

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

纜線數據機的更誤解碼與渦輪等化器研究



**Research in Error-Correction Decoding and Turbo
Equalization for Cable Modem Receivers**

研 究 生：劉明瑋

指 導 教 授：林大衛 博士

中 華 民 國 九 三 年 六 月

纜線數據機的更誤解碼與渦輪等化器研究

**Research in Error-Correction Decoding and Turbo
Equalization for Cable Modem Receivers**

研 究 生：劉明瑋

Student : Ming-Wei Liu

指 導 教 授：林大衛 博士

Advisor : Dr. David W. Lin

國 立 交 通 大 學

電子工程學系 電子研究所碩士班



碩 士 論 文

A Thesis

Submitted to Institute of Electronics

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

in

Electronics Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 三 年 六 月

纜線數據機的更誤解碼與渦輪等化器研究

研究生：劉明璋

指導教授：林大衛博士

國立交通大學
電子工程學系 電子工程研究所

摘要

由於高速傳輸需求的增加新的科技遂興起，一種基於正交調幅調變器 (quadrature amplitude modulation) 纜線數據機 (cable modem) 的網路標準 ITU-T J.83B 產生了。

本論文研究基於 ITU-T J.83B 主要包含兩個方面：更誤解碼與渦輪等化器。

ITU-T J.83B 的前置更誤碼主要包含四個層次，分別為里德-所羅門碼 (Reed-Solomon code)，交錯器 (interleaver)，隨機性發生器 (randomizer)，和格狀編碼調變器 (trellis-coded modulation)。我們首先分別對於每個層次分析它們的特性。對於里德-所羅門碼，我們得到它的錯誤檢查矩陣和理論錯誤率。對於格狀編碼調變器，我們解釋它具有的旋轉不變性，也藉由既有的編碼架構展延出 1024-正交調幅調變器。此外，我們透過其穿孔迴旋碼 (punctured convolutional code) 的重量分配頻譜得到她的理論錯誤率。最後，我們在模擬里德-所羅門碼和在白色高斯雜訊通道下的格狀編碼調變器，這些結果和分析值在高的訊號雜訊比下是相當接近的。

關於渦輪等化器，我們結合格狀編碼調變器和使用最小平均平方演算法 (least mean square) 及多模量演算法 (multi-modulus algorithm) 的等化器得到兩種渦輪等化器的演算法。我們在纜線的通道下模擬而結果證明了使用軟性輸出的維特比解碼器 (soft-output Viterbi decoding) 可以獲得比傳統維特比解碼器多 0.2 dB 增益。

Research in Error-Correction Decoding and Turbo Equalization for Cable Modem Receivers

Student: Ming-Wei Liu

Advisor: Dr. David W. Lin

*Department of Electronics & Institute of Electronics
National Chiao Tung University*



Abstract

Due to the demand for high data transmission rate, a cable modem networks standard, ITU-T J.83B, based on QAM (quadrature amplitude modulation) technique is produced.

This work mainly contains two parts established on ITU-T J.83B: the error correction decoding and turbo equalization for cable modem receivers.

For forward error correction, we analyze the property of the coding schemes respectively containing Reed-Solomon code, interleaver, randomizer and trellis-coded modulation (TCM). As to the extended RS code, we derive its parity check matrix, symbol error rate bound. For the TCM we explain the rotational invariance characteristic of the QAM mapper, and also expand the exiting QAM modes scheme to 1024-QAM. Besides, we acquire its upper bound through the weight distribution spectrum of the punctured convolutional code and modulation mapping. Finally, we simulate the RS code and the TCM under AWGN, and the analytic results are close to the simulation.

Concerning the turbo equalization, we combine the TCM and decision feedback equalizer using least mean square and multi-modulus algorithm to obtain two turbo equalization algorithms. We simulate these algorithms in the cable modem channel, and the results show that soft-output Viterbi decoding gains more than Viterbi algorithm in the decoding of TCM by 0.2 dB.

誌謝

本論文承蒙恩師林大衛教授細心的指導與教誨，方得以順利完成。在研究所生涯的兩年中，林教授不僅在學術研究上給予學生指導，在研究態度亦給許相當多的建議，在此對林教授獻上最大的感謝之意。

此外，感謝洪昆健學長平常能一起討論，並且在我遇到困難時熱心的給予幫助。也要感謝陳繼大學長在程式設計方面不吝指教；吳俊榮學長、林郁男學長與我分享研究與生活方面的經驗；還有筱晴、宗書、盈縈、明哲、建統、仰哲、岳賢等同學、學弟妹與我彼此砥礪、互相討論，讓兩年的研究生涯充滿歡樂與回憶。

同時也要感謝朋友們能包容我不斷的多嘴，在生活中給予支持，使我能在辛苦的研究過程中保持心靈的平衡。

最後，要感謝我的家人。如果我有任何一絲絲成就，你們的栽培與鼓勵是這一切的基石。



要感謝的人很多，僅以這篇論文獻給在研究所生涯中難忘的人，謝謝。

劉明璋

民國九十三年六月 於新竹

Contents

1	Introduction	1
1.1	Scope of the Work	1
1.2	Organization of This Thesis	2
2	Introduction to Cable Modem System	3
2.1	Cable Modem System	3
2.2	Downstream Cable Modem Channel Characteristics	3
2.2.1	Transmission Loss	3
2.2.2	Multipath Reflection	4
2.3	Channel Models	8
3	Algorithm of FEC Encoder	11
3.1	Reed-Solomon Encoder	11
3.1.1	Reed-Solomon Encoder in ITU-T J.83B	12
3.2	Interleaver	13
3.3	Randomizer	14
3.4	Trellis-coded Modulation	15
3.4.1	64-QAM Modulation Mode	15
3.4.2	256-QAM Modulation Mode	15
3.4.3	1024-QAM Modulation Mode	17
3.4.4	Binary Convolutional Coder	17
3.4.5	Rotationally Invariant Pre-coder	18
3.4.6	QAM Mapper	19

3.4.7	Rotational Invariance	19
4	Algorithm of FEC Decoder	27
4.1	Decoding of Trellis-Coded Modulation	27
4.1.1	Decoding Architecture	27
4.1.2	Decoding of Differential Precoder	27
4.1.3	Decoding of PBCC	28
4.1.4	Symbol Error Rate and Bit Error Rate	30
4.2	De-randomizer	36
4.3	De-interleaver	36
4.4	Reed-Solomon decoder	37
4.4.1	Extended RS codes	37
4.5	Concatenated Coding Simulation Results	41
5	Turbo Equalization	44
5.1	Principle of the Transmission Scheme	45
5.2	Turbo-Equalizer with Viterbi Decoder	46
5.2.1	Equalizer Structure	48
5.2.2	Adaptative Algorithm	49
5.2.3	Channel Decoder	58
5.2.4	Simulation Results of Turbo Equalizer with Viterbi Decoder	62
5.3	Turbo Equalizer with Soft-input Soft-output Decoder	65
5.3.1	Symbol to Binary Converter (SBC)	66
5.3.2	Soft-input Soft-output (SISO) Channel Decoder	66
5.3.3	Introduction to SOVA Algorithm	67
5.3.4	Binary to Symbol Converter (BSC)	69
5.3.5	Simulation Results of Turbo Equalizer with SOVA Decoder	70
5.4	Turbo Equalizer Conclusion	71
6	Conclusion and Future Work	73

