QPSK, 16−QAM and 64−QAM, even if not all the combinations of code-modulation are available.

8.1.2 PUSC Permutation

This is a complicated block which implements the insertion of the pilots and the permutation as described in Section 7.4.2.

First of all we divide the data in even and odd symbols and we insert the pilots to form a cluster. For every FFT size the cluster is made of one even and one odd symbol and each of them is made of 12 data subcarriers and 2 pilots.

For the even symbols pilots have indexes 5 and 9, for the odd 1 and 13, assuming we are numerating the carriers starting from 1.

These clusters, which are made of adjacent subcarriers, are called *physical clusters* and are renumbering into *logical clusters* as defined in 7.10.

Logical clusters are allocated to groups (see Table 7.6), then subcarriers are clustered into groups whose dimension is equal to the number of channels for the FFT size and, finally, through Equation (7.11), they are allocated to subchannels.

8.1.3 Frame assembler

At the output of the PUSC permutation block we have the data and pilots we want to transmit on the OFDMA symbols, so, in the frame assembler virtual and DC subcarriers are added to form the complete OFDMA symbol. At the beginning of the frame we add the preamble which is one OFDM symbol modulated with a boosted BPSK. For a 512 system, it contains data only over 143 subcarriers, generated with a PRBS generator as described in Section 7.4.1.

8.1.4 Channel

Once we have the frame we want to transmit the data over a wireless channel. In this simulator we can use four different types of channel: AWGN, exponential with Rayleigh fading, SCME [110] and Winner [111].

8.1.5 Frame deassembler and Data/pilots separation

Aim of the frame deassembler block is simply the division of preamble and useful OFDM symbols. More complex is the data/pilots separation block: it extracts pilots from the receiver symbols, performing the PUSC inverse permutation.

In this block we find the subchannels, the groups, the logical and physical clusters. Once we have the latter ones we know exactly the position of the pilots and so it is easy to extract them from the received symbols.

8.1.6 Channel estimation

In this case we have four inputs: the transmitted pilots and preamble and the received ones. Channel estimation can be performed through a reduced rank LS (see Chapter 2) estimator on both preamble and pilots or a MMSE estimator on preamble

and a LS estimator on pilots. The choice of these estimators is motivated from the results presented in Chapter 2.

8.1.7 Receiver

At this point we have the channel estimation and the received data so we perform the equalization. Actually to measure the loss due to the channel estimation we can perform also the equalization with the perfect channel, i.e. we assume the knowledge of the channel.

The output of the equalizer is the input for the soft demapper, then the LLR are de-interleaved and decoded. Finally we perform a reshape to have the bits ordered as when they were generated.

8.2 System parameters

We consider a 512 subcarrier system. In this case we use 360 subcarriers for data transmission and 60 for pilots. All the remaining except the DC are virtual subcarriers so we don't transmit anything over them. The bandwidth is 5 MHz and the carrier frequency 2 GHz, thus the symbol period is $T_s = 1.0286e-04$ where we consider a CP equal to 1/8 of the useful symbol.

A single frame has one preamble and 24 OFDMA symbols and each iteration of our simulations performs the transmission of 200 frames.

Finally, mobile speed can vary between 3, 50 or 130 km/h.

Table 8.1 resumes all the parameters for the considered system.

8.3 Infinite computational resource

Figure 8.5 represents the performance of the WiMAX implemented system when we consider an AWGN channel and a CC codes. We consider all the rate and modulation enumerated in the standard [2].

Assuming we know the channel, we plot the performance when we consider LDPC and CC codes with rate 1/2. As said above for the considered system, the LDPC subchannel concatenation rule doesn't give a valid codeword size, but if we use the slots concatenation defined for the CC, we have a codeword of 576 bits.

Figures 8.6, 8.7 and 8.8 represents the performance for the considered system when we assume to concatenate the data through the slots concatenation rule (Table 7.8) and to encode them through the CC and LDPC codes with rate 1/2. The modulation considered are QPSK (blue lines), 16−QAM (green line) and 64−QAM (magenta line). Two different styles are used: the solid lines represent performance for the CC and the dotted lines performance for the LDPC code.

Observing these figures we can conclude that especially for QPSK and 16−QAM it could be a good idea to use the LDPC codes with a different concatenation rule.

FFT size	512
Channel bandwidth	5 MHz
Carrier frequency	2 GHz
Sampling factor	1.12
CP ratio	1/8
Sampling frequency	5.6 MHz
Subcarrier spacing	$1.0938e+04$
Useful symbol time	$9.1429e-05$
CP time	$1.1429e-05$
OFDMA symbol time	$1.0286e-04$
Sampling time	$1.7857e-07$
Number of OFDM symbols (downlink)	24
Number of preamble OFDM symbols	1
Number of DC subcarriers	1
Number of guard subcarrier, Left	46
Number of guard subcarrier, Right	45
Number of subcarriers per cluster	14
Number of clusters	30
Number of data subcarriers in each symbol per subchannel	24
Number of subchannels	15
PermBase	5

Table 8.1: Parameters for the considered system.

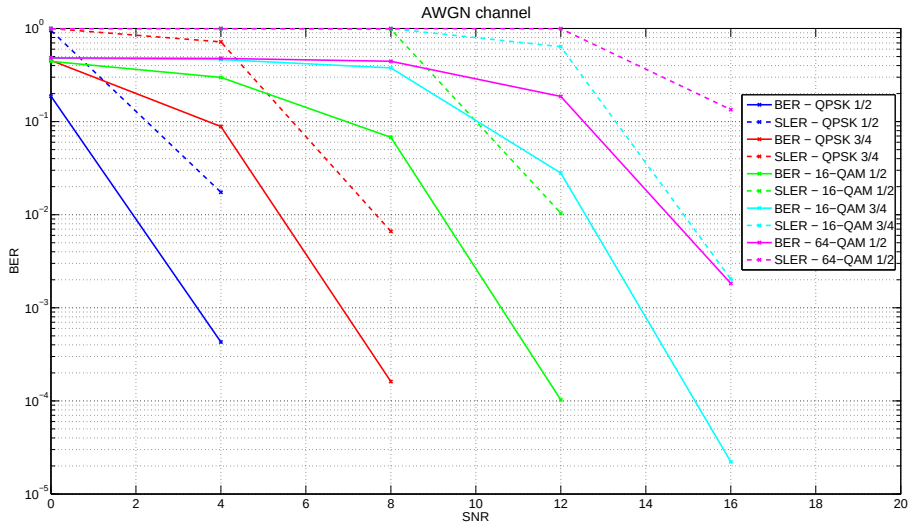

Figure 8.5: Bit error rate performance for a WiMAX system with AWGN channel and CC.

In Chapter 6 we proposed a concatenation of LDPC and turbo codes as outer and inner codes, respectively. In that Chapter we considered concatenation with overall rate equal to 1/3, 5/16 and 1/4. Unfortunately, for the first two rates results were disappointing. For the rate 1/4 some improvements in the performance are available, so

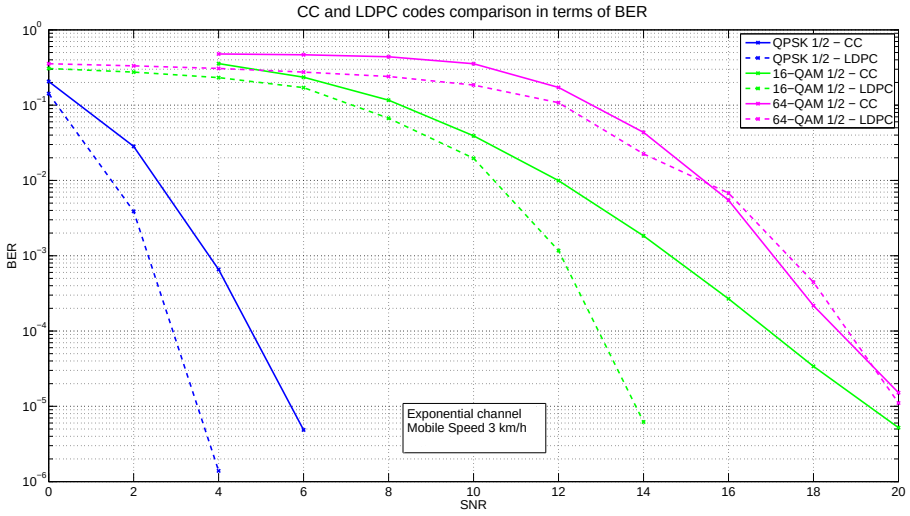

Figure 8.6: Comparison in terms of bit error rate between the convolutional code and the LDPC code when the mobile station moves at 3 km/h.

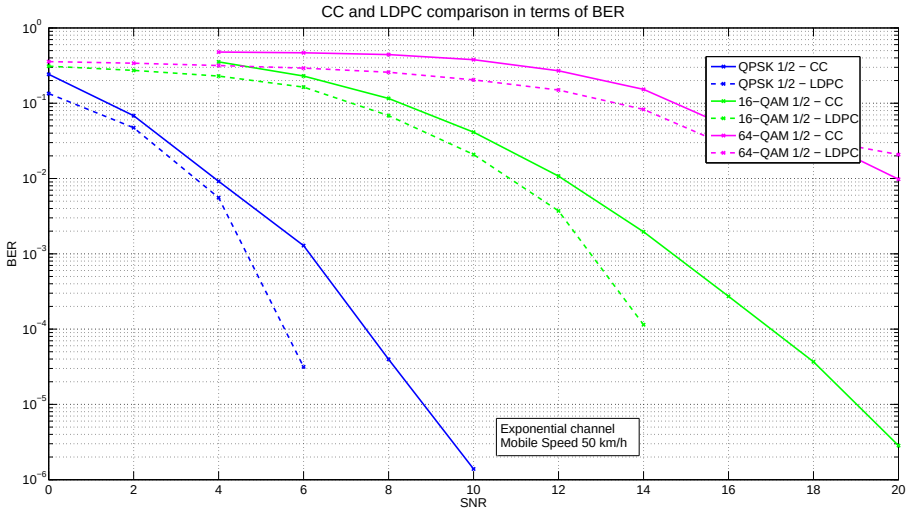

Figure 8.7: Comparison in terms of bit error rate between the convolutional code and the LDPC code when the mobile station moves at 50 km/h.

we decided to apply this kind of analysis to the WiMAX system. Considering the slots concatenation rule defined in Table 7.8, we introduced in the system, for the QPSK modulation other three types of coding: a regular LDPC code with rate 3/4, a turbo code with rate [1] 1/3 and the concatenation of these two codes for an overall rate equal to 1/4.

[1] The turbo code used here is not the one defined in the WiMAX standard, but the one described in Chapter 6, because we want to compare the performance of the proposed algorithm with a "common" turbo code, i.e. a code made by two identical convolutional codes.

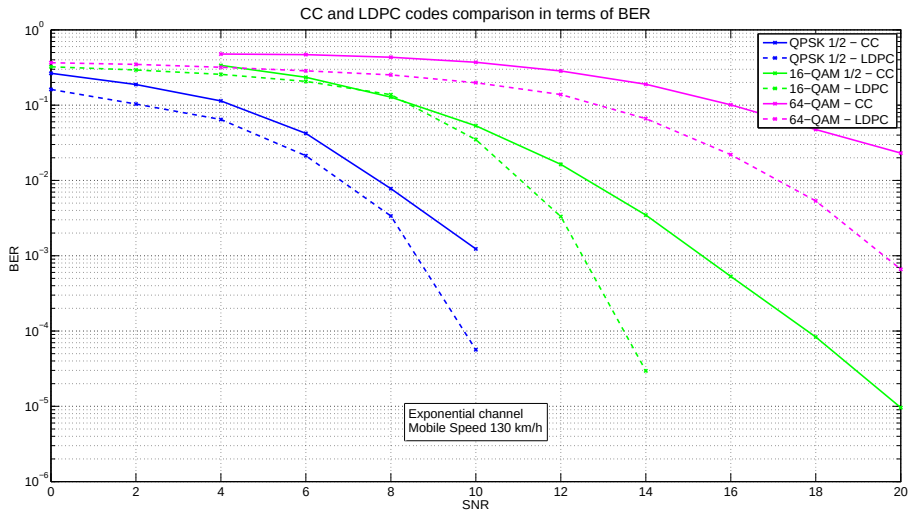

Figure 8.8: Comparison in terms of bit error rate between the convolutional code and the LDPC code when the mobile station moves at 130 km/h.

The parity-check matrix for the regular LDPC code has been found as described in Section 6.2.2, while the turbo code is the one defined in the 3GPP standard [49] and described in Section 6.2.1.

Figures 8.9 and 8.10 represent the performance of the system with a QPSK modulation, an AWGN channel and the coding types described above. In Figure 8.9 the performance is represented in terms of SNR, just to show the results when we consider the same level of signal-to-noise ratio. This is not a good term of comparison because the energy used for each information bit changes when we consider different coding rates. To avoid this problem we represent the same BER in terms of E_b/N_0 in Figure 8.10; the conversion follows the equation:

$$\frac{E_b}{N_0} = \frac{SNR}{R \log_2(M)}$$

where R is the coding rate and M the cardinality of the modulation alphabet. From this figure we can see that after 4 dB of E_b/N_0 the concatenation scheme outperforms the convolutional one. Unfortunately, the computational complexity in the concatenation encoding and decoding is higher than in the CC case. Thus, the choice of the coding type must be made considering a trade-off between the computational complexity and the required BER.

8.4 Finite computational resource

In this Section we present the results when we consider channel estimation and tracking. We perform a LS estimation on the preamble and use this as the starting point for the tracking of the channel over the 24 OFDM symbols. The tracking is

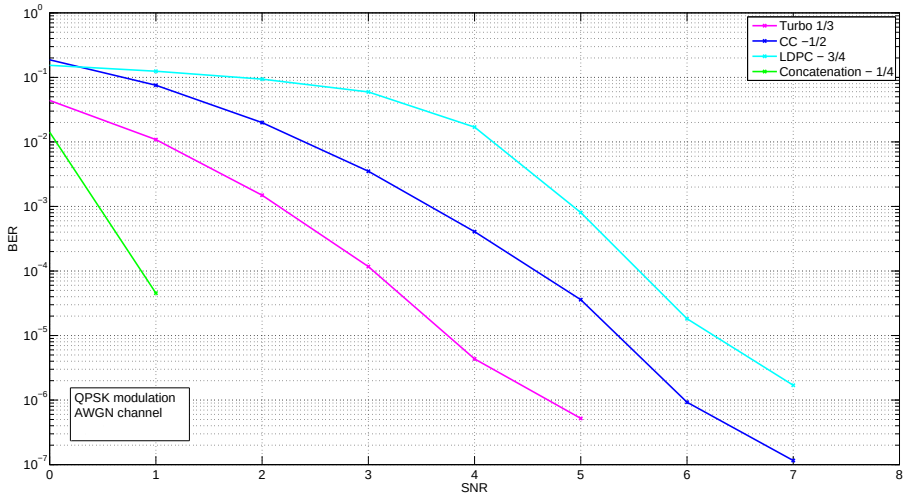

Figure 8.9: Performance in terms of bit error between CC and concatenation. For a more complete analysis also the performance of the constituent codes (Turbo and LDPC) are represented.