List of Tables

2.1	Maximum Loss for Drop Cable (dB/100ft at 68F) with Different Cable Diameters and Frequency	5
2.2	Multipath Echo Model According to DOCSIS 1.0 Standard	6
2.3	Cable Propagation Models	9
3.1	Differential Encoding Rule for QAM	22
4.1	Weight Distribution of the PBCC in TCM	34
5.1	Conditions of Simulation 0.1	51
5.2	Summary of Blind Algorithms	55
5.3	Conditions of Simulation 1.1	57
5.4	Conditions of Simulation 2.1	62
5.5	Conditions of Simulation 3.1—Turbo Equalizer	64
5.6	Complexity Comparison of MAP and SOVA Algorithms Implemented in Turbo Equalization (from [22])	67
5.7	Conditions of Simulation 4.1.—Turbo Equalizer.	70
5.8	Conditions of Simulation 4.2.—Turbo Equalizer.	72

List of Figures

2.1	A cable plant model.	7
2.2	A drop configuration.	8
2.3	Magnitude and impulse responses of filtered CH0.	10
3.1	Layers of the FEC in J.83B.	12
3.2	Interleaving functional block diagram [1, p. 22].	14
3.3	Randomizer (7-bit symbol scrambler [1, p. 25]).	15
3.4	TCM encoder for 64-QAM [1, p. 26].	16
3.5	TCM encoder for 256-QAM [1, p. 28].	16
3.6	Proposed TCM encoder for 1024-QAM.	17
3.7	Punctured binary convolutional encoder.	18
3.8	Differential precoder.	19
3.9	64-QAM constellation [1, p. 32].	20
3.10	256-QAM constellation [1, p. 33].	21
3.11	Symbol map.	22
3.12	256-state trellis.	23
3.13	256-state trellis.	23
3.14	Mapping of coded bit along in-phase axis of 64-QAM mapper.	24
3.15	Mapping of uncoded bits in the first quadrant of 64-QAM.	25
3.16	Mapping of uncoded bits in the first quadrant of 256-QAM.	25
3.17	Mapping of uncoded bits in other quadrants by rotating the mapping in the first quadrant.	26
4.1	TCM decoding architecture.	28

4.2	Performance of two 16-state decoders and one 256-state viterbi decoder in AWGN.	29
4.3	Effect on SER of Viterbi decoding depth.	30
4.4	State transition diagram of example the PBCC.	35
4.5	SER performance of TCM in 64-QAM, 256-QAM, and 1024-QAM modes.	35
4.6	BER performance of TCM in 64-QAM, 256-QAM, and 1024-QAM modes.	36
4.7	(a) Performance of $t=2$ and $t=3$ for RS code. (b) Performance of $t=3$ with correct parity check symbol and ideal block code.	40
4.8	64-QAM trellis group [1, p. 27].	41
4.9	(a) SER of 64-QAM FEC overall simulation results. (b) BER of 64-QAM FEC overall simulation results.	43
5.1	Principle of the transmission scheme.	46
5.2	Turbo equalizer principle.	47
5.3	Schematic diagram of module $p - 1$	47
5.4	Proposed turbo equalizer structure.	47
5.5	DFE structure.	48
5.6	Adaptive DFE structure.	50
5.7	Simulation 0.1. LMS with variable step size.	52
5.8	Principle of blind algorithms. (a) RCA. (b) CMA. (c) MMA.	54
5.9	MMA adaptive filter using DFE.	56
5.10	Simulation 1.1 Received symbols.	58
5.11	Simulation 1.1 of MMA. (a) Before convergence. (b) After convergence.	59
5.12	Simulation 1.1 of CMA. (a) Before convergence. (b) After convergence.	60
5.13	Simulation 2.1 of MMA. (a) Case 1, input SNR = 25 dB. (b) Case 2, noise-free.	61
5.14	SNR rise between MMA and LMS algorithms.	63
5.15	Simulation 3.1. Simultaion results of turbo equalizer with Viterbi decoder.	65
5.16	Schematic diagram of module $p - 1$	65

5.17	Example of symbol to binary converter for 64-QAM.	66
5.18	Transfer function from LLR to estimated symbol.	69
5.19	Simulation 4.1. Simulation results of turbo equalizer with SOVA decoder.	71
5.20	Simulation 4.1. Simulation results of turbo equalizer with SOVA decoder.	72



Chapter 1

Introduction

1.1 Scope of the Work

Due to the vigorous development of multi-media, the demand for high data transmission rate increases considerably. Since the general voice modem can not satisfies the data rate requirement, new technology like cable modem rises to offer high speed connections. Cable modem technology contains three main streams: DOCSIS, DAVIC, and IEEE 802.14, and our study is based on the DOCSIS standard (J.83B).

This work studies two parts of J.83B: one is the forward error correction and the other is the turbo equalization.

The forward error correction in J.83B consists of four layers as Reed-Solomon (RS) coding, interleaving, randomization, and trellis coding. We analysis the properties of each layer and then simulate by personal computer. The main challenges of the forward error correction consist in:

- The extend Reed-Solomon code is not a conventional Reed-Solomon code. Its single parity check byte extends the classical code by one byte.
- The trellis-coded modulation (TCM) is a combination of modulation and error control coding. Analysis of it requires jointly consider the contained punctured high rate convolutional code and the modulation mapper.
- Extend the coding scheme from the existing modulation modes to higher level mod-

ulation mode.

- The construction and simulation of the encoders and decoders in each layer.

For turbo equalization, we first simulate the decision feedback equalizer with training based least-mean-square (LMS) algorithm and different blind equalization algorithms. Then we combine the TCM and equalizer to implement a turbo equalizer. Furthermore, we employ the soft-output Viterbi algorithm (SOVA) to substitute the conventional Viterbi algorithm and derive the improvement.

1.2 Organization of This Thesis

Organization of this thesis is:

- Chapter 1 is the scope of the work and the organization of this thesis.
- Chapter 2 introduces the cable modem system and its downstream channel characteristics.
- Chapter 3 briefly describes the encoding scheme of J.83B and analysis its property.
- Chapter 4 presents the decoding algorithms, their theoretical bounds, and simulation results.
- Chapter 5 contains two different turbo equalization algorithms, the simulation results and comparison.
- Chapter 6 sums up the conclusion and the future work.

Chapter 2

Introduction to Cable Modem System

2.1 Cable Modem System

With the increasing demands from users, the capacity of CATV changes from several channels to more than 100 channels. The transmission network also alters from star/tree coaxial network architecture to hybrid fiber coaxial (HFC) cell based network. The section closer to the end users and away from the head-end uses coaxial cables for signal transmission.

The coaxial section architectures combination of three hierarchical system: trunk system, distribution (feeder) system and drop system. Our study concerns the coaxial section of the plant.

2.2 Downstream Cable Modem Channel Characteristics

2.2.1 Transmission Loss

The primary loss mechanisms in coaxial cables are frequency and temperature dependency of the inner conductor and the dielectric outer conductor losses. The frequency dependent transmission loss at RF frequency is influenced by the skin effect. While DC the current flows uniformly through the cross-section of the conductor, the current tends to crowd around the conductor surface as the RF current is increased. Companioned with

higher frequency, this effect increases impedance of a given conductor. A useful parameter is cable loss ratio (CLR) given by

$$CLR = \sqrt{\frac{f_1}{f_2}} \quad (2.1)$$

where f_1 and f_2 are two different frequencies.