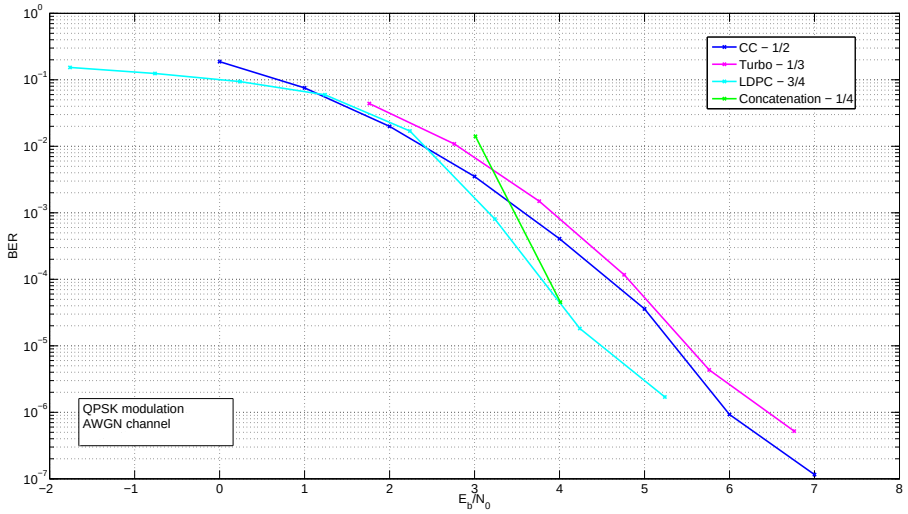

Figure 8.10: Comparison in terms of bit error rate between the performance of convolutional, turbo and LDPC codes and concatenation of turbo and LDPC. Here we represent the performance presented in Figure 8.9 in terms of E_b/N_0.

performed through a LS estimation over the pilots.

Figures 8.11, 8.12 and 8.13 represent the performance when we consider perfect channel knowledge ($\times$ marker) and channel estimation ($\circ$ marker). The channel used is the exponential channel with Rayleigh fading and the mobile speed is 3, 50 and 130 km/h. The QPSK 1/2 is represented with blue lines, the QPSK 3/4 with red lines, the 16−QAM 1/2 with green lines, the 16−QAM 3/4 with cyan lines, the 64−QAM 1/2

with magenta lines and finally the 64−QAM 2/3 and 3/4 with blue and yellow lines.

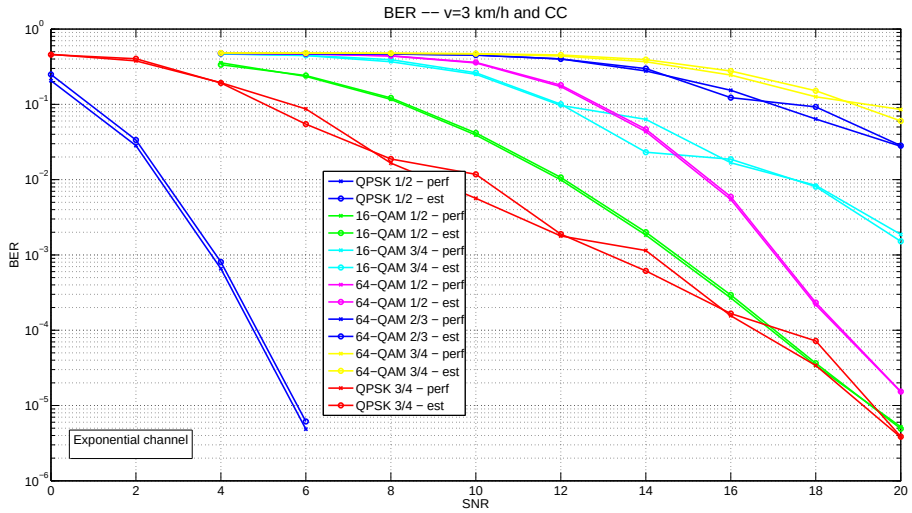

Figure 8.11: Comparison in terms of bit error rate between different types of modulation and CC coding rates, with perfect channel knowledge and estimation.

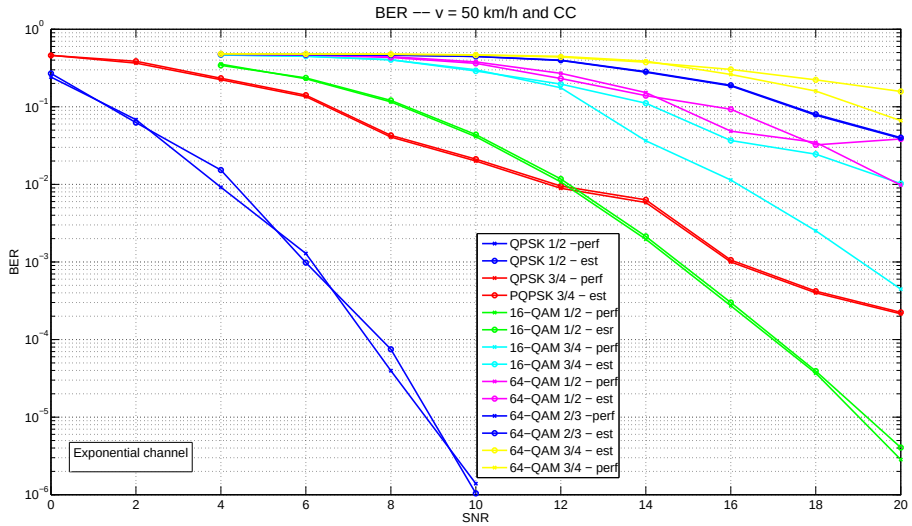

Figure 8.12: Comparison in terms of bit error rate between different types of modulation and CC coding rates, with perfect channel knowledge and estimation.

Figures 8.14, 8.15 and 8.16 represent the performance for three different modulations: QPSK, 16−QAM and 64−QAM, respectively. The coding rate is equal to 1/2 and we consider LDPC (dotted lines) and convolutional (solid lines) codes. For all the modulation schemes we assume that the exponential channel is affected by Doppler frequency due to three different speeds: 3 (blue lines), 50 (green lines) and 130 (magenta lines) km/h. For all the cases we consider perfect channel knowledge (× marker) and

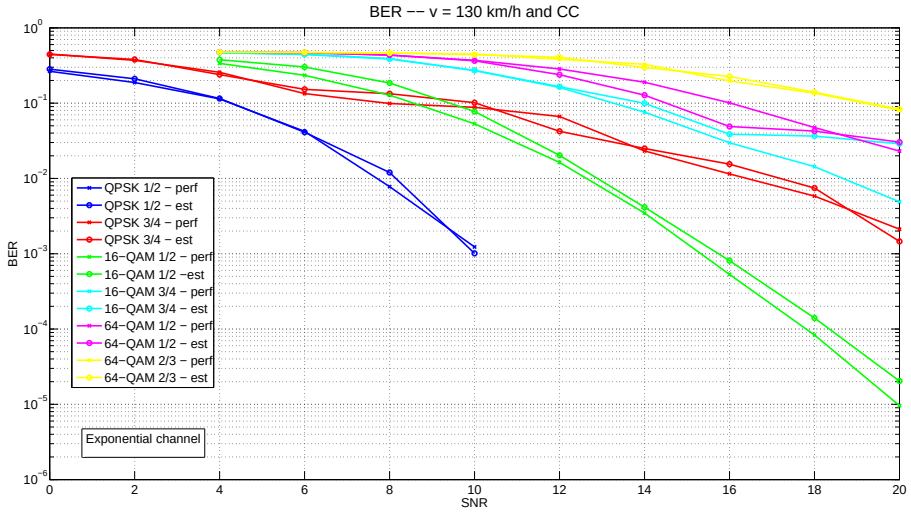

Figure 8.13: Comparison in terms of bit error rate between different types of modulation and CC coding rates, with perfect channel knowledge and estimation.

estimation (∘ marker).

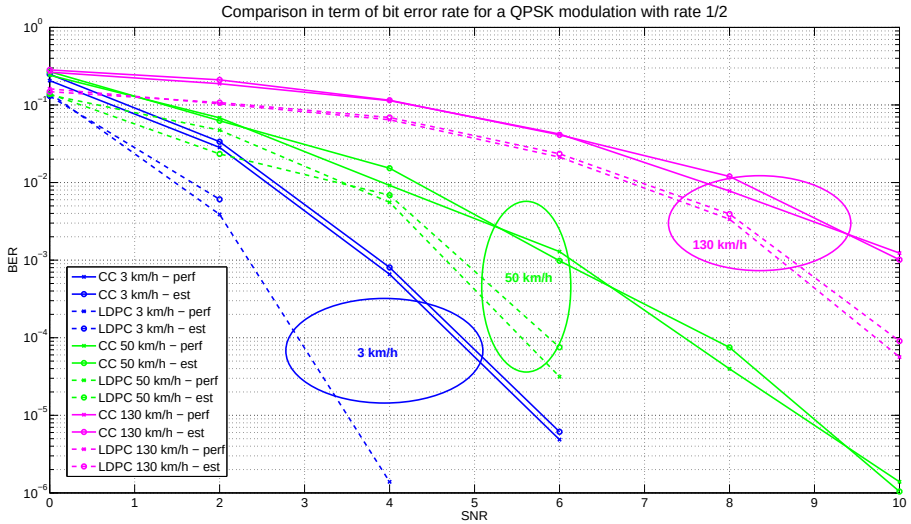

Figure 8.14: Comparison in terms of bit error rate for a QPSK modulation and a coding rate equal to 1/2. Results for different mobile speeds are represented.

Finally, we implemented our concatenation scheme for the channel coding in the WiMAX system. In Figures 8.17, 8.18 and 8.19 we represent performance when we consider a WiMAX system with QPSK modulation and exponential channel for different mobile speeds: 3, 50 and 130 km/h, respectively. The channel coding is implemented with the proposed concatenation scheme with overall rate equal to 1/4. Performance for the proposed algorithm are very good, in fact the BER is about 10^{-7} for 1.4 dB of

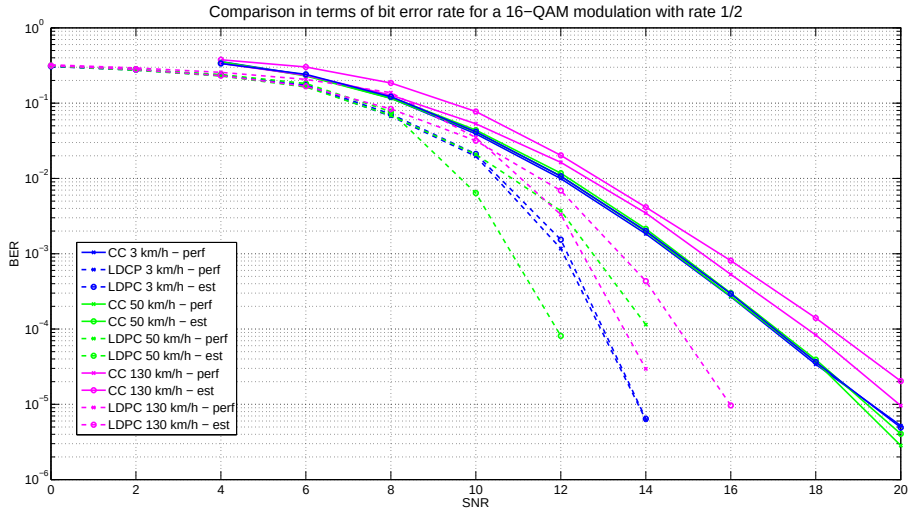

Figure 8.15: Comparison in terms of bit error rate for a QPSK modulation and a coding rate equal to 1/2. Results for different mobile speeds are represented..

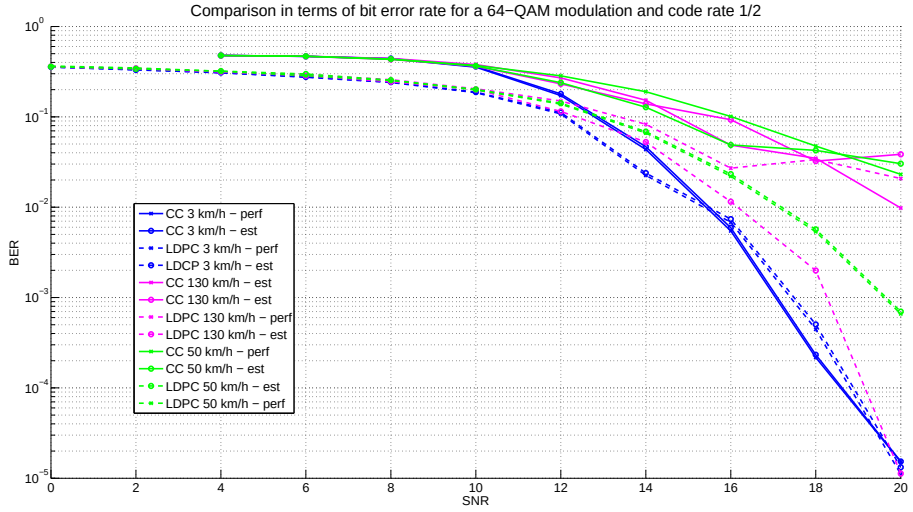

Figure 8.16: Comparison in terms of bit error rate for a QPSK modulation and a coding rate equal to 1/2. Results for different mobile speeds are represented.

SNR. For all the Figures, the channel estimation is computed through the LS algorithm, as justified above.

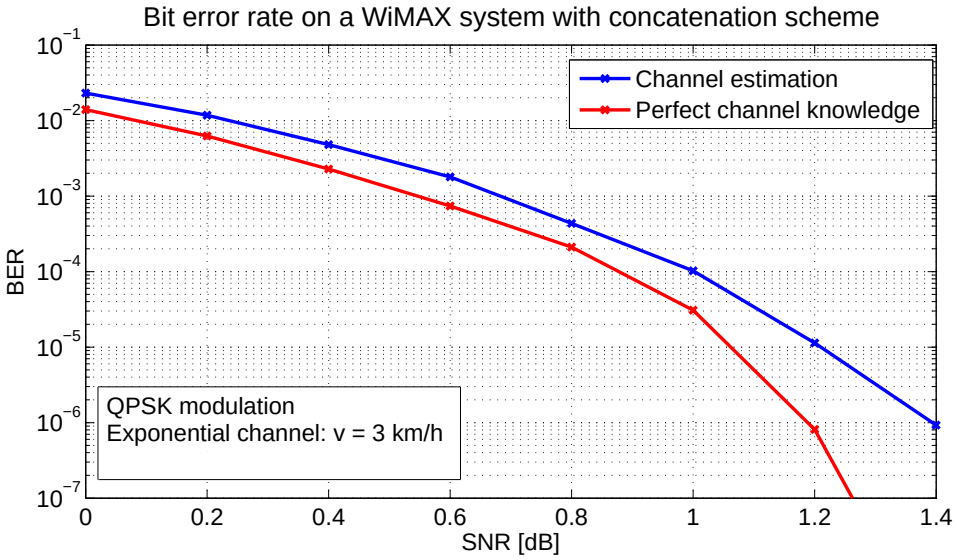

Figure 8.17: Performance, in terms of bit error rate, for a WiMAX system with QPSK modulation, exponential channel (mobile speed equal to 3 km/h) and concatenation scheme.

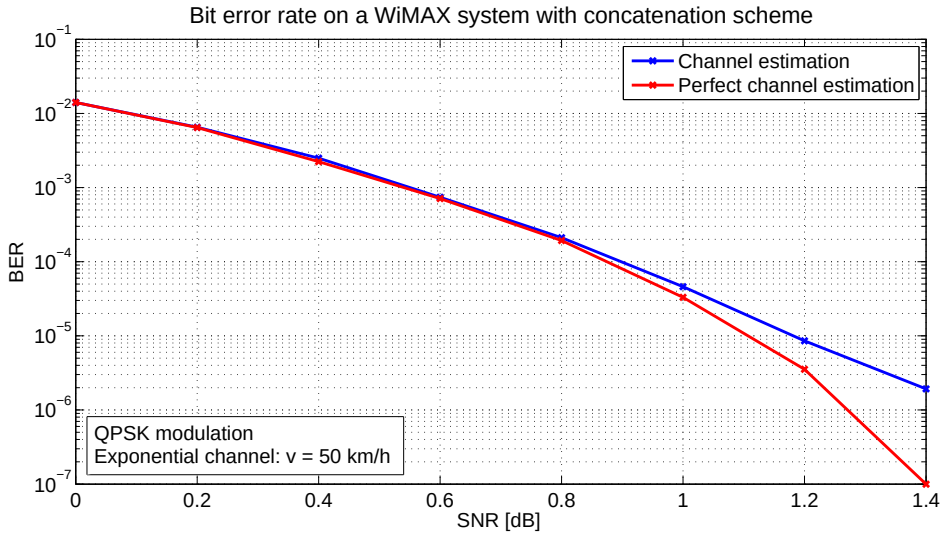

Figure 8.18: Performance, in terms of bit error rate, for a WiMAX system with QPSK modulation, exponential channel (mobile speed equal to 50 km/h) and concatenation scheme.

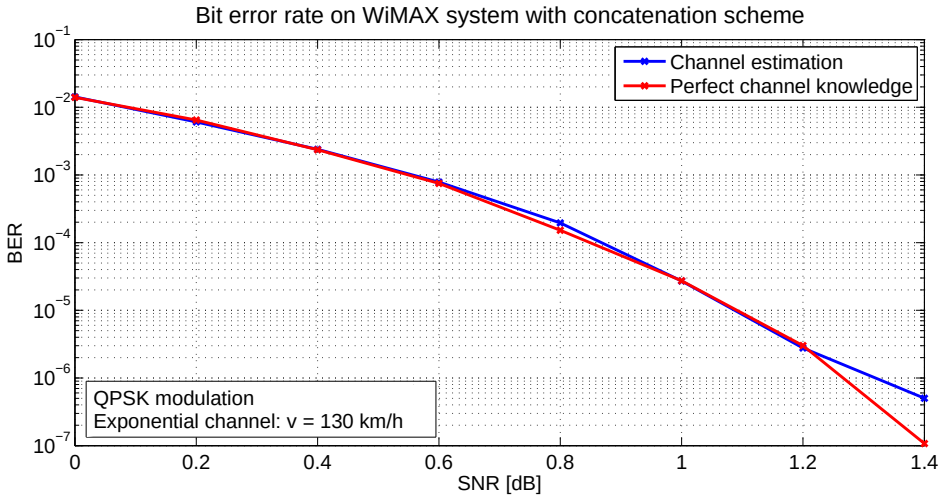

Figure 8.19: Performance, in terms of bit error rate, for a WiMAX system with QPSK modulation, exponential channel (mobile speed equal to 130 km/h) and concatenation scheme.

Chapter 9

Conclusions

Final aim of this work is the performance of a WiMAX system. Actually two big problems for wireless systems have been analyzed and studied in a general environment and then the conclusions of these studies have been applied to a WiMAX system.

The first studied problem is the channel estimation in an OFDM system.

We suppose the use of a comb-type structure, which means that in an OFDM symbol some fixed subcarriers are dedicated to the transmission of pilots.

After the study of the channel estimation we can find in literature, in the time and frequency domain, we implemented a quasi-compliant WiMAX standard simulator and analyzed the performance of different channel estimation in the frequency domain. For quasi-compliant WiMAX standard simulator we mean a simulator with the parameters given in the standards [1] and [2], but where we do not implement the subcarriers permutation and all the code-modulation combinations. We simply consider a system coded with a convolutional code 1/2 and modulated QPSK.

The choice of considering channel estimation in frequency domain derives from the considered system: it is the most obvious choice for an OFDM system.

The estimation techniques considered are Least Square (LS) and Linear Minimum Mean Square Error (LMMSE) and finally an adaptive technique. Since we want to perform the estimation of the channel over a frame of 24 OFDM data symbols and a preamble, we assume the channel is slow-time variant, which means it is constant on an OFDM symbol and varies very slowly from a symbol to the successive one. The former two techniques can be used both for the preamble estimation and the channel tracking through the pilots, whereas the last one, explained in Section 2.4 takes, as initial estimation, the LS or the LMMSE preamble estimation and simply performs the tracking through the pilots.

The channel used in the simulations is the SCME [110] and, since the system considered is a mobile one, we assume the mobile station moves with three different speeds: 3, 50 and 130 km/h. About the LMMSE, two different versions could be used, one is independent of the transmitted data and the other one depends on them.

From results presented in Chapter 2 we can see that the LS and LMMSE dependent on transmitted data are both good estimators, even if the LMMSE one is computational

more complex than the LS one. The adaptive estimator performs well for speeds under 50 km/h, but fails for speed equal to 130 km/h. Thus for slow mobile speeds we can choose between LS or adaptive estimator, and rather the adaptive one can be chosen, because it is better from a computational point of view. For high speeds the LS estimator seems to be the best choice. These conclusions are straightly related to a system with the WiMAX parameters; a different number of pilots can completely change the results.

For the problem of the channel estimation another solution has been explored: a joint channel estimation and data detection through the expectation maximization algorithm can be performed.

This solution is very complex and so it makes sense to use it only if the bit error rate gain, with respect to the performance of the techniques described above, is significant. Unfortunately, results presented in Chapter 3 do not show any significant gain and so this is not the solution we are looking for.

The second big problem considered in this work is the channel coding with particular attention to the Low-Density Parity-Check (LDPC) codes. These codes have drawn much attention due to their near-capacity error correction performance. They are valid competitors of the turbo codes, because, even if they don't have as good performance as turbo codes for low SNR, they present lower error floor at high SNR and their decoding is not trellis based. In fact, their decoding is performed through an Iterative Decoding based on Belief Propagation (IDBP) algorithm which is a symbol-by-symbol soft-in soft-out decoding algorithm which iteratively processes the received symbol in order to improve its reliability.

The computation complexity of this algorithm is, however, high and so a modified version has been proposed. Results presented in Chapter 5 are quite promising, even if a method to handle the not-a-number cases is needed.

The best known codes, i.e. LDPC and turbo, have good performance if the codewords are very long, but, in our days, in many applications, such as voice, delay is an important issue and large block sizes commonly used for these codes cannot be applied.

The solution we proposed to this problem, to obtain better performance than the one given by the known block codes, as Hamming, or convolutional codes, is the concatenation of short LDPC and turbo codes as outer and inner codes, respectively. The turbo codes, due to their structures, are not able to correct some error sequences, so, if we construct the LDPC code such that the possible wrong bits are checked by different check nodes, the performance of the concatenation should be good.

Results presented in Chapter 6 show that the choice of the overall rate and of the rates of the constituent codes is very important. First of all the loss due to the puncturing of the turbo code cannot be recovered with the LDPC code. Thus, the overall code must be chosen such that turbo rate is 1/3. On the other hand it is well known that performance of LDPC codes with rates external to the interval [1/3, 1/2] are not so good. By the way, the overall rate equal to 1/4 shows good results even if this is a very low rate and this means we add a lot of redundancy and so we introduce some delay. This is not a good point, since we started investigating this concatenation scheme because we want to use it in systems with tight constraints on delay.

Finally, we implemented a WiMAX standard-compliant simulator for the transmission in downlink.

We assume synchronization and apply the presented channel estimation and coding.

Results presented in Chapter 8 show the loss of performance due to the mobile speed and the fact that LS estimator over the pilots gives very good results for speeds of 3 and 50 km/h, but experiences some loss at 130 km/h, especially for 64−QAM modulation.

For the channel coding we find out that, if we use the standard-defined LDPC code with the slots concatenation rule rather than the subchannels one, there are some gains in terms of bit error rate.

About the proposed concatenation scheme, results show some performance gains when the channel is AWGN, but these gains could be lost if the channel experiences any kind of fading, for example Rayleigh fading.

Observing the performance of the WiMAX system, we can conclude saying that the reduced rank LS estimator is a good technique for the channel estimation and LDPC codes could be used in place of the convolutional code to obtain better performance.

Appendix A

Convergence theorem for EM algorithm

Let

$$k(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta}) = \frac{f(\boldsymbol{x}|\boldsymbol{\theta})}{g(\boldsymbol{y}|\boldsymbol{\theta})}$$

and note that $k(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta})$ may be interpreted as a conditional density. Then the log-likelihood function $L_y(\boldsymbol{\theta}) = \log g(\boldsymbol{y}|\boldsymbol{\theta})$ may be written as

$$L_y(\boldsymbol{\theta}) = \log f(\boldsymbol{x}|\boldsymbol{\theta}) - \log k(\boldsymbol{y}|\boldsymbol{\theta}).$$

Define

$$H(\boldsymbol{\theta'}|\boldsymbol{\theta}) = E[\log k(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta'})|\boldsymbol{y},\boldsymbol{\theta}].$$

Let $M : \boldsymbol{\theta}^{[k]} \rightarrow \boldsymbol{\theta}^{[k+1]}$ represent the mapping defined by the EM algorithm in the following Equations:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{[k]}) = E[\log f(\boldsymbol{x}|\boldsymbol{\theta})|\boldsymbol{y},\boldsymbol{\theta}^{[k]}],$$

$$\boldsymbol{\theta}^{[k+1]} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{[k]}),$$

so that $\boldsymbol{\theta}^{[k+1]} = M(\boldsymbol{\theta}^{[k]})$.

Theorem 2. *$L_y(M(\boldsymbol{\theta}^{[k]})) \geq L_y(\boldsymbol{\theta})$, with equality if and only if*

$$Q(M(\boldsymbol{\theta})|\boldsymbol{\theta}) = Q(\boldsymbol{\theta}|\boldsymbol{\theta})$$

and

$$k(\boldsymbol{x}|\boldsymbol{y},M(\boldsymbol{\theta})) = k(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta}).$$

That is, the likelihood function increases at each iteration of the EM algorithm, until the conditions for equality are satisfied and a fixed point of the iteration is reached. A proof of the theorem may be found in [51]. In $\boldsymbol{\theta}^*$ is an ML parameter estimate, so that $L_y(\boldsymbol{\theta}^*) \geq L_y(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \Theta$, then $L_y(M(\boldsymbol{\theta}^*)) \geq L_y(\boldsymbol{\theta}^*)$. In other words, ML estimates are fixed points of the EM algorithm. Since the likelihood function is bounded (for distribution of practical interest), the sequence of parameter estimates

$\boldsymbol{\theta}^*[0], \boldsymbol{\theta}^*[1], \ldots, \boldsymbol{\theta}^*[k]$ yields a bounded non-decreasing sequence $L_y(\boldsymbol{\theta}^{[0]}) \leq L_y(\boldsymbol{\theta}^{[1]}) \leq \cdots \leq L_y(\boldsymbol{\theta}^{[k]})$ which must converge as $k \to \infty$.

The theorem falls short of proving that the fixed points of the EM algorithm are in fact ML estimates. The latter in true, under rather general conditions, but the proof is somewhat involved and is not presented here.

Appendix B

Channel length estimation

B.1 Introduction

As said in Chapter 2, the performance naturally depends on the accuracy of the channel estimates. Conventional channel estimation algorithms preset a fixed length for the channel; in Chapter 2, we assumed to know the exact value of the channel length impulse response. For wireless communications, however, the actual length of the channel is environment-dependent. Therefore, unless the receiver has explicit knowledge of the propagation environment, the actual length of the wireless channel is unknown. The effect of mismatch between the length of the channel estimator and actual length of the discrete-time equivalent channel has also been investigated in the literature [112]. It has been shown that an irreducible BER floor results when the channel estimator length is shorter than the actual channel [112]. On the other hand, if the channel estimator length is longer than the actual channel, the equalizer becomes unnecessarily complex. Furthermore, a performance loss that increases with the channel estimator length is still incurred [112].

In [113], the authors proposed a least squares approach for joint synchronization, channel length estimation and channel estimation. In fact, it is desirable to modify conventional channel estimation algorithms so that the length of the channel estimator adapts to the actual channel. The method proposed in [113] relies on the minimization of the mean square error of the estimated channel coefficients for different predetermined CIR lengths. To apply this method, the channel window (the difference between the maximal and the minimal CIR length) must be known. Simulation results in [113] show that the proposed scheme is very robust against variations of the propagation environment and achieves the best trade-off between performance and complexity.