Generally speaking, cable plants are designed to operate over a wide range or temperature from -40 to 70 centigrade degree. With the temperature increases, the attenuation increases, resulting in the cable loss slowly and linearly increases [2, pp. 43–44].

Trunk and Feeder Cable

To provide low RF loss, the cables with outer shield diameters ranging from 0.412 to 1.125 inches are used. Common diameters of this family of cables include 0.412, 0.500, 0.625, 0.750, 0.825, and 1.000 inch. They are generally labeled by the outer shield diameter in thousands of an inch; for example “500 cable” refers to a cable of 0.500 inch. Larger cables are used in longer distance connection, and smaller cables are used in shorter distance connection where the main losses are tap losses rather than cable losses [3, pp. 395–396]. Taps are used to split the transmitted signals to drop cables.

Drop Cable

Drop cable has a smaller diameter than the feeder cable, and is used between the tap and subscriber home terminal. Table 2.1 shows the drop cable loss (dB/100feet) at 20 degree centigrade. The diameters of this family of drop cables include 0.240 (59 series foam), 0.272(6 series foam), 0.318 (7 series foam), and 0.395 (11 series foam) inch [2, pp. 43–44].

2.2.2 Multipath Reflection

The transmitted signal is partially reflected at where there is mismatch, resulting in prolonged channel impulse response. The reflection is due to impedance mismatch at various places along the transmission path, caused by cable imperfections and cable junctions, as

Table 2.1: Maximum Loss for Drop Cable (dB/100ft at 68F) with Different Cable Diameters and Frequency

Frequency (MHz)	59Series Form	6Series Form	7Series Form	11Series Form
5	0.86	0.58	0.47	0.38
30	1.51	1.18	0.92	0.71
40	1.74	1.37	1.06	0.82
50	1.95	1.53	1.19	0.92
110	2.82	2.24	1.73	1.36
174	3.47	2.75	2.14	1.72
220	3.88	3.11	2.41	1.96
300	4.45	3.55	2.82	2.25
350	4.80	3.85	3.05	2.42
400	5.10	4.15	3.27	2.26
450	5.40	4.40	3.46	2.75
550	5.95	4.90	3.85	3.04
600	6.20	5.10	4.05	3.18
750	6.97	5.65	4.57	3.65
865	7.52	6.10	4.93	3.98
1000	8.12	6.55	5.32	4.35

Table 2.2: Multipath Echo Model According to DOCSIS 1.0 Standard

Echo Time Delay	Echo Magnitude (downstream)	Echo Magnitude (upstream)
0 to 0.5 μs	-10 dBc	-10 dBc
$\leq 1.0 \mu s$	-15 dBc	-20 dBc
$\leq 1.5 \mu s$	-20 dBc	-30 dBc
$> 1.2 \mu s$	-30 dBc	-30 dBc

well as the inevitably used splitters and taps. This can be seen on an analog TV as ghosting, or it can result in a loss of the receiver synchronization in the digitally demodulated picture.

Multipath reflections can be presented as:

$$h(t) = \sum_i A_i \delta(t - t_i) e^{j\phi_i} \quad (2.2)$$

or

$$H(\omega) = \sum_i A_i e^{j(\omega t_i + \phi_i)}. \quad (2.3)$$

There are various discrete multipath reflection models such as IEEE 802.14 and DOCSIS 1.0 [2, p. 55] for both forward and return-path channels. While DOCSIS 1.0 model assumes a single dominant echo, the IEEE 802.14 assumes multiple echoes within the time-delay range. Table 2.2 shows the DOCSIS 1.0 model with maximum echo power and delay time.

Trunk Amplifier, Bridger Amplifier, and Line Extender

Trunk amplifiers, spaced 20–22 dB from one another, are moderate-gain amplifiers with a typical output of 30–36dBmV that are used to provide high CNR with low nonlinear distortion (-80 dBc) [2, p. 44]. Amplifiers are spaced in about 2000-ft depending on the bandwidth.

The output power of bridger amplifier is in the range of 40–50 dBmV, but higher nonlinear distortions also exist [2, p. 44]. The nonlinearity in line extender amplifiers is also high. To reduce the effect of nonlinear distortions, a maximum of two to four

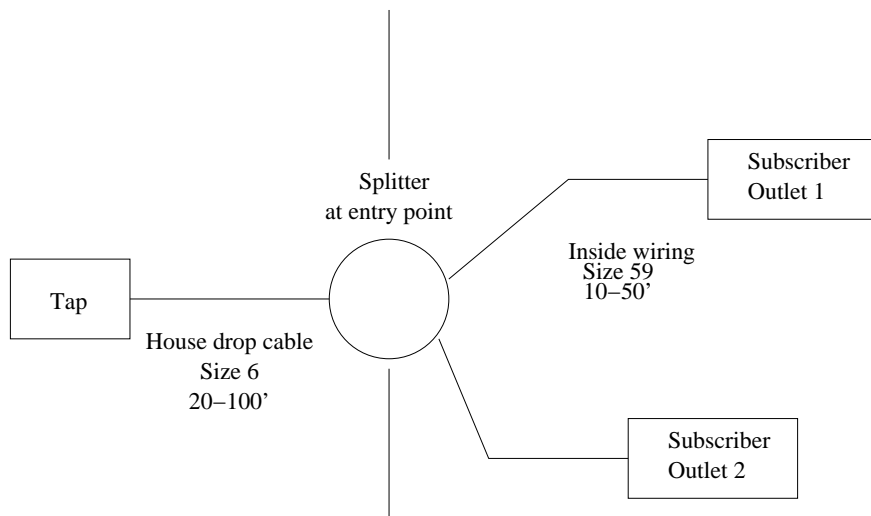


Figure 2.2: A drop configuration.

transmission loss between two trunk amplifiers. The space between two taps is 100 to 180 feet [5, pp. 305–306]. It is reasonable since in North America, many dwelling sections are 100 feet square blocks. The distance of line extenders is also reasonable for the requirement of 120 to 350 m mentioned above. 750 cables are used in feeder cables, 6-Series is used in drop cables, and 59-Series is used in home wiring.

2.3 Channel Models

Now we use the channel model shown in Fig. 2.1 to derive some channel models as shown in Table 2.3. We transmit an impulse from the transmitter and chose an outlet in home to be the receiver. The return loss of taps are assumed to be 16 dB, of outlet to be 4 dB, and the directivity is 10 dB. Since an outlet may be left unterminated, so it should be assumed that 100% reflection occurs at each subscriber outlet. Besides, the length of drop cables are different and the taps are assumed four-way 20-dB taps with through loss 1 dB [2, p. 46], [3, pp. 560-567].

Fig. 2.3 shows the frequency response and channel response of CH0 in Table 2.3. This figure is drawn by two times oversampling. And the frequency response is the equivalent low-pass signal.