In the literature, there are some other available methods for estimating the Channel Impulse Response (CIR) length, e.g., [114], [115] and [116]. The method described in [114] estimates the CIR length by using the estimated SNR. The method in [115] rests on the generalized Akaike information criterion. It was shown there that the CIR length is usually underestimated. In [116], the estimation of the CIR length is based on a given factor R which is defined by the ratio of the channel power to the total power

of the channel plus noise. The ratio R is defined in [116] as a constant factor within the interval $[0.9, 0.95]$. Since the noise power and the channel power are unknown, such an estimation of the CIR length based on a predefined ratio R is not a precise solution.

In [117], to overcome the difficulties in the CIR length estimation of time-variant channels in the presence of strong additive noise, an auxiliary function is established to distinguish between the statistical characteristics of the additive noise and those of the multipath channel. The difference between the statistical characteristics of the additive noise and the multipath channel is that the CIR coefficients are only located in the window of the CIR length, whereas the additive noise per channel tap is uniformly distributed over the whole length of the estimated CIR. The proposed algorithm provides reliable information on the estimated CIR length even at low SNRs. The estimated CIR length and noise variance are very useful in many areas, such as channel estimation, channel coding, data equalization, adaptive filter implementation, and OFDM systems with adaptive guard interval length.

B.2 Algorithm for the noise variance and the CIR length estimation

The considered algorithm combines two different challenges (CIR length and noise variance estimation) to one task. To implement it, we assume that the channel is already estimated by a conventional method [118]. Even though the channel is estimated, the CIR length remains unknown. This is due to the fact that the estimated CIR is usually impaired by additive noise. However, a good estimate of both the CIR length and the SNR can be obtained by applying the proposed algorithm.

We consider the estimated channel coefficient $\hat{h}_{k,i}$ corresponding to the i^{th} OFDM symbol and the k^{th} channel tap. Let us assume that the distance between two neighboring channel tap gains is equidistant and equal to the sampling interval t_a of the system, then the relationship between the channel tap index k and the corresponding propagation delay τ_k is given by $\tau_k = k \cdot t_a$. The estimated channel coefficient $\hat{h}_{k,i}$ is the sum of the true channel coefficient $h_{k,i}$ and a zero-mean noise component $n_{k,i}$ [118], i.e.,

$$\hat{h}_{k,i} = h_{k,i} + n_{k,i}. \tag{B.1}$$

It is worth mentioning that the noise variance to be estimated is the variance of the noise component $n_{k,i}$ of the estimated channel coefficient $\hat{h}_{k,i}$

$$\sigma_n^2[k] = E[|n_{k,i}|2] \tag{B.2}$$

where $E[|n_{k,i}|2]$ is the expectation value of $|n_{k,i}|2$ over the OFDM symbol index i. The symbol $\sigma_n^2[k]$ denotes the variance of the noise component of the k^{th} estimated channel tap.

As mentioned above, the estimated channel coefficient contains two different compo-

nents. The first component is the true channel coefficient, which is located only in the window of the CIR length. The second component is the additive noise component $n_{k,i}$, which is uniformly distributed over the whole length of the estimated CIR. Therefore, the channel tap index k is omitted in the symbol representing the noise variance, i.e., $\sigma_n^2[k]$ is replaced by σ_n^2. Using the estimated noise variance σ_n^2 and the signal power, the estimated SNR can easily be calculated [118]. If L is presumed to be the true CIR length, then the new estimated channel coefficients $\tilde{h}_{k,i}^L$, $k = 0, 1, \ldots, N_k - 1$, are equal to the first L samples of the estimated channel coefficients $\hat{h}_{k,i}$ and zero elsewhere, i.e.,
.

$$\tilde{h}_{k,i}^L = \begin{cases} \hat{h}_{k,i}, & 0 \le k < L \\ 0, & L \le k \le N_K - 1. \end{cases} \tag{B.3}$$

The presumed CIR length L is in the range $0 < L \le N_K - 1$, since the true CIR length must be larger than zero and is assumed to be smaller than the estimated CIR length. The mean squared error between $\tilde{h}_{k,i}^L$ and $\hat{h}_{k,i}$ is

$$\begin{aligned} e(L) &= E[\sum_{k=0}^{N_K-1} |\hat{h}_{k,i} - \tilde{h}_{k,i}^L|^2] \\ &= E[sum_{k=L}^{N_K-1} |\hat{h}_{k,i}|^2]. \end{aligned} \tag{B.4}$$

Thus, the mean squared error $e(L)$ is the sum of the average squared magnitude values of the estimated channel taps from the L^{th} channel tap to the last channel tap $\hat{h}_{N_K-1,i}$. Substituting $\hat{h}_{k,i}$ from (B.1) into (B.4), it follows that

$$\begin{aligned} e(L) &= E[sum_{k=L}^{N_K-1} |h_{k,i} + n_{k,i}|^2] \\ &= \sum_{k=L}^{N_k-1} \left\{ E[|h_{k,i}|^2] + E[|n_{k,i}|^2] \right\} \\ &= \sum_{k=L}^{N_K-1} \rho_k + (N_K - L)\sigma_n^2 \end{aligned} \tag{B.5}$$

where $\rho_k = E[|h_{k,i}|^2]$ is the average power of the k^{th} path. In (B.5), let $e_1(L) = \sum_{k=L}^{N_K-1} \rho_k$ be the first term and $e_2(L) = (N_K - L)\sigma_n^2$ be the second term of the mean squared error $e(L)$, it can be seen that $e_1(L)$ stems completely from the channel, whereas $e_2(L)$ originates from the noise components. The noise-related term $e_2(L)$ can be compensated by adding an appropriate variance to this function. According to the expression of $e_2(L)$, the appropriate variance should be equal to $L\sigma_{pre}^2$, where σ_{pre}^2 is called the presumed noise variance. The compensation of the noise-related term $e_2(L)$ establishes the auxiliary function

$$f(L) = \sum_{k=L}^{N_K-1} \rho_k + (N_K - L)\sigma_n^2 + L\sigma_{pre}^2. \tag{B.6}$$

The characteristics of the auxiliary function $f(L)$ presented in Figure B.1 depend on the following cases determined by the presumed noise variance:

a) If the presumed noise variance is larger than the true noise variance, i.e., $\sigma^2_{pre} > \sigma^2_n$, then there exists always a unique minimum value of the auxiliary function $f(L_{f,min}) = min(f(L))$, where $L_{f,min} \leq N_P$. If σ^2_{pre} is close to σ^2_n, then $L_{f,min}$ is also close to N_P.

b) If the presumed noise variance is exactly equal to the true noise variance, i.e., $\sigma^2_{pre} = \sigma^2_n$, then $f(L)$ becomes

$$f(L) = \sum_{k=L}^{N_K-1} \rho_k + N_K\sigma^2_n. \tag{B.7}$$

In this case, the auxiliary function $f(L)$ is a monotonously decreasing function within the true CIR length, and is equal to $N_K\sigma^2_n$ outside the true CIR length.

c) If the presumed noise variance is smaller than the true noise variance, i.e., $\sigma^2_{pre} < \sigma^2_n$, then $f(L)$ is a monotonously decreasing function within the whole length of the estimated CIR, and reaches the minimum value at $L = N_K - 1$.

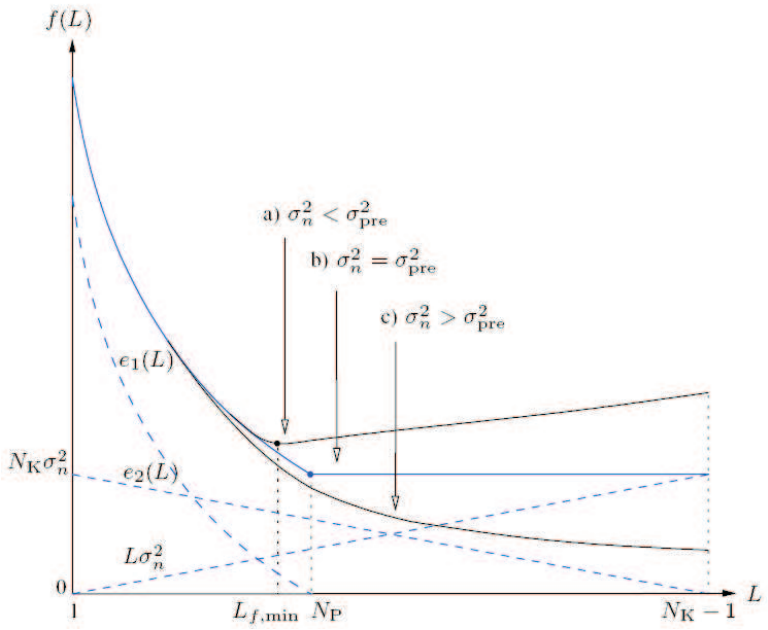

Figure B.1: The auxiliary function $f(L)$ for different cases of the presumed noise variance σ^2_{pre}.

Based on the characteristics of the auxiliary function $f(L)$, an algorithm called the noise variance and CIR length estimation (NCLE) is proposed in [117]. If the presumed noise variance σ^2_{pre} is reduced step by step from the possible maximum value to the possible minimum value of the true noise variance, then the curve of $f(L)$ will be

changed from case a) to case c). Each step is considered as one iteration towards the reduction of the presumed noise variance. The amount $\Delta\sigma^2$, which is used to reduce the presumed noise variance σ^2_{pre} in each iteration, is called the step size. If this step size is very small in comparison to the true noise variance, then case b) might appear. Otherwise, case a) skips directly over to case c). When the situation in case c) appears for the first time, then the presumed noise variance of the previous iteration is very close to the true noise variance and the decision of the estimated noise variance will be made. The shape of $f(L)$ at the previous iteration corresponds of course either to case a) or to case b).

If case a) appears, then the estimated CIR length $\hat{N}_P$ is assigned to be $L_{f,min}$, where $f(L_{f,min}) = \min(f(L))$. As explained in case a), the estimated CIR length is shorter than or equal to the true CIR length.

Case b) might appear, if the presumed noise variance is very close to the true noise variance. Since the theoretical auxiliary function $f(L)$ in case b) is constant over the range from $L = N_P$ to $L = N_K - 1$ (see Figure B.1), it follows that $f(L)$ does not have a unique minimum value like in case a). However, if the minimum value of the auxiliary function $f(L)$ is still computed by a numerical method, then a minimum value can be found. This is due to the fact that the realized auxiliary function is practically not constant in the interval mentioned above. In this case, the value of L corresponding to the minimum value of $f(L)$, i.e., $L_{f,min}$, is always larger than or equal to the true CIR length. The estimated CIR length can be assigned to this value, and, thus, it is also larger than or equal to the true CIR length. To ensure that the estimated CIR length is close to the true CIR length, the procedure of establishing the auxiliary function $f(L)$ and seeking its minimum value should be repeated N_E times. A single execution of this procedure is called an experiment. The estimated CIR length of each experiment is stored in a vector $\vec{\mathcal{L}}$. Analogously, the estimated noise variances are stored in a vector denoted by $\vec{\mathcal{N}}$. After N_E experiments, the final result of the estimated CIR length is the minimum element of the vector $\vec{\mathcal{L}}$. The final estimated noise variance is the average value of the elements of the vector $\vec{\mathcal{N}}$.

The initial value of the presumed noise variance can be determined by

$$\sigma^2_{pre} = E[\sum_{k=1}^{N_k-1} |\hat{h}(k)|^2/(N_k - 1)]. \tag{B.8}$$

Figure B.2 shows a flowchart of the NCLE algorithm.

The application of the algorithm to our system shows the same results presented in [117].

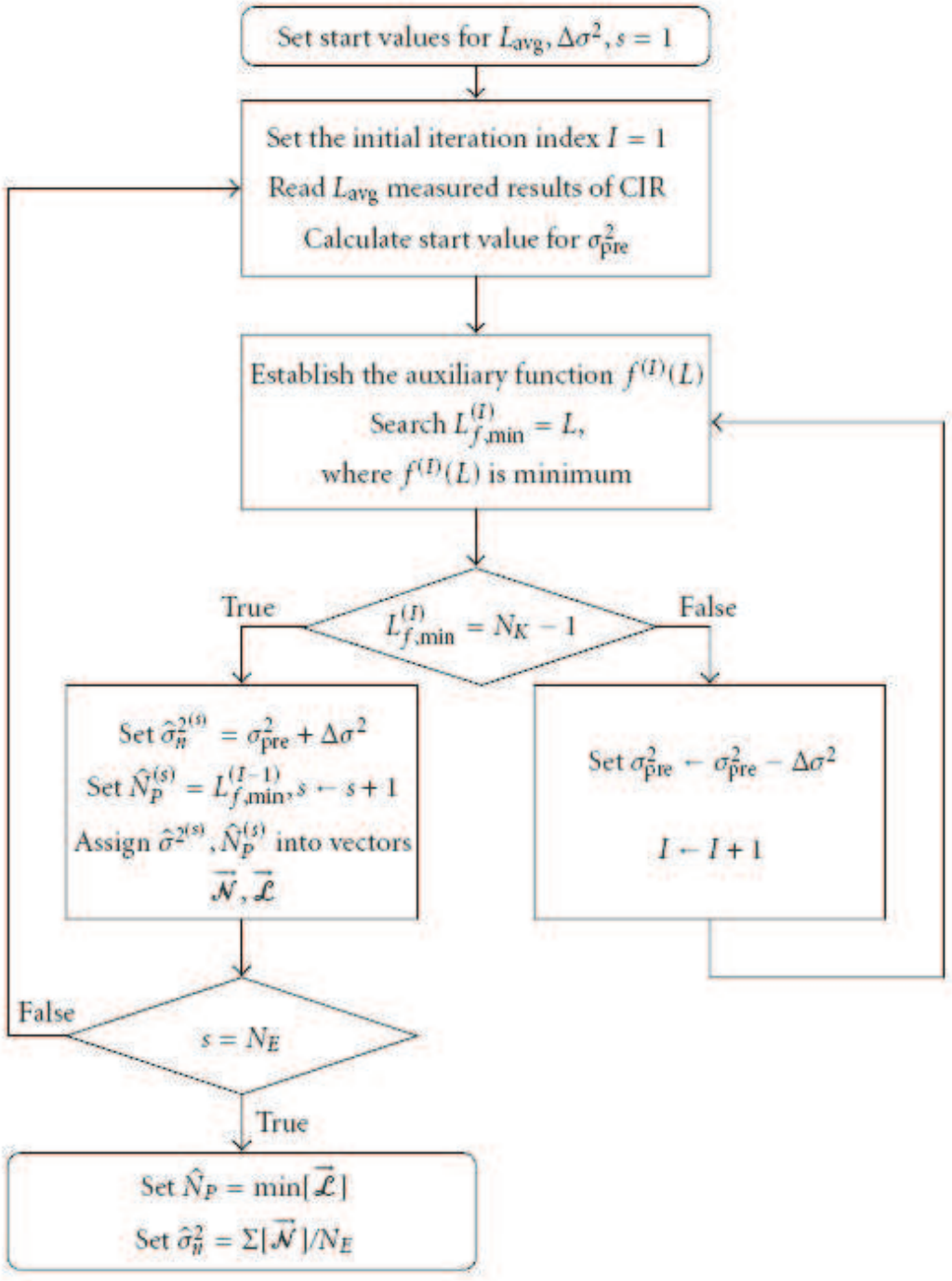

Figure B.2: Flowchart of the NCLE algorithm.

Appendix C

Optional subchannel allocations in WiMAX systems

C.1 PUSC variations

C.1.1 Potentially optional allocations

First we remember that the allocations defined potentially optional are those which are not explicitly mentioned as mandatory in the standard [2], but reading it they appear so, hence they are defined optional. Therefor in [109] it is required to implement some of them to have a certified product.

The potentially optional allocation are PUSC with all subchannels and PUSC with dedicated pilots, Full Usage of Subchannel (FUSC) and FUSC with dedicated pilots; all these except the last one are required to implement.

PUSC with all subchannels and PUSC with dedicated pilots

Before defining these allocations we must explain what is a DL_MAP. It is a MAC message that define burst start times for both time division multiplex and time division multiple access by a SS on the downlink. It includes the following parameters:

- PHY synchronization
- Downlink Channel Descriptor (DCD) count
- Base station ID

The DL_MAP_IEs describe the location, coding and modulation schemes of the bursts.

In the DL_MAP a BS may transmit the Downlink Interval Usage Code (DIUC) to 15 and this indicates that the subsequent allocation shall use a specific permutation and/or a specific transmit diversity mode.

Table 279 at page 377 of [2] shows the format of the information element for the OFDMA downlink. There are two important fields:

1. *Space Time Coding* (STC) which indicates the STC mode that will be used by the transmitter for the allocations

2. Dedicated Pilots; this is an optional field and is used to support the use of open loop precoding and closed loop transmission in which the MS has no knowledge of the precoding/beamforming matrix.

   When the bit of the dedicated pilots is set to one it means that the pilot symbols are precoded/beamformed in the same way as are the corresponding data subcarriers. In this case an MS should use only the pilots that are specific to its allocation for channel estimations.

C.2 Other allocation methods

C.2.1 Full Usage of Subchannel (FUSC)

Also in this case, as in the one of PUSC, the symbol structure is constructed using pilots, data and zero subcarriers. The symbol is first allocated with the appropriate pilots and zero subcarriers and, then, all the remaining subcarriers are used as data subcarriers (these will be divided into subchannels).

Each segment uses both sets of variable/constant pilot-sets.

The parameters that characterize the FUSC symbol structure and that depend on FFT size are:

- **Number of DC subcarriers**; its index is given by $N_{FFT}/2$, counting from 0)
- **Number of guard subcarriers, Left**, i.e. on the left size of the spectrum
- **Number of guard subcarriers, Right**, i.e. on the right side of the spectrum
- **Number of used subcarriers**
- **Pilots**
- **VariableSet#**0
- **ConstantSet#**0
- **VariableSet#**1
- **ConstantSet#**1
- **Number of data subcarriers**
- **Number of data subcarriers per subchannel**
- **Number of subchannels**
- **Permutation base**

Parameter	FFT size			
	2048	1024	512	128
Number of DC subcarriers	1	1	1	1
Number of guard subcarriers, Left	173	87	43	11
Number of guard subcarriers, Right	172	86	42	10
Number of used subcarriers	1703	851	427	107
Pilots	–	–	–	4
VariableSet#0	71	12	18	5
ConstantSet#0	12	2	3	4
VariableSet#1	71	12	18	1
ConstantSet#1	12	2	3	0
Number of data subcarriers	1536	768	384	96
Number of data subcarriers per subchannel	48	48	48	48
Number of subchannels	32	16	8	2
PermBase				

Table C.1: Symbol parameters for the available FFT size (i.e. 2048, 1024, 512 and 128).

In Table C.1 the parameters for the FUSC symbol structure are placed.

The variable set of pilots embedded within the symbol of each segment shall obey the following rule:

$$PilotsLocation = VariableSet\#x + 6 \cdot (FUSC\_SymbolNumber \mod 2) \quad \text{(C.1)}$$

where $FUSC\_SymbolNumber$ counts the FUSC symbols used in the current zone starting from 0.

The pilots in the ConstantSet are decided with the following rule:

$$ConstantSet\#k \rightarrow PilotosLocation = 72 * (2 * n + k) + 9 \quad \text{(C.2)}$$

where $n = 0, 1, \ldots, Value - 1$ and $Value$ is the value found in Table C.1 in the rows dedicated to the Constant sets.

Each subchannel is composed of 48 subcarriers. The subchannel indexes are formulated using a Reed-Solomon series and are allocated out of the data subcarriers domain.

After mapping all pilots, the remainder of the used subcarriers are used to define the data subchannels. In fact the remaining subcarriers are partitioned into groups of contiguous subcarriers. Each subchannel consists of one subcarrier from each of these groups. The number of these groups is equal to the number of subcarriers per subchannel $N_{subcarriers}$ and the number of subcarriers in a group is equal to the number of subchannels $N_{subchannels}$. Hence, the number of data subchannel is equal to $N_{subcarriers} \cdot N_{subchannels}$

The exact partition into subchannels is according to the permutation formula:

$$\begin{aligned} subcarrier(k,s) &= N_{subchannels} \cdot n_k + \{p_s[n_k \mod N_{subchannels}] \\ &+ DL\_PermBase\} \mod N_{subchannels} \end{aligned} \tag{C.3}$$

where:

- $\mathbf{subcarrier(n,s)}$ is the subcarrier index of the subcarrier n in subchannel $s = 0, 1, \ldots, N_{subchannels} - 1$ is the index number of a subchannel
- $\mathbf{n_k} = (k + 13 \cdot s) \mod N_{subcarriers}$, where $k = 0, 1, \ldots, N_{subcarriers} - 1$is the subcarrier index in a subchannel
- $\mathbf{N_{subchannels}}$ is the number of subchannels
- $\mathbf{p_s[j]}$ is the series obtained by rotating basic permutation sequence cyclically to the left s times
- $\mathbf{N_{subcarriers}}$ is the number of data subcarriers allocated to a subchannel in an OFDMA symbol

The whole data tones in a symbol are partitioned into groups of contiguous data subcarriers. Each subchannel consists of one subcarrier from each of these groups. The number of subcarriers in a group is, then, equal to the number of subchannels, said N_s, which is determined by FFT size. The exact partitioning into subchannels is according to the DL permutation formula:

$$Carriers(s,m) = \begin{cases} N_s \times k + [s + P_{1,c_1}(k') + P_{2,c_2}(k')] & 0 < c_1,\, c_2 < N_s \\ N_s \times k + [s + P_{1,c_1}(k')] & c_1 \neq 0,\, c_2 = 0 \\ N_s \times k + [s + P_{2,c_2}(k')] & c_1 = 0,\, c_2 \neq 0 \\ N_s \times k + s & c_1 = 0,\, c_2 = 0 \end{cases}$$

where:

- $\mathbf{carriers(s,m)}$ subcarrier index of m-th subcarrier in subchannel s
- $\mathbf{k} = (m + s * 23) \mod 48$, $k' = k \mod (N_s - 1)$
- $\mathbf{m} = 0, \ldots, 47$ is the subcarrier in subchannel index
- $\mathbf{s} = 0, 1, \ldots, N_s - 1$ is the index number of a subchannel
- $\mathbf{p_{1,c_1}}$ is the j-th element of a sequence obtained by rotating basic permutation sequence P_1 cyclically to the left c_1 times
- $\mathbf{p_{2,c_2}}$ is the j-th element of a sequence obtained by rotating basic permutation sequence P_2 cyclically to the left c_2 times
- $\mathbf{c_1} = DL\_PermBase \mod N_s$

- $\mathbf{c_2} = \lfloor DL\_PermBase/N_s \rfloor$

FFT size	$\mathbf{N_s}$
2048	32
1024	16
512	8
128	2

Table C.2: Values of N_s for every FFT size.

C.2.2 Optional allocations

These are the allocations defined optional in the standard [2]. They are the optional FUSC and the O-FUSC with dedicated pilots the AMC (1×6, 2×3 and 3×2) and the same ones, but with dedicated pilots and the PUSC-ASCA. In [109] only the AMC 2×3 is required to be implemented.

Optional FUSC

The minimal block of processing is given by the 48 data carriers symbols. First, the pilot carriers are allocated and the remaining carriers are used exclusively for data transmission. The N_{used} subcarriers (except the DC carrier) are dived into groups of 9 contiguous subcarriers and in each of these one pilot is allocated. Its position varies according to the index of OFDMA symbol which contains the subcarriers. The equation is:

$$3 \times l + 1 \tag{C.4}$$

where $l = m \mod 3$ and m is the symbol index.

Hence, the number of pilot subcarriers is given by $(N_{used} - 1)/9$, because we need to exclude the DC carrier and their position is given by $9k + 3m + 1$, where $k = 0, 1, \ldots,$ *Number of pilot subcarriers* $- 1$ and $m = (\textit{symbol index}) \mod 3$.

Optional Adjacent Subcarrier Permutation (AMC)

Since now we have considered *"distributed subcarrier permutations"*, but a BS may change to the *"adjacent subcarrier permutation"*. This one can be used to take advantage of the structure of the adjacent subcarrier permutation in parts of DL subframe that are indicated accordingly by the DL-MAP an UL subframe that are indicated by the UL-MAP.

After this change a BS can only transmit/receive traffic using the adjacent subcarrier permutation during the allocated period and shall return to the distributed mode only at the beginning of a new DL subframe.

Symbol data within a subchannel is assigned to adjacent subcarriers and pilots and data subcarriers are assigned fixed positions in the frequency domain within an OFDMA symbol.

To define adjacent subcarrier permutation, a BIN, which is a set of 9 contiguous subcarriers within an OFDMA symbol represented in Figure C.1, is a basic allocation unit both in downlink and in uplink.

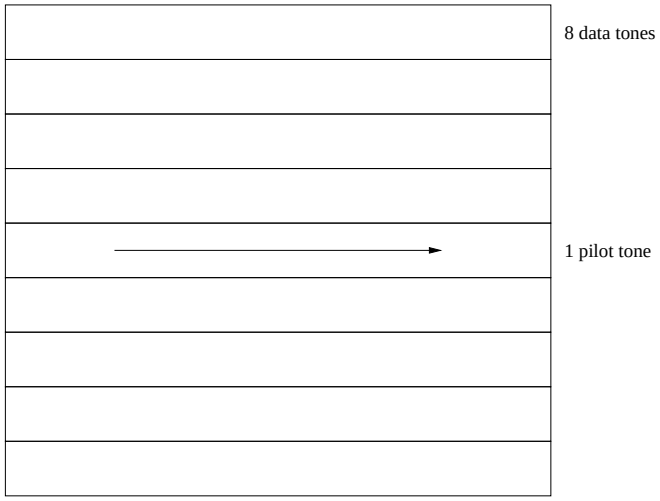

Figure C.1: Bin structure.

AMC subchannel consists of 6 contiguous bins in a same band. AMC allocation, instead, can be made by two mechanisms: by subchannel index reference in UL-MAP and by subchannel allocation in a band using H-ARQ map.

A slot consists of N bins by M symbols ($N \times M$). A group of 4 rows is called a *physical band*. There are four types of AMC subchannels, which are different in the collection of six bins in a band:

- **default type**: a slot consists o 6 consecutive bins;
- a slot is defined as 2 bins by 3 symbols;
- a slot is defined as 3 bins by 2 symbols;
- a slot is defined as 1 bins by 6 symbols;

The enumeration of bins in a slot is the following: the available bins in a band are enumerated starting from the lowest bin in the first symbol to the last bin in the symbol and then going to the lowest bin in the next symbol and so on.

The parameters are the same enumerated before: Number of DC subcarriers, Number of guard subcarriers: left and right, N_{used}, Total number of subcarriers, Number of pilots, Number of data subcarriers. In addition, there are three new parameters:

1. **Number of physical band**
2. **Number of bins per physical band**
3. **Number data subcarriers per subchannel**

Parameter	FFT size			
	2048	1024	512	128
Number of bins per physical band	4	4	4	4
Number of physical band	48	24	12	6
Number of data subcarriers per subchannel	48	48	48	48

Table C.3: New parameters introduced with the AMC permutation.

and they are dependent on the FFT size, as shown in Table C.3

We remember that an AMC subchannel consists of 6 contiguous bins which are made of 9 subcarriers (8 data and 1 pilot) and so the data subcarriers in a subchannel are $8 \times 6 = 48$, we observe also that only in the first type the number of data subcarriers is the same that we have in a subchannel.

The number of bins per physical band is fixed to 4 and the number of physical bins is obtained dividing the number of data subcarriers per 32 deriving from 8×4 i.e. the number of data subcarriers in a bin per the number of bins.

Optional Permutation for PUSC Adjacent Subcarrier Allocation (PUSC-ASCA)

Symbol structure shall use the parameters defined for regular PUSC and the same cluster structure shall be maintained.

Allocation of subcarriers to subchannels shall be performed in the following manner:

1. The subcarriers are divided into physical clusters, each of 14 adjacent subcarriers
2. Cluster used for a specific DL allocation shall be the first $2 * SubchannelOffset$
3. The clusters are concatenated into blocks using the rule described in Table C.4, where

Number of subchannels	Clusters concatenated
$n \leq 12$	1 block of $2 \times n$ clusters
$n = 12 \times k$	k blocks of 24 clusters
$n > 12$, $n \neq 12 \times k$	$(k-1)$ blocks of 24 clusters 1 block of $2 \times \lceil (m+12)/2 \rceil$ clusters 1 block of $2 \times \lfloor (m+12)/2 \rfloor$ clusters

Table C.4: Allocation of subcarriers to subchannels.