Table 2.3: Cable Propagation Models

	Magnitude	Delay (ns)		Magnitude	Delay (ns)
CH0	0,9438	0	CH3	1	0
	0.1263	161.15		0.1043	210.7
	0.1699	313.26		0.0028	421.4
	0.0622	-0.1696		0.0037	481.1
				0.0093	511.7
CH1	1	0	CH4	1	0
	0.1411	90.3		0.0151	331.1
	0.0063	150.5		0.0119	391.3
	0.0255	180.6		0.0076	421.4
				0.0021	451.5
CH2	1	0			
	0.1431	90.3			
	0.0050	174.58			
	0.0114	180.6			
	0.0052	189.6			
	0.0049	195.65			

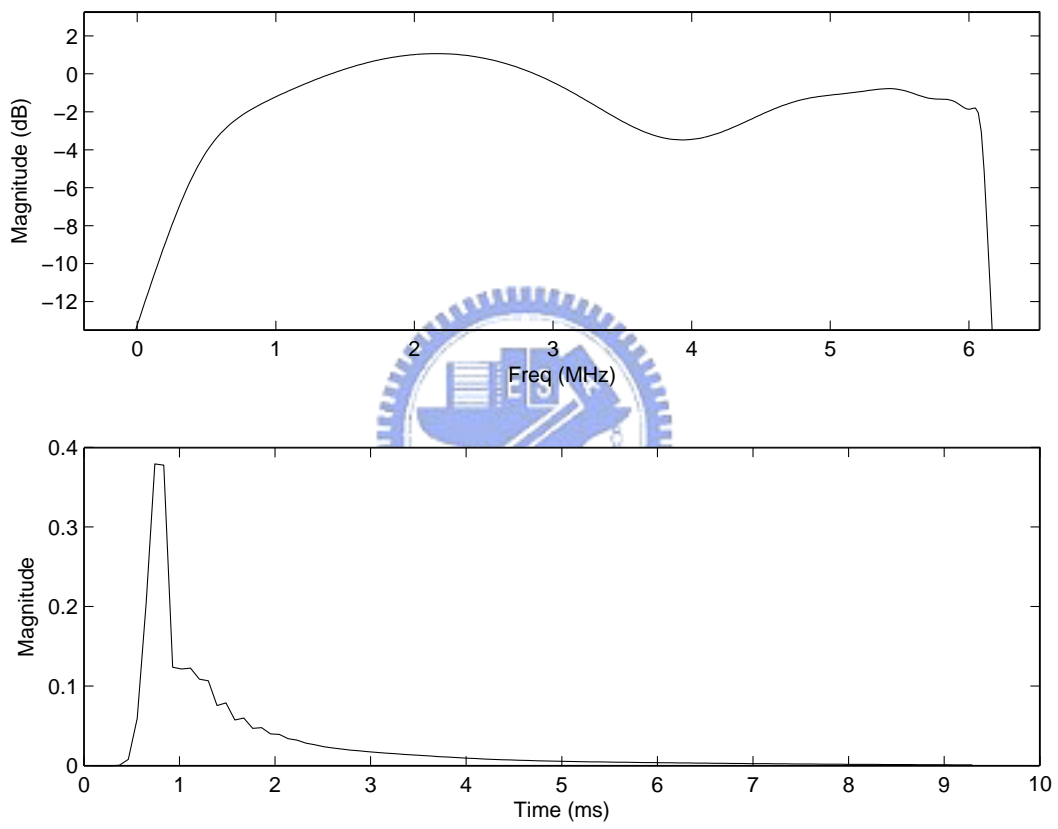


Figure 2.3: Magnitude and impulse responses of filtered CH0.

Chapter 3

Algorithm of FEC Encoder

The forward error correction (FEC) definition is composed of four layers in J.83B, as shown in Fig. 3.1. The FEC section uses various types of error correction algorithms and interleaving techniques by the cable channel situation.

- Reed-Solomon (RS) Coding — Provides block coding and decoding.
- Interleaving — Disperses the symbols to protect bursty errors.
- Randomization — Randomizes the data from interleaver to admit effective QAM demodulator synchronization.
- Trellis Coding — Combines the punctured convolutional encoding and modulation.

3.1 Reed-Solomon Encoder

Due to the enormous advances in the digital techniques for computers, new digital communication systems are now rapidly replacing the former analogue systems. The Reed-Solomon code is one of the most powerful techniques to ensure the completeness of transmitted or stored digital information.

The Reed-Solomon codes are nonbinary codes with code symbols from a Galois field. They were discovered in 1960 by I. Reed and G. Solomon. the work was done when they were at MIT Laboratory. In the decades since their discovery, RS codes have had

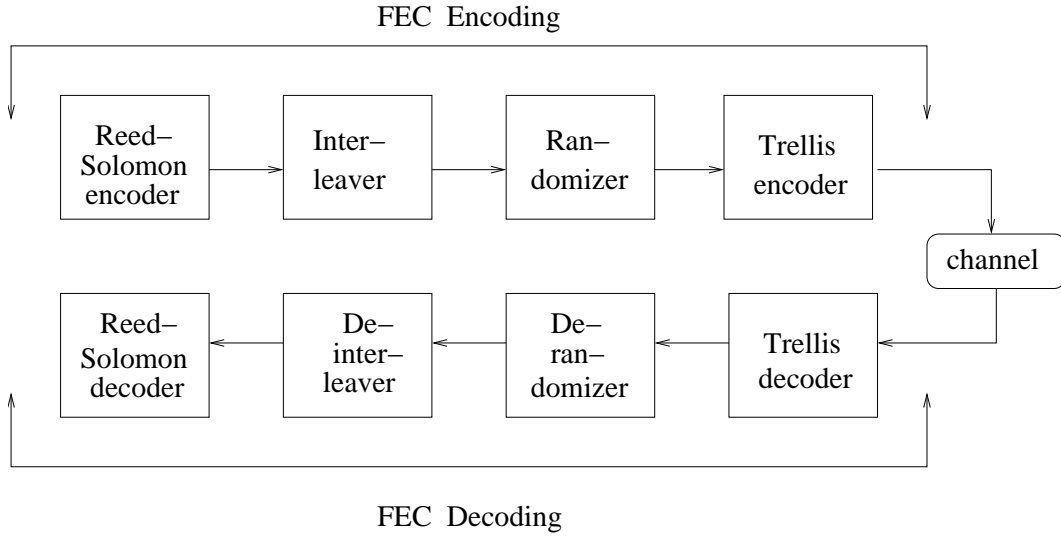
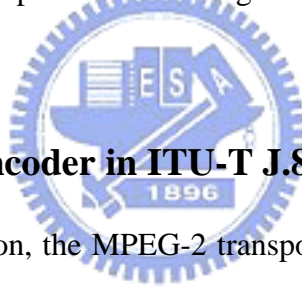


Figure 3.1: Layers of the FEC in J.83B.

countless applications from compact discs and digital TV in living room to space craft and satellite in outer space.



3.1.1 Reed-Solomon Encoder in ITU-T J.83B

In ITU-T J.83B recommendation, the MPEG-2 transport stream is Reed-Solomon (RS) encoded using $(128, 122)$ code with capability of correcting up to $t = 3$ symbol errors per block over $GF(128)$. Although the FEC frame format differs between 64-QAM and 256-QAM, each modulation type utilizes the same RS code.

A systematic encoder implements a $t = 3$, $(128, 122)$ extended RS code over $GF(128)$. The primitive polynomial forms the field over $GF(128)$ as:

$$p(x) = x^7 + x^3 + 1$$

where

$$p(\alpha) = 0.$$

The generator polynomial is:

$$\begin{aligned} g(x) &= (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)(x + \alpha^5) \\ &= x^5 + \alpha^{52}x^4 + \alpha^{116}x^3 + \alpha^{119}x^2 + \alpha^{61}x + \alpha^{15}. \end{aligned} \quad (3.1)$$

The message polynomial consists of 122, 7-bit symbols, and is described as below:

$$m(x) = m_{121}x^{121} + m_{120}x^{120} + \dots + m_1x + m_0.$$

The message polynomial is first multiplied by x^5 , then divided by the generator polynomial $g(x)$ to form a remainder, described by the following:

$$r(x) = r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0.$$

The remainder polynomial provides 5 parity symbols, and then is added to the message polynomial to form a 127-symbols codeword.

The following polynomial describes the generated code word.

$$c(x) = m_{121}x^{126} + m_{120}x^{125} + m_{119}x^{124} + \dots + r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0.$$

An extended code, c_- , is evaluated through the code word at the sixth power of α .

$$c_- = c(\alpha^6). \quad (3.2)$$

This extended code is then added to the last symbol of a transmitted Reed-Solomon block. The extended code word appears as follows:

$$\begin{aligned} \hat{c} &= xc(x) + c_- \\ &= m_{121}x^{127} + m_{120}x^{126} + \dots + m_1x^7 + m_0x^6 + r_4x^5 + r_3x^4 + r_2x^3 + r_1x^2 + r_0x + c_-. \end{aligned} \quad (3.3)$$

The order of transmitted RS code block from the encoder output is constructed as (order sent is left to right):

$$m_{121}m_{120}m_{119} \dots m_1m_0r_4r_3r_2r_1r_0c_-.$$

3.2 Interleaver

Interleaving is a form of time diversity that mitigates the effects of bursty errors. Several techniques aim at reducing channel effects either by supplying block interleaving or convolutional interleaving. A channel is considered fully interleaved when consecutive symbols

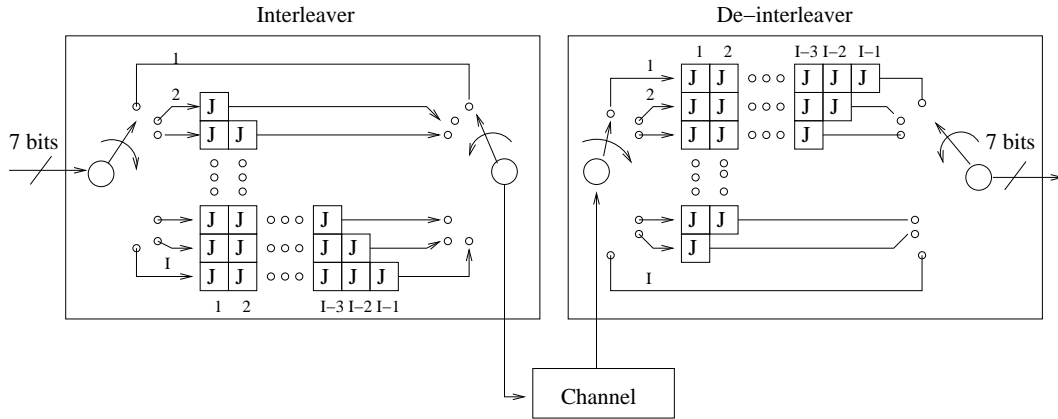


Figure 3.2: Interleaving functional block diagram [1, p. 22].

are rearranged to be affected independent by channel errors and partially interleaved when consecutive symbols of the received sequence are affected by the same bursty errors.

Both 64-QAM and 256-QAM employ a convolutional interleaver between the RS block coding and the randomizer. As Fig. 3.2 depicts, the interleaving commutator initially points to the top-most branch, and the RS code symbols are sequentially shift into the bank of I registers. The first interleaver has zero delay, the second one has a J symbol period of delay, the third has $2 * J$ symbol periods of delay, and so on.

To resist different channel conditions, users can choose only one depth of interleaving in 64-QAM ($I = 128, J = 1$), while five alternative modes in 256-QAM.

3.3 Randomizer

Both 64-QAM and 256-QAM introduce a randomizer shown in Fig. 3.3 to uniformize the distribution in the constellation. The randomizer adds a Pseudorandom Noise (PN) sequence of 7-bit symbols over $GF(128)$ (i.e., bit-wise exclusive-OR) to the symbol from the convolutional interleaver.

The randomizer is initialized as pre-loading to the “all-ones” state during the FEC frame trailer, and is enabled at the first data symbol.

A linear feedback shift register is applied to specify a $GF(128)$ polynomial defined as follows:

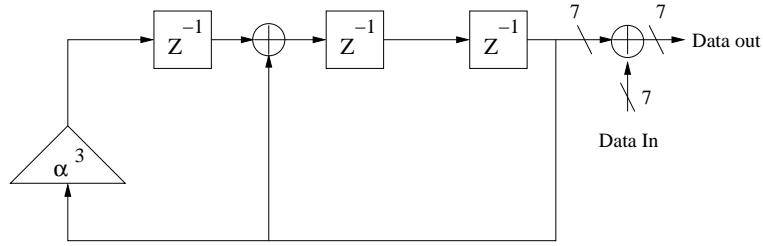


Figure 3.3: Randomizer (7-bit symbol scrambler [1, p. 25]).

$$f(x) = x^3 + x + \alpha^3$$

where

$$\alpha^7 + \alpha^3 + 1 = 0.$$

3.4 Trellis-coded Modulation

3.4.1 64-QAM Modulation Mode

For 64-QAM, the input of the trellis coded modulator is a 28-bits sequence labeled in pairs of A symbols and B symbols as shown in Fig. 3.4. As depicted, all 28 bits are assigned to a trellis group of five 6-bits QAM symbols and therefore the overall rate of the TCM code is $\frac{28}{30}$. The TCM encoding can be approximately divided into three parts which will be discussed in further detail later. The first part is the differential precoder used to produce $\frac{\pi}{2}$ rotational invariance, the second part is a rate- $\frac{1}{2}$ binary convolutional encoder with $\frac{4}{5}$ puncturing (PBCC), and the third part is the QAM mapper.

3.4.2 256-QAM Modulation Mode

256-QAM modulation mode employs a similar trellis coding as 64-QAM with the same PBCC. As description in Fig. 3.5, all 38 bits are assigned to a trellis group of five 8-bits QAM symbols and the overall rate of the TCM code is therefore $\frac{38}{40}$.