- **n** is the number of allocated subchannels
- $\mathbf{k} = \lfloor n/12 \rfloor$
- $\mathbf{m} = n \mod 12$

4. Per block, remove the pilot carriers from the clusters associated with the section, take the remaining data subcarriers and using the same procedure for the partition of data subcarriers in FUSC permutation; the subcarriers are partitioned into subchannels containing 24 data subcarriers in each OFDMA symbol.

We can define also an adjacent subcarrier allocation using distributed clusters for the PUSC mode; symbol structure shall use the same parameters of the regular PUSC and the allocation of subcarriers to subchannels will be:

1. divide the subcarriers into 120 physical clusters within 14 adjacent subcarriers
2. renumber the physical clusters into logical clusters using the formula

$$\begin{aligned} LogicalCluster &= RenumberingSequence((PhysicalCluster \\ &+ 13 * IDcell) \mod 120) \end{aligned} \tag{C.5}$$

3. divide the clusters into 6 major groups
4. allocating carriers to subchannel in each major group depends on the specific allocation performed.

Appendix D

Renumbering sequence for PUSC permutation

FFT size	Renumbering Sequence
2048	6, 108, 37, 81, 31, 100, 42, 116, 32, 107, 30, 93, 54, 78, 10, 75, 50, 111, 58, 106, 23, 105, 16, 117, 39, 95, 7, 115, 25, 119, 53, 71, 22, 98, 28, 79, 17, 63, 27, 72, 29, 86, 5, 101, 49, 104, 9, 68, 1, 73, 36, 74, 43, 62, 20, 84, 52, 64, 34, 60, 66, 48, 97, 21, 91, 40, 102, 56, 92, 47, 90, 33, 114, 18, 70, 15, 110, 51, 118, 46, 83, 45, 76, 57, 99, 35, 67, 55, 85, 59, 113, 11, 82, 38, 88, 19, 77, 3, 87, 12, 89, 26, 65, 41, 109, 44, 69, 8, 61, 13, 96, 14, 103, 2, 80, 24, 112, 4, 94, 0
1024	6, 48, 37, 21, 31, 40, 42, 56, 32, 47, 30, 33, 54, 18, 10, 15, 50, 51, 58, 46, 23, 45, 16, 57, 39, 35, 7, 55, 25, 59, 53, 11, 22, 38, 28, 19, 17, 3, 27, 12, 29, 26, 5, 41, 49, 44, 9, 8, 1, 13, 36, 14, 43, 2, 20, 24, 52, 4, 34, 0
512	12, 13, 26, 9, 5, 15, 21, 6, 28, 4, 2, 7, 10, 18, 29, 17, 16, 3, 20, 24, 14, 8, 23, 1, 25, 27, 22, 19, 11, 0
128	2, 3, 1, 5, 0, 4

Table D.1: Renumbering sequences for the available FFT size (i.e. 2048, 1024, 512 and 128).

Appendix E

Optional codes in WiMAX standard

E.1 Block Turbo Coding (BTC)

This is optional and not required to be implemented.

BTC is based on the product of two simple component codes: the **binary extended Hamming codes** and the **parity check codes**. To create the extended Hamming codes, an overall even parity check bit is added at the end of each code word. The generator polynomials for OFDMA Hamming code are reported in the Table E.1.

n′	k′	Generator polynomial
15	11	$n^4 + x + 1$
31	26	$x^5 + x^2 + 1$
63	57	$x^6 + x + 1$

Table E.1: OFDMA Hamming code generator polynomials.

The component codes are used in a two-dimensional matrix form, as we can see in Figure E.1.

The K_x information bits in the rows are encoded into n_x bits using the component block (n_x, K_x) code specified for the respective composite code.

After encoding the rows, the columns are encoded using a block code (n_y, K_y), where the check bits of the first code are also encoded.

The overall block size is (n, K), where $n = n_x \times n_y$ and $K = K_x \times K_y$, and so the code rate is $R = R_x \times R_y$, with $R_i = K_i/n_i$, $i = x, y$. The Hamming distance is $d = d_x \times d_y$. The first bit in the first row is the LSB and the last bit in the last row is the MSB and the transmission if the block over the channel shall occur in a linear fashion, with all bit of the first row transmitted left to right followed by the second row and so on.

To match a required pocket size, BTC may be shortened by removing symbols from the BTC array. This operation requires three steps:

1. remove I_x rows and I_y columns which is equivalent to shortening the constituent codes that make up the product codes

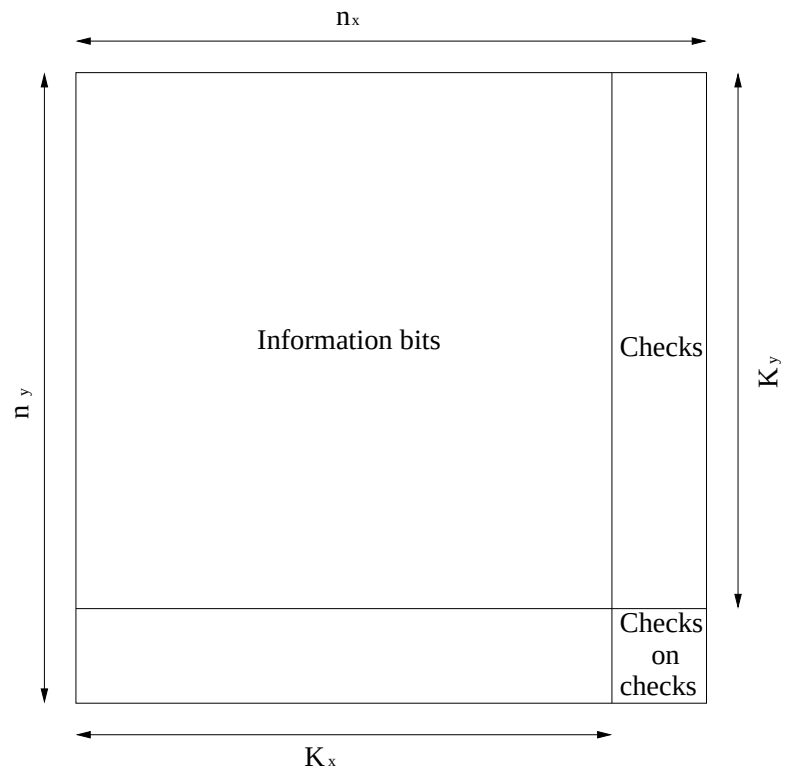

Figure E.1: Block Turbo Coding (BTC) structure.

2. remove B individual bits from the first row starting with the LSB

3. if at this point there isn't an integral number of data bytes the Q left over LSB are zero-filled by the encoder.

In Figure E.2 these three steps are represented.

The new coded block length is $(n_x - I_x)(n_y - I_y) - B$ and the corresponding information length is $(n_x - I_x)(n_y - I_y) - B - Q$ and so the code rate is given by:

$$R = \frac{(K_x - I_x)(K_y - I_y) - B - Q}{(n_x - I_x)(n_y - I_y) - B}. \tag{E.1}$$

In Table E.2 the block sizes for the optimal modulation and coding schemes using BTC are given.

Table E.3 gives the code parameters for each of possible data and coded block sizes.

E.2 Convolutional Turbo Code (CTC)

In the standard [2] this code is defined as optional, but in [109] it is required to be implemented.

CTC can be used for the support of optional H-ARQ. It used a **double binary circular recursive systematic convolutional code**. These bits of the data to be

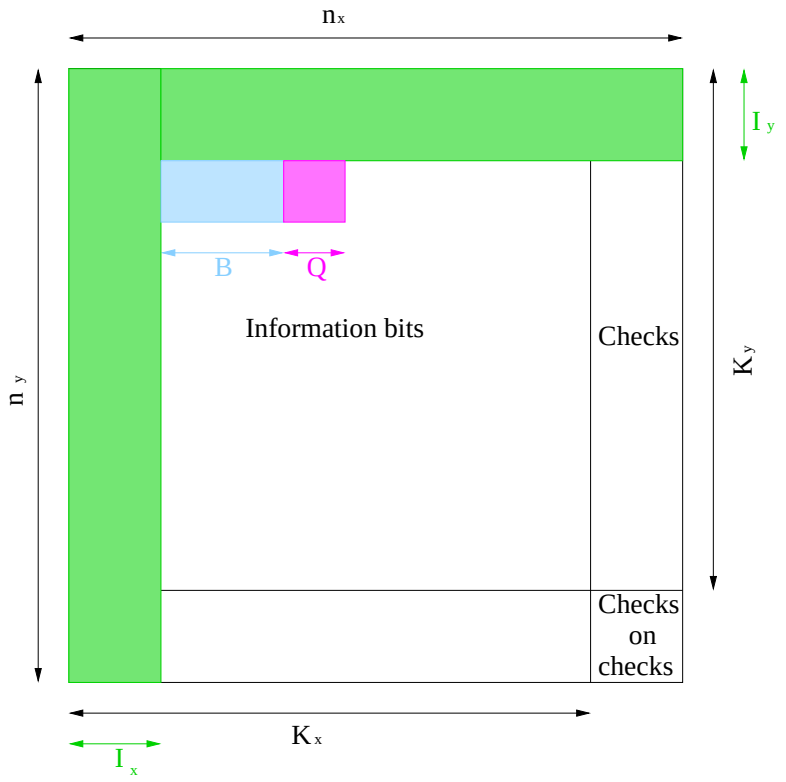

Figure E.2: Shortened BTC structure.

Encoding rate	QPSK		16 - QAM		64 - QAM		Coded bytes
	1/2	3/4	1/2	3/4	1/2	3/4	
Allowed data (bytes)	6	9					12
	12	20	16	20			24
	18	25			16	25	36
	23	35	23	35			48
	31						60
	40		40		40		72

Table E.2: Useful data payload for a subchannel.

Data bytes	Coded bytes	Constituent	Code parameters
6	12	$(31, 32)(16, 11)$	$I_x = 4$, $I_y = 8$, $B = 0$, $Q = 6$
9	12	$(16, 15)(16, 15)$	$I_x = 6$, $I_y = 6$, $B = 4$, $Q = 5$
12	24	$(32, 31)(16, 11)$	$I_x = 14$, $I_y = 5$, $B = 6$, $Q = 0$
20	24	$(16, 15)(16, 15)$	$I_x = 2$, $I_y = 2$, $B = 4$, $Q = 5$
18	36	$(32, 31)(16, 11)$	$I_x = 5$, $I_y = 5$, $B = 9$, $Q = 3$

Table E.3: Optional channel coding per modulation.

encoded are alternatively fed to A and B, starting with the MSB of the first byte being fed to A (see Figure E.3).

The encoded is fed by blocks of k bits on N couples ($k = 2 * N$ bits) ad for all the frame size k is a multiple of 8 and N a multiple of 4.

The polynomial defining the connection are:

- for the feedback branch: $1 + D + D^3$
- for the Y parity bit: $1 + D^2 + D^3$
- for the W parity bit: $1 + D^3$.

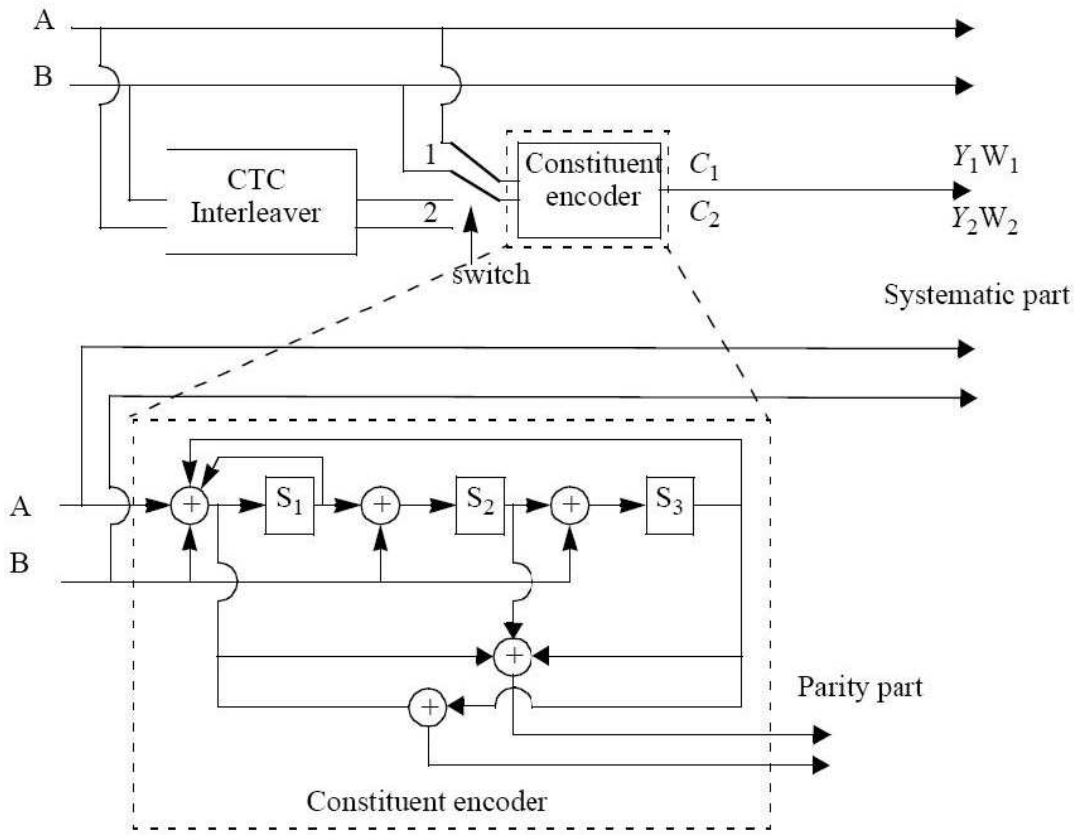

Figure E.3: CTC encoder.

The encoder block size shall depend on the number of slots allocated and the modulation specified for the current transmission. Concatenation of the number of slots shall be performed in order to make larger blocks of coding where it is possible, with the limitation of not exceeding the larger block under size for applied modulation and coding. The concatenation rule shall not be used when using ARQ.

For any modulation and FEC rate we define the following parameters:

- j: dependent on the modulation and FEC rate,
- n: number of allocated slots/repetition factor,
- k: defined as $\lfloor n/j \rfloor$,
- m: defined as $\mod (n, j)$.