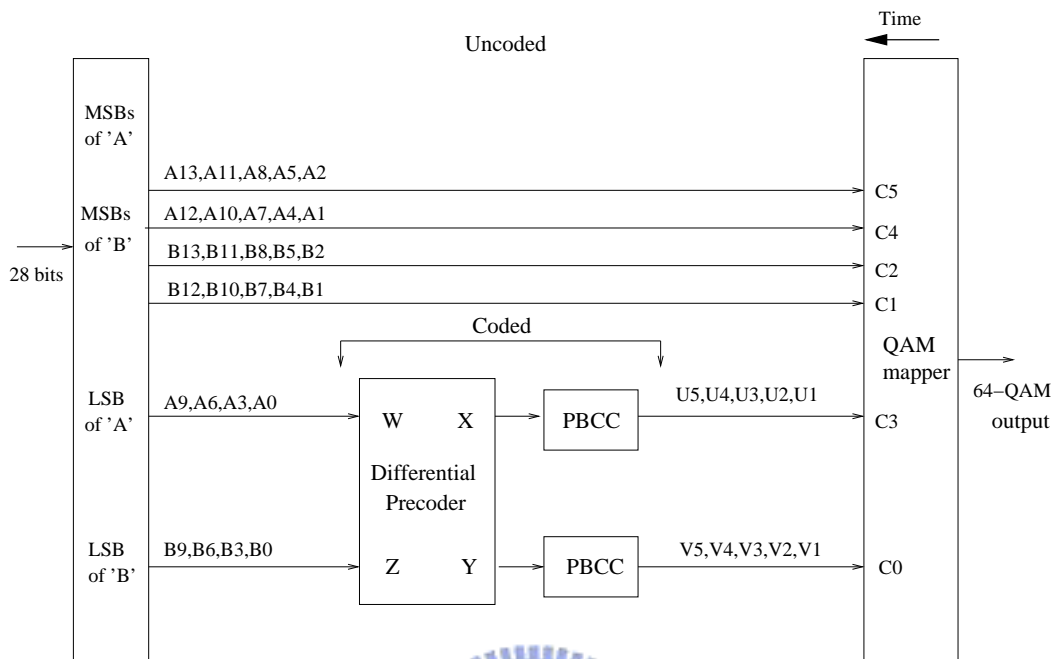


Figure 3.4: TCM encoder for 64-QAM [1, p. 26].

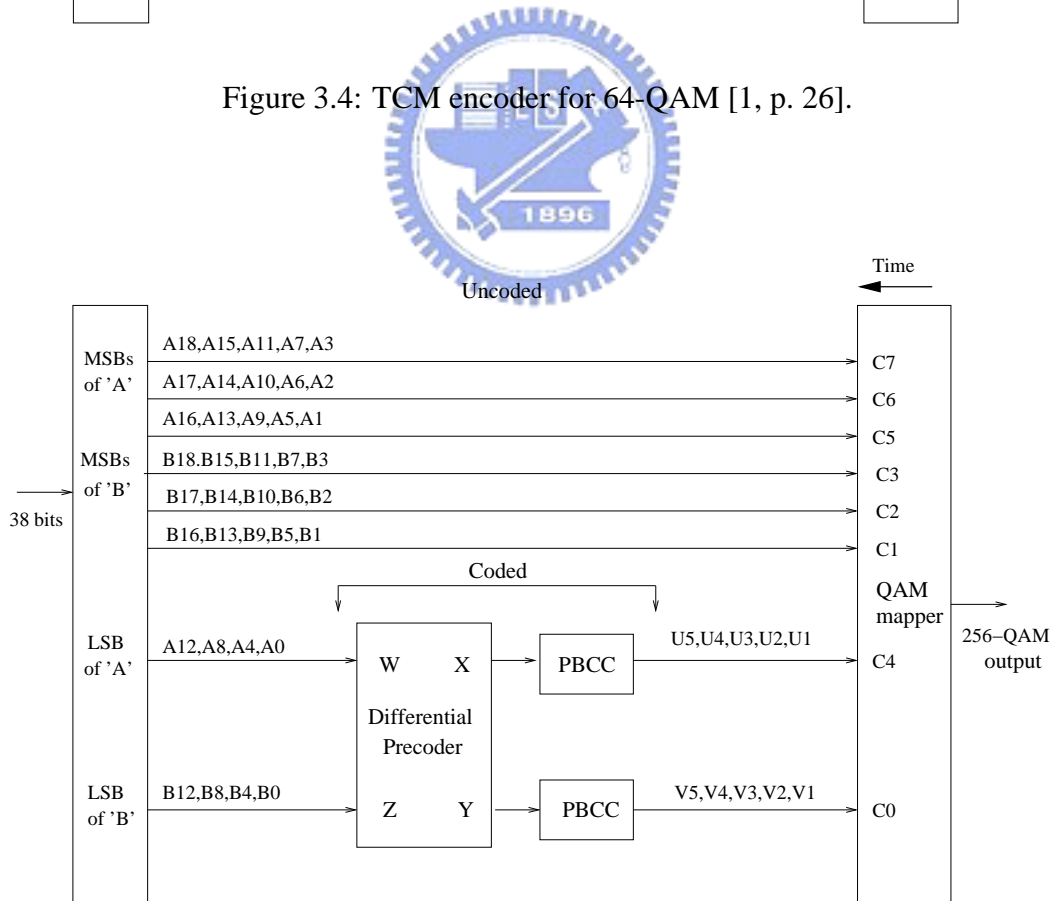


Figure 3.5: TCM encoder for 256-QAM [1, p. 28].

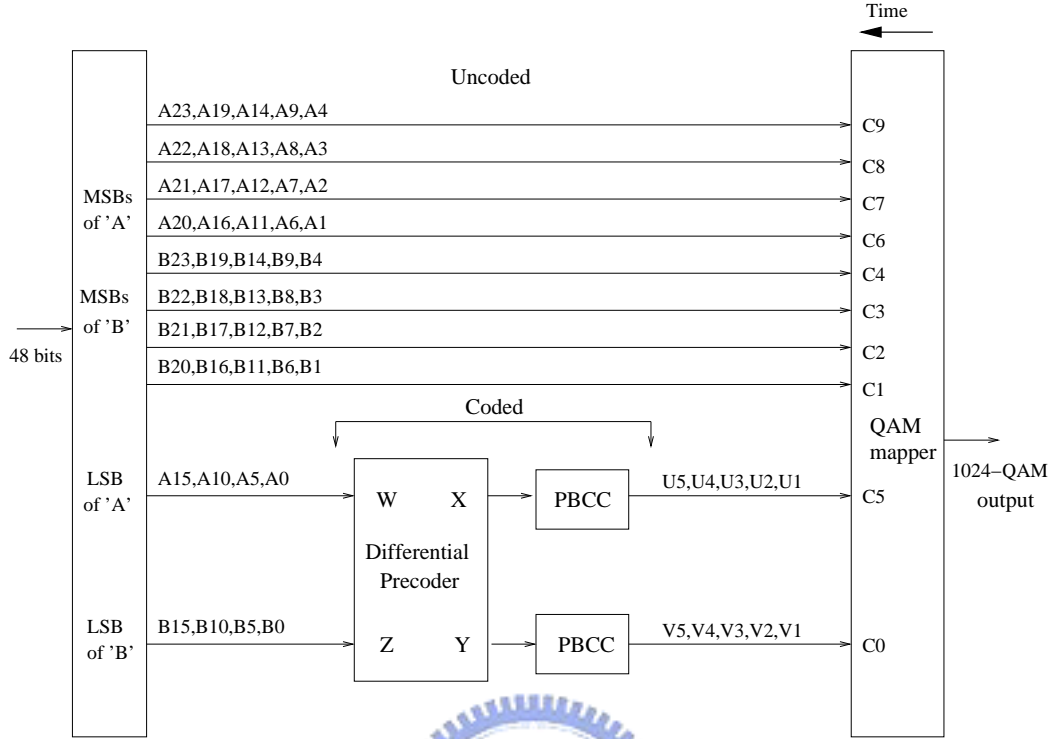


Figure 3.6: Proposed TCM encoder for 1024-QAM.

3.4.3 1024-QAM Modulation Mode

Although ITU-T J.83B recommendation only support 64-QAM and 256-QAM modulation modes, we can develop a 1024-QAM modulation mode from its regular coding rule of the two existing modes for further study. The 1024-QAM mode is shown in Fig. 3.6. We label and order the input sequence in pairs of A symbols and B symbols like those in 64-QAM and 256-QAM modes, and 48 bits are group into five 10-bits QAM symbols. The overall rate of the 1024-QAM modulation mode TCM codes are $\frac{48}{50}$. Furthermore, we utilize the same differential precoder and the same PBCC.

3.4.4 Binary Convolutional Coder

The TCM includes a rate- $\frac{5}{4}$ binary convolutional code based on a rate- $\frac{1}{2}$ convolutional encoder to introduce redundancy into the LSBs of the trellis group. The rate- $\frac{1}{2}$ convolutional encoder is a 16-state encoder with generator $G=[25,37]$ (octal) and puncturing matrix $P=[0001;1111]$ (where “0” denotes punctured bit positions). The structure of the

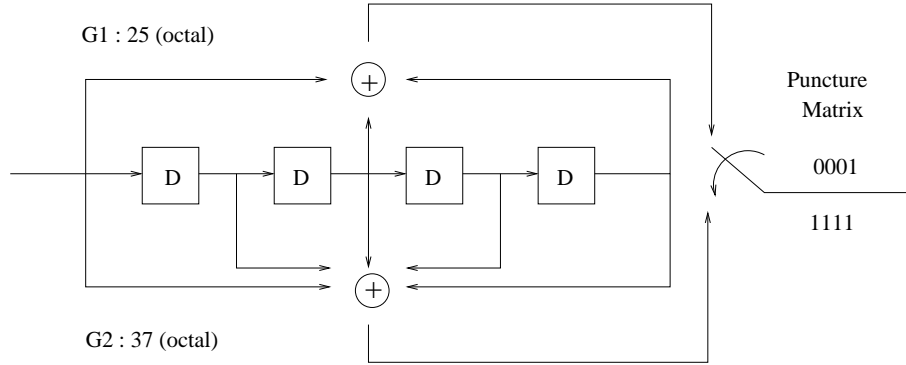


Figure 3.7: Punctured binary convolutional encoder.

punctured binary convolutional coder is shown in Fig. 3.7.

We notice from Figs. 3.4, 3.5, and 3.6 that there are two separate PBCC coding branches. Therefore, we can treat the QAM as two separate PAMs. According to this characteristic, the complexity of analysis can be reduced to two 2^4 -state trellises instead of a 2^8 -state trellis. The simulation proof will be discussed in next chapter.

3.4.5 Rotationally Invariant Pre-coder

Both 64- and 256-QAM modulation use the differential pre-coder to cause the trellis-coded modulation 90° rotationally invariant. With the differential precoder, the information is carried in the phase change rather than the absolute phase.

The LSB of the in-phase and the quadrature components are differentially encoded as follows:

$$X_j = W_j + X_{j-1} + Z_j(X_{j-1} + Y_{j-1}), \quad (3.4)$$

$$Y_j = Z_j + W_j + Y_{j-1} + Z_j(X_{j-1} + Y_{j-1}), \quad (3.5)$$

where the subscripts are time indexes, as illustrated in Fig. 3.8. In 64-QAM constellation, as shown in Fig. 3.9, C_0 and C_3 are differentially encoded, while in 256-QAM constellation, in Fig. 3.10, C_0 and C_4 are encoded instead.

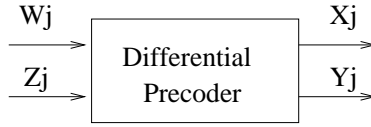
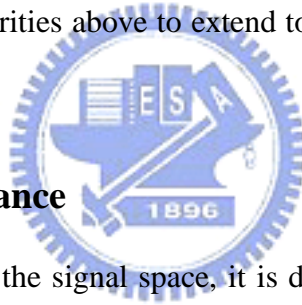


Figure 3.8: Differential precoder.

3.4.6 QAM Mapper

To implement rotationally invariant TCM, we use a certain mapping pattern [9], [10]. The way to map has two steps. From Figs. 3.4 and 3.5, we see that, for each QAM symbol, only two bits are encoded while the rest are not. Hence, we map the coded and the uncoded bits separately. The coded bits are mapped according to set partitioning with two levels, which results in the enlargement of the minimum free distance [9], [10]. Secondly, the uncoded bits are mapped with rotational invariance.

We can extend those regularities above to extend to larger constellations, like 1024-QAM.



3.4.7 Rotational Invariance

Due to the phase ambiguity in the signal space, it is desirable to design the code to be transparent to signal element rotations. The trellis-coded modulation in J.87B ingeniously involves this characteristic called rotational invariance. This characteristic is useful in blind equalization and will be discussed in next chapter. We analysis the composition elements of rotational invariance in the following discussion.

Differential Precoder

The physical meaning of (3.4) and (3.5) can be understood with the help of Fig. 3.11 and Table 3.1. Table 3.1 and Fig. 3.11 explain the relationship between the input bit pair and the corresponding phase change from the previous symbol. For instance, let the previous output bit pair be $(X_{j-1}, Y_{j-1}) = (0, 1)$, and let the current input bit pair be $(Z_j, W_j) = (0, 1)$. Then form Table 3.1, the phase rotate $\frac{\pi}{2}$ clockwise and obtain the current output bit pair as $(X_j, Y_j) = (1, 1)$. By this, an error caused by a carrier phase rotation will not propagate.