Table E.5 shows the rules used for slots concatenation, while Table E.4 gives the values of j for different modulations and rates:

Number of slots	**Slots concatenation**
$n < j$, $n \neq 7$	1 block of n slots
$n = 7$	1 block of 4slots 1 block of 3slots
$n > j$	if mod $(n, j) = 0$ k blocks of j slots else $k - 1$ blocks of j slots 1 block of $\lceil((m + j)/2)\rceil$ slots (+1 if equal to seven) 1 block of $\lfloor((m + j)/2)\rfloor$ slots (+1 if equal to seven)

Table E.4: Slots concatenation rules for CTC.

Modulation rate	**j**
QPSK 1/2	10
QPSK 3/4	6
16−QAM 1/2	5
16−QAM 3/4	3
64−QAM 1/2	3
64−QAM 2/3	2
64−QAM 3/4	2
64−QAM 5/6	2

Table E.5: Encoding slot concatenation for different rates in CTC.

E.2.1 CTC interleaver

It is composed of two states:

Step 1: Switch alternate couples

Let the sequence $u_0 = [(A_0, B_0), (A_1, B_1), \ldots, (A_{N-1}, B_{N-1})]$ be the input to the first encoding C_1.

for $i = 0, \ldots, N - 1$

mod $(i, 2) = 0$

let $(A_i, B_i) \rightarrow (B_i, A_i)$ (i.e. switch the couple)

this step gives a sequence $u_1 = [(B_0, A_0), (B_1, A_1), \ldots, (B_{N-1}, A_{N-1})] =$

$= [u_1(0), u_1(1), \ldots, u_1(N - 1)]$.

Step 2: $\boldsymbol{P(j)}$

The function $P(j)$ provides the address of the couple of the sequence u_1 that shall be mapped onto the address j of the interleaved sequence (i.e., $u_2(j) = u_1(P(j))$).

for $j = 0, \ldots, N-1$

switch mod $(j, 4)$:

case 0:$P(j) = (P_0 \cdot j + 1)_{mod_4}$

case 1:$P(j) = (P_0 \cdot j + 1 + N/2 + P_1)_{mod_4}$

case 2:$P(j) = (P_0 \cdot j + 1 + P_2)_{mod_4}$

case 0:$P(j) = (P_0 \cdot j + 1 + N/2 + 3)_{mod_4}$

This step gives a sequence $u_2 = [u_1(P(0)), u_1(P(1)), u_1(P(2)), u_1(P(3)), \ldots, u_1(P(N-1))] = [(B_{P(0)}, A_{P(0)}), (A_{P(1)}, B_{P(1)}), (B_{P(2)}, A_{P(2)}), (A_{P(3)}, B_{P(3)}), \ldots, (A_{P(N-1)}, B_{P(N-1)})]$. Sequence u_2 is the input to the second encoding C_2.

The values of N, P_0, P_1, P_2 and P_3 for every modulation and rate are listed in Table 326 of the standard [1]-[2].

E.2.2 Determination of CTC circulation state

The state of the encoder is denoted $\mathcal{S}$ $(0 \leq \mathcal{S} \leq 7)$ with $\mathcal{S} = 4s_1 + 2s_2 + s_3$ (see Figure E.3). The circulation states Sc_1 and Sc_2 are determined by the following operations:

1. Initialize the encoder with state 0. Encode the sequence in the natural order for the determination of Sc_1 or in the interleaved order for determination of Sc_2. In both cases the final state of the encoder is $S0_{N-1}$;

2. According to the length N of the sequence, use Table E.6 to find Sc_1 and Sc_2.

mod $(N, 7)$	**$S0_{N-1}$**							
	0	1	2	3	4	5	6	7
1	0	6	4	2	7	1	3	5
2	0	3	7	4	5	6	2	1
3	0	5	3	6	2	7	1	4
4	0	4	1	5	6	2	7	3
5	0	2	5	7	1	3	4	6
6	0	7	6	1	3	4	5	2

Table E.6: Circulation state look-up table (Sc).

E.2.3 Subpacket generation

Proposed FEC structure punctures the mother codeword to generate a subpacket with various coding rates. The subpacket is also used as HARQ packet transmission. Figure E.4 shows a block diagram of subpacket generation. 1/3 CTC encoded codeword goes through interleaving block and the puncturing is performed. Figure E.4 shows block diagram of the interleaving block. The puncturing is performed to select the consecutive interleaved bit sequence that starts at any point of whole codeword. For the first transmission, the subpacket is generated to select the consecutive interleaved

bit sequence that starts from the first bit of the systematic part of the mother codeword. The length of the subpacket is chosen according to the needed coding rate reflecting the channel condition. The first subpacket can also be used as a codeword with the needed coding rate for a burst where HARQ is not applied.

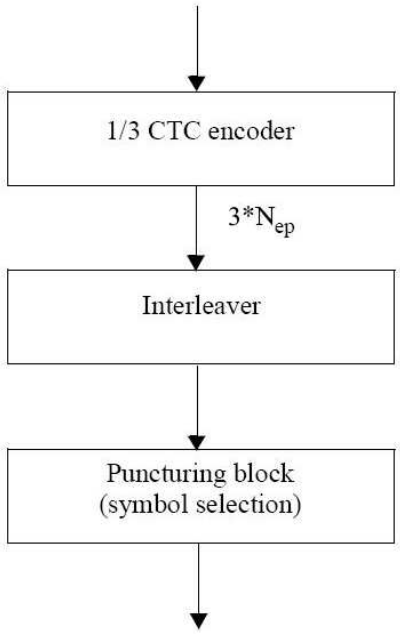

Figure E.4: Block diagram of subpacket generation.

Symbol separation

All of the encoded symbols shall be demultiplexed into six subblocks denoted A, B, Y_1, Y_2, W_1, and W_2. The encoder output symbols shall be sequentially distributed into six subblocks with the first N encoder output symbols going to the A subblock, the second N encoder output going to the B subblock, the third N to the Y_1 subblock, the fourth N to the Y_2 subblock, the fifth N to the W_1 subblock, the sixth N to the W_2 subblock, etc.

Subblock interleaving

The six subblock shall be interleaved separately. The interleaving of performed by the unit of symbol. The sequence of interleaver output symbols for each subblock shall be generated by the procedure described below. The entire subblock of symbols to be interleaved is written into an array at addresses from 0 to the number of the symbols minus one $(N-1)$, and the interleaved symbols are read out in a permuted order with the i-th symbol being read from an address, AD_i $i = 0, 1, \ldots, N-1$, as follows:

1. Determine the subblock interleaver parameters, m and J. Table E.7 gives these parameters.

2. Initialize i and k to 0.

3. Form a tentative output address T_k according to the formula:

$$T_k = 2^m \mod (k, J) + BRO_m(\lfloor k/J \rfloor)$$

where $BRO_m(y)$ indicates the bit-reverse m-bit value of y (i.e. $BRO_3(6) = 3$).

4. If T_k is less than N $AD_i = T_k$ and increment i and k by 1. Otherwise, discard T_k and increment k only.

5. Repeat steps 1 and 2 until all N interleaver output addresses are obtained.

Block size (bits) N_{EP}	**N**	**Subblock interleaver parameters**	
		m	J
48	24	3	3
72	36	4	3
96	48	4	3
144	72	5	3
192	96	5	3
216	108	5	4
240	120	6	2
288	144	6	3
360	180	6	3
384	192	6	3
432	216	6	4
480	240	7	2

Table E.7: Parameters for the subblock interleavers.

Symbol grouping

The channel interleaver output sequence shall consist of the interleaved A and B subblock sequence, followed by a symbol-by-symbol multiplexed sequence of the interleaved Y_1 and Y_2 subblock sequence, followed by a symbol-by-symbol multiplexed sequence of the interleaved W_1 and W_2 subblock sequence. The symbol-by-symbol multiplexed sequence of interleaved Y_1 and Y_2 subblock sequences shall consist of the first output bit from the Y_1 subblock interleaved, the first output bit from the Y_2 subblock interleaved, the second output bit from the Y_1 subblock interleaved, the second output bit from the Y_2 subblock interleaved, etc. The symbol-by-symbol multiplexed sequence of interleaved W_1 and W_2 subblock sequences shall consist of the first output bit from the W_1 subblock interleaved, the first output bit from the W_2 subblock interleaved, the second output bit from the W_1 subblock interleaved, the second output bit from the W_2 subblock interleaved, etc. Figure E.5 shows the interleaving scheme.

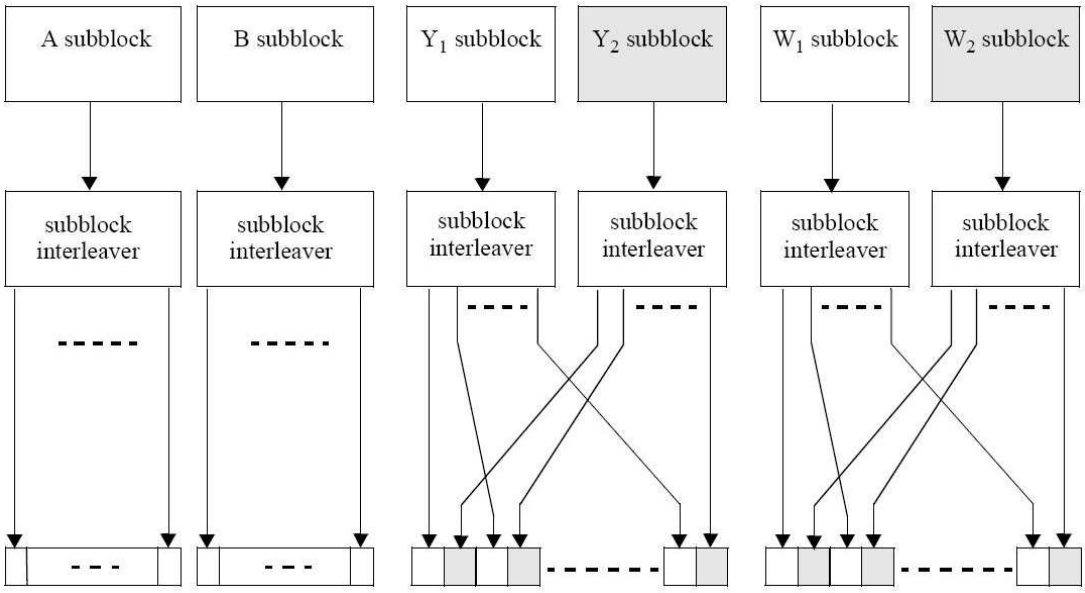

Figure E.5: Block diagram of the interleaving scheme.

Bibliography

[1] IEEE 802.16-2004 Standard, Air interface for fixed broadband wireless access systems. October 2004.

[2] IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004) Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment 2:Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1. February 2006.

[3] R. G.Gallager. Low density parity check codes. *IEEE Transaction on Information Theory, IT-8*, pages 21–28, January 1962.

[4] R. G. Gallager. *Low density paroty check codes.* MIT Press, Cambridge, 1963.

[5] D. J. C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Trans. on Inform. Theory*, IT-45(2):399–431, March 1999.

[6] C. R. P. Hartmann Y. S. Hang and C. C. Chen. Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes. *IEEE Transaction on Information Theory*, 39(5):1514–1523, September 1993.

[7] Y. S. Hang. A new treatment of priority-first search maximum-likelihood soft-decision decoding of linear block codes. *IEEE - IST*, page 394, June 29 - July 4 1997.

[8] C. R. P. Hartmann Y. S. Hang and K. G. Mehrotra. Decoding linear block codes using priory-first search: performance analysis and suboptimal version. *IEEE Transaction on Information Theory*, 44(3):1233–1246, May 1998.

[9] A. Wu W. K. Leung, W. L. Lee and Li Ping. Efficient implementation technique of LDPC decoder. *Electronics Letters*, 37(20):1231–1232, 27th September 2001.

[10] A. G. Scandurra, A. L. Dai Pra, L. Arone, L. Passoni, and J. Castiñeira Moreira. A genetic-algorithm based decoder for low density parity check codes. *Latin America Applied Research*, pages 169–172, 2006.

[11] TGn Sync proposal. Available at *http://www.tgnsync.org/home*.

[12] WWISE proposal. Available at *http://www.wwise.org*.

[13] 3GPP-LTE proposal. Available at *http://www.3gpp1.net/ftp/specs/archive/36_series*.

[14] L. J. Cemini Y. Li and N. R. Sollenberger. Robust channel estimation for OFDM systems with rapid dispersive fading channels. *IEEE Trans. Commun.*, 46:902–915, July 1998.

[15] J. McCool M. Dentino and B. Widrow. Adaptive filtering in the frequency domain. *Proceeding of the IEEE*, 66(12):1658–1659, December 1978.

[16] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, pages 47–60, November 1996.

[17] J.choi. A joint channel estimation and detection for frequency-domain equalization using an approximate EM algorithm. *Signal processing*, (84):865–880, 2004.

[18] D. J. C. MacKay. *Information theory, inference, and learning algorithms*. University Press, Cambridge, 2003.

[19] A. Glavieux C. Berrou and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. *Proc. ICC '93*, pages 1064–1070, May 1993.

[20] C. Berrou and A. Glavieux. Near optimum error-correcting coding and decoding: Turbo-codes. *IEEE Trans. Commun.*, COMM-44:1261–1271, October 1996.

[21] C. Berrou and A. Glavieux. Reflection on the prize paper: near optimum error-correcting coding and decoding: turbo codes. *IEEE Inform. Theory Soc. Newsletter*, 48(2):24–31, June 1998.

[22] S. Lin Y. Kou and M. P. C. Fossorier. Low density parity check codes based on finite geometries: a rediscovery and new results. *IEEE Trans. Inform. Theory*, 47(7):2711–2736, November 2001.

[23] J. A. Seok S. H. Lee and E. K. Joo. Serial concatenation of LDPC and turbo code for the next generation mobile communications. *Wireless and optical communications networks (WOCN)*, pages 425–427, March 2005.

[24] J. A. Seok S. H. Lee and E. K. Joo. Serially concatenated LDPC and turbo code with global iteration. *Proc. 18th International Conference on Systems Engineering (ISCEng'05)*, pages 201–204, August 2005.

[25] S. H. Lee and E. K. Joo. The effect of Block Interleaving in an LDPC-turbo concatenated code. *ETRI Journal*, 28(5):672–675, October 2006.

[26] O. Edfors, M. Sandell, J. J. van de Beck, S. K. Wilson, and O. P. Borjesson. OFDM channel estimation by singular values decomposistion. *IEEE Trans. Commun.*, 46:931–939, July 1998.