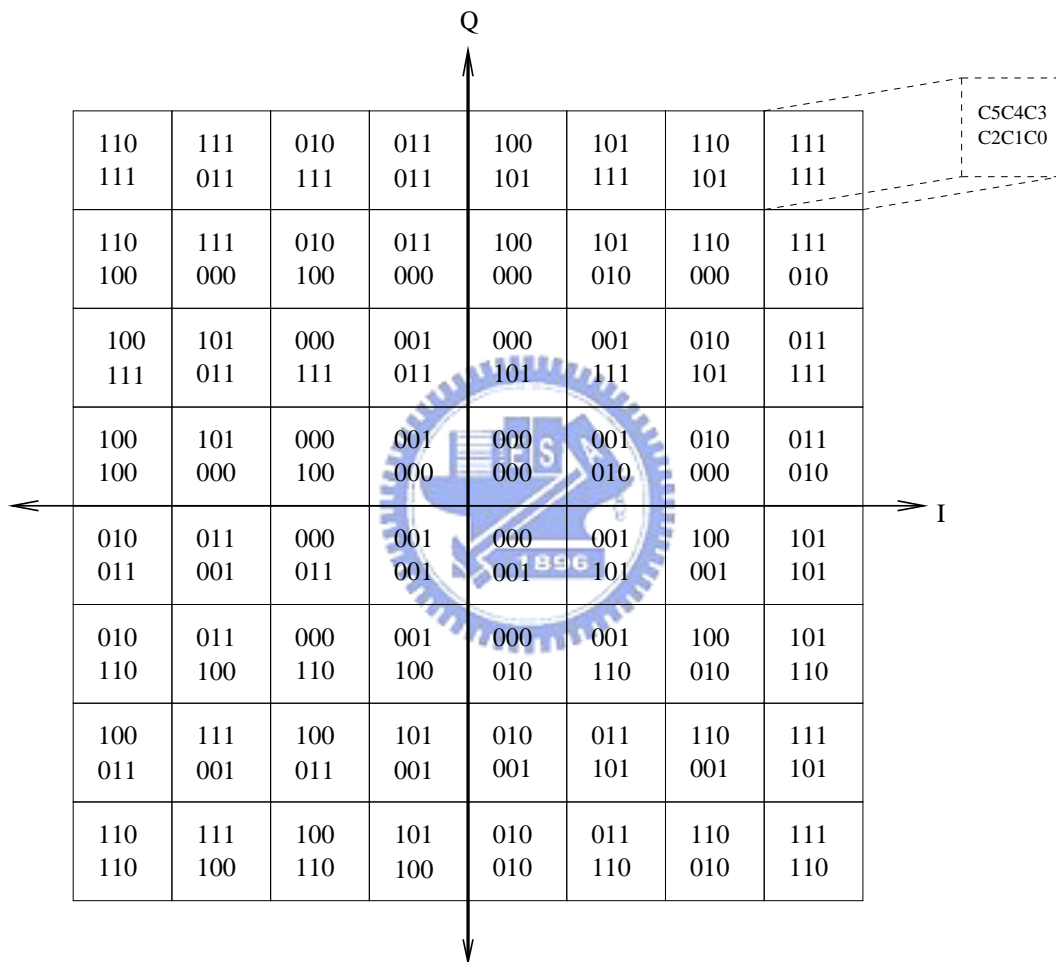


Figure 3.9: 64-QAM constellation [1, p. 32].

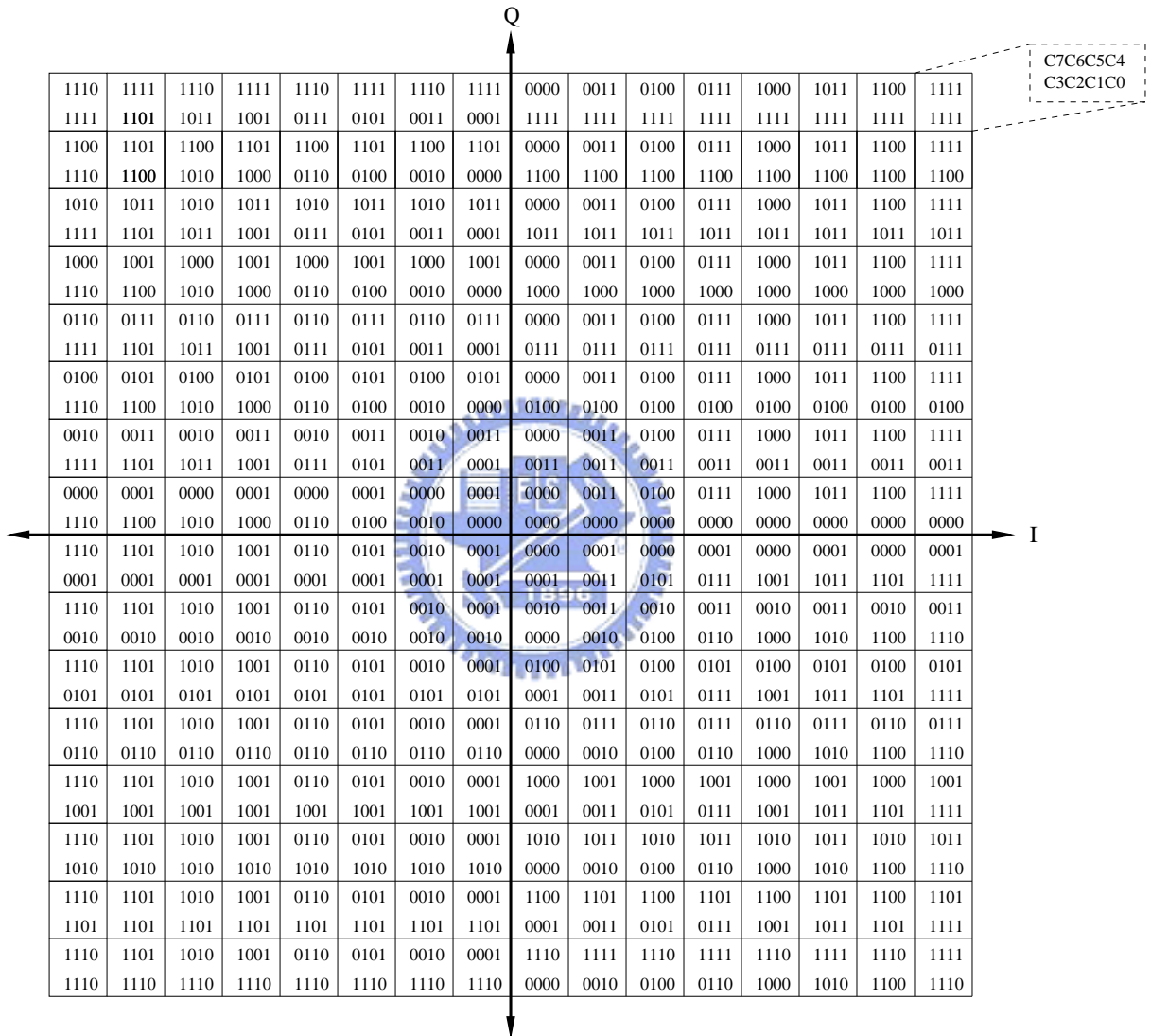


Figure 3.10: 256-QAM constellation [1, p. 33].

Table 3.1: Differential Encoding Rule for QAM

$W_j Z_j$	Phase Change from Previous Symbol
00	0
01	$\frac{\pi}{2}$
10	π
11	$-\frac{\pi}{2}$

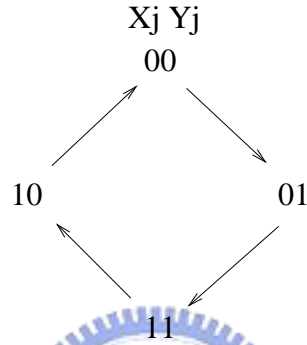


Figure 3.11: Symbol map.

PBCC

To see why the two PBCCs can satisfy rotational invariance, consider a part of the 2^8 -state trellis of the encoder in Figs. 3.12, 3.13

In Fig. 3.13, $S_i, i = 0, 1, \dots, 7$, represent the state of the convolutional encoder. Let the current transmitted signal be A and the next signal be C as shown in the figure. If the received signals are rotated clockwise by $-\frac{\pi}{2}$, then they will become B and D instead as shown in Fig. 3.12. The code is rotationally invariant because for each path $A \rightarrow C \rightarrow \dots$ there exists a valid path $B \rightarrow D \rightarrow \dots$ of $-\frac{\pi}{2}$ -rotated symbols through the trellis [7, p. 463].

QAM Mapper

To analyze the mapping, we first disregard the uncoded bits and examine the in-phase axis of the QAM mapper given in (3.9) and (3.10), from [1, pp. 32, 33]. The coded bits alternates in the fashion shown in Fig. 3.14. By this mapping methodology, the minimum