[27] M. Morelli and U. Mengali. A comparison of pilot-aided channel estimation methods for OFDM systems. *IEEE Transaction on Signal Processing*, 49(12), December 2001.

[28] N. Benvenuto and G. Cherubini. *Algorithms for communications systems and their applications.* Wiley, Chichester, 2002.

[29] V. Tarokh and H. Jafarkhani. A differential detection scheme for transmit diversity. *IEEE Journal on Selected Areas in Communications*, 18(7), July 2000.

[30] R. van Nee and R. Prasad. *OFDM for Wireless Multimedia Communications.*

[31] J. Rinne and M. Renfors. Pilot spacing in orthogonal frequency division multiplexing systems on practical channels. *IEEE Trans. Consum. Electron.*, 42:959–962, November 1996.

[32] Y. Le. Pilot-symbol-aided channel estimation for OFDM in wireless systems. *IEEE Trans. Veh. Technol.*, 48:1207–1215, July 2000.

[33] S. Kaiser P. Hoeher and I. Robertson. Two-dimensional pilot-symbol-aided channel estimation by Wiener filtering. *Proc. ICASSP'97*, pages 1845–1848, 1997.

[34] G. Fock M. Speth, S. Fechtel and H. Meyr. Broadband transmission using OFDM: system performance and receiver complexity. *Proc. Int. Zurich Seminar on Broadband Communications*, pages 99–104, 1998.

[35] R. Negi and J. Cioffi. Pilot tone selection for channel estimation in a mobile OFDM system. *IEEE Trans. Consum. Electron.*, 44:1122–1128, August 1998.

[36] Yang B, K. Ben Letaief, R. S. Cheng, and Z. Cao. Channel estimation for OFDM transmission in multipath fading channels based on parametric channel modeling. *IEEE Trans. Commun.*, 49(3):467–479, March 2001.

[37] J. T. Chen, A. Paulraj, and U. Reddy. Multichannel maximum-likelihood sequence estimation (MLSE) equalizer for GSM using a parametric channel model. *IEEE Trans. Commun.*, 47:53–63, January 1999.

[38] W. C. Jakes. *Microwave mobile communications.* IEEE Press, New Jork, 1993.

[39] D. Schafhuber, G. Matz, and F. Hlawatsch. Adaptive Winner filters for time-varying channel estimation in wireless OFDM systems. *IEEE ICASSP 2003*, 4:688–691, April 2003.

[40] G. Auer, S. Sand, and A. Dammann. Comparison of low complexity OFDM channel estimation techniques. *Proc of Int OFDM Workshop*, 2003.

[41] Proakis. *Digital communications.* McGraw Hill, Boston, MA, fourth edition, 2001.

[42] D. Johnson. Cramèr-Rao Bound. August 2003.

[43] J. J. van de Beek, O. Edfors, M. Sandell, S. K. Wilson, and P. O. Borjesson. On channel estimation in OFDM system. *Proc. VTC'95*, 2:815–819, September 1995.

[44] M. Hsieh and C. Wei. Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency selective fading channels. *IEEE Trans. Consumer Electron.*, 44(3), February 1998.

[45] S.M. Kay. *Fundamentals of Statistical Signal Processing: Vol. 1: Estimation Theory*. Prentice-Hall, EnglewoodCliBs, NJ, 1993.

[46] S. Haykin. *Adaptive filter theory*. Prentice-Hall, New Yersey, 1996.

[47] E. G. Larsson, G. Liu, J. Li, and G. B. Giannakis. Joint Symbol Timing and Channel Estimate for OFDM Based WLANs. *IEEE Commun. Letters*, 5(8):325–327, August 2001.

[48] L. Deneire, P. Vandenameele, L. van der Perre, B. Gyselinckx, and M. Engels. A low-complexity ML channel estimator for OFDM. *IEEE Trans. Comm.*, 51(2):135–140, February 2003.

[49] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD). 2002-2003.

[50] T. S. Rappaport. *Wireless communications*. Prentice Hall PTR, New Jersey, 1996.

[51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statiscal Soc. Ser. B*, 39(1):1–38, 1977.

[52] For an extensive list of references to papers describing applications of the EM algorithm, see `http://www.engineering/usu.edu/Departments/ece/Publications/Moon` on the Wold-Wide Web.

[53] C. Jiang. The use of mixture models to detect affects of major genes on quantitative characteristics in a plant-breeding experiment. *Genetics*, 136(1):383–394, 1994.

[54] R. Redner and H. F. Walker. Mixture densities, maximum-likelihood estimation and the EM algorithm (review). *SIAM rev.*, 26(2):195–237, 1984.

[55] J. Schmee and G. J. Hahn. Simple method for regression analysis with censored data. *Technometris*, 21(4):417–432, 1979.

[56] L. Little and D. Rubin. On jointly estimating parameters and missing data by maximizing the complete-data likelihood. *Am. Statistn.*, 37(3):200–218, 1983.

[57] L. A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Med. Im.*, 1:113–122, October 1982.

[58] D. L. Snyder and D. G. Politte. Image reconstruction from list-mode data in an emission tomography system having time-of-flight measurements. *IEEE Nucl. S.*, 30(3):1843–1849, 1983.

[59] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *P. IEEE*, 77(2):257–286, 1989.

[60] L. Rabiner and B. H. Juang. *Fundamentals of speech recognition.* 1993.

[61] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-time processing of speech signals.* 1993.

[62] M. Segal and E. Weinstein. Parameter estimation of continuous dynamical linear systems given discrete time observations. *P. IEEE*, 75(5):727–729, 1987.

[63] S. Zabin and H. Poor. Efficient estimation of class-A noise parameters via the EM algorithm. *IEEE Trans. Info. T.*, 37(1):60–72, 1991.

[64] A. Isaksson. Identification of ARX models subject to missing data. *IEEE Auto C*, 38(5):813–819, 1993.

[65] A. Katsaggelos and K. Lay. Maximum likelihood blur identification and image restoration using the algorithm. *IEEE Trans. Sig. Proc.*, 39(3):729–733, 1991.

[66] A. Ansari and R. Viswanathan. Application of EM algorithm to the detection of direct sequence signal in pulsed noise jamming. *IEEE Trans. Com.*, 41(8):1151–1154, 1993.

[67] M. Feder. Parameter estimation and extraction of helicopter signals observed with a wide-band interference. *IEEE Trans. Sig. Proc.*, 41(1):233–244, 1993.

[68] G. Kaleh. Joint parameter estimation and symbol detection for linear and non-linear unknown channels. *IEEE Trans. Com.*, 42(7):2506–2513, 1994.

[69] W. Byrne. Alternating minimization and Boltzman machine learning. *IEEE Trans. Neural Net.*, 3(4):612–620, 1992.

[70] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Comp.*, 6(2):181–214, 1994.

[71] R. Streit and T. Luginbuh. ML training of probabilistic neural networks. *IEEE Trans. Neural Net.*, 5(5):764–783, 1994.

[72] M. Miller and D. Fuhrmann. Maximum likelihood narrow-band direction finding and the EM algorithm. *IEEE Trans. ASSP*, 38(9):1560–1577, 1990.

[73] S. Vaseghi and P. Rayner. Detection and suppression of impulsive noise in speech communication systems. *IEEE Proc-I*, 137(1):38–46, 1990.

[74] S. E. Bialkowski. Expectation-Maximization (EM) algorithm for regression, deconvolution and smoothing of shot-noise limited data. *Journal of Chemometrics*, 1991.

[75] C. Georghiades and D. Snyder. The EM algorithm for symbol unsynchronized sequence detection. *IEEE Commun.*, 39(1):54–61, 1991.

[76] N. Antoniadis and A. Hero. Time delay estimation for filtered Poisson processes using an EM-type algorithm. *IEEE Trans. Sig. Proc.*, 42(8):2112–2123, 1994.

[77] M. Segal and E. Weinstein. The cascade EM algorithm. *P. IEEE*, 76(10):1388–1390, 1988.

[78] C. Gyluai, S. Bialkowski, G. S. Stiles, and L. Powers. A comparison of three multiplatform message-passing interfaces on an expectation-maximization algorithm. *Proceedings of the 1993 World Conference on Transputers*, pages 451–464, 1993.

[79] R. E. Blahut. Computation of channel capacity and rate distortion functions. *IEEE Trans. Infor. Th.*, 18:460–473, July 1972.

[80] I. Csiszar and G. Tusnday. Information geometry and alternating minimization procedures. *Statistics and Decisions, Supplement Issue I*, 1984.

[81] C. Wu. On the convergence properties of the EM algorithm. *Ann. Statist.*, 11(1):95–103, 1983.

[82] R. A. Boyles. On the convergence of the EM algorithm. *J. Roy Sta. B.*, 45(1):47–50, 1983.

[83] D. Slepian. A class of binary signalling alphabeths. *Bell Sys. Tech. J.*, 35:203–234, Jenuary 1956.

[84] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density party check codes. *IEEE Electronic Letters*, 32(18):1645–1646, August 1996.

[85] M. A. Shokrollahi T. J. Richardson and R. L. Urbanke. Design of capacity - approaching irregular Low-Density Parity-Check codes. *IEEE transaction on information theory*, 47(2):619–637, February 2002.

[86] M. Luby, M. Mitzenmacher, A. Shokrollahi D. Spielman, and V. Stemann. Pratical loss-resilient codes. *Proc. 29th Annu. ACM Symp. Theory of Computing*, pages 150–159, 1997.

[87] M. Luby, M. Mitzenmacher, A. Shokrollahi D. Spielman, and V. Stemann. Pratical loss-resilient codes. *IEEE Trans. Inform. Theory*, 47:569–584, February 2001.

[88] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. *Proc.* 30*th Annu. ACM Symp. Theory of Computing*, pages 248–258, 1998.

[89] S. Wilson D. MacKay and M. Davey. Comparison of construction of irregular Gallager codes. *IEEE Trans. Commun.*, 47:1449–1454, October 1999.

[90] S. Lin and D. J. Costello. *Error control coding.* Pearson, Prentice Hall, New York, 2004.

[91] M. Sipser and D. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42:1710–1722, Novenber 1996.

[92] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. *Proc. 29th Annual ACM Symp. Theory of Computing*, pages 150–159, 1997.

[93] T. J. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inform Theory*, pages 638–656, Febbruary 2001.

[94] S. Andreassen, M. Woldbye, B. Falck, and S. Andersen. MUNIN-A causal probabilistic network for the interpretation of electromyographic findings. *Proc. 10th Nat. Conf. on AI (AAAI: Menlo Park CA., 1987)*, pages 121–123.

[95] S. W. Golomb, R. E. Peile, and R. A. Scholtz. *Basic Concepts in Information Theory and Coding: The Adventures of Secret Agent 00111.* Plenum Press, New York, 1994.

[96] Z. Liu and D. A. Pados. Low complexity decoding of finite geometry LDPC codes. *IEEE Intrernational Conference on Communications (ICC'03)*, 4:2713–2717, 2003.

[97] V. J. Easton and J. H. McColl. Statistics glossary - v1.1. Available online at *http://www.stats.gla.ac.uk/steps/glossary/confidence_intervals.html.*

[98] N. J. Nilson. *Principles of artificial intelligence.* Tioga Pubblishing Co., Palo Alto, CA, 1980.

[99] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection.* MIT Press, Cambridge, MA, 1992.

[100] Z. Michalewicz. *Genetic algorithms+data structure = evolution programs.* Springer-Verlag, Berlin, 1992.

[101] K. Dontas and K. De Jong. Discovery of maximal distance codes using genetic algorithms. *Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, pages 805–811, 1990.

[102] L. Eshelman J. Schaffer, J. Caruana and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proc. of the Third international Conference on Genetic Algorithms*, pages 51–60, 1989.

[103] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transaction of Information Theory, IT-27*, pages 533 – 547, September 1981.

[104] D. MacKay and R. Neal. Good codes based on very sparse matrices. *Cryptography and Coding, 5th IMA Conf., C. Boyd, Ed. Lecture Notes in Computer Science*, October 1995.

[105] N. Alon and M. Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Trans. Inform. Theory*, 42:1732–1736, November 1996.

[106] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *Proc. ACM SIGCOMM '98*, pages 56–67, January 1998.

[107] H. Behairy and S. C. Chang. Parallel concatenated Gallager codes for CDMA applications. *Proc. IEEE MILCOM 2001*, 2:1002–1006, November 2001.

[108] M. A. Shokrollahi T. J. Richardson and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[109] WiMAX Forum Mobile "WiMAX sistem profile". February 2006.

[110] D. S. Baum, J. Salo, G. Del Galdo, M. Milojevic, P. Kyosti, and J. Hansen. An interim channel model for beyond-3G systems. *Proc. IEEE VTC 05*, May 2005.

[111] Milan Narandzic, Christian Schneider, Reiner Thoma, Tommi Jamsa, Pekka Kyosti, and Xiongwen Zhao. Comparison of SCM, SCME, and WINNER Channel Models. *Proc. IEEE VTC 05*, pages 413–417, April 2007.

[112] Yumin Lee and Jyh-Hau Chen. The Effect of Channel Estimator Memory Mismatch on the Performance of MLSE in Wireless Data Communications. *IEEE International Conference on Communications*, April 2002.

[113] J. Chen and Y. Lee. Joint synchronization, channel length estimation, and channel estimation for the maximum likelihood sequence estimator for high speed wireless communications. *IEEE Vehicular Technology Conf. (VTC 2002 - Fall)*, 3:1535–1539, 2002.

[114] G. E. Bottomley, J.-C. Chen, and D. Koilpillai. System and methods for selecting an appropriate detection technique in a radiocommunication system. *U.S. Patent 6333953B1*, December 2001.

[115] E. G. Larsson, G. Liu, J. Li, and G. B. Giannakis. Joint symbol timing and channel estimation for OFDM based WLANs. *IEEE Com. Letters*, 5(8):325–327, August 2001.

[116] Y. Zhao and A. Huang. A novel channel estimation method for OFDM mobile communication systems based on pilot signals and transformdomain processing. *IEEE Vehicular Technology Conf.*, 3:2089–2093, 1997.

[117] V. D. Nguyen, H. P. Kuchenbecker, and M. Patzold. Estimation of the channel impulse response length and the noise variance for OFDM systems. *IEEE Vehicular Technology Conference (VTC 2005-Spring)*, 1:429–433, June 2005.

[118] V. D. Nguyen. Channel impulse response length estimation and interference cancellation for OFDM systems. 2004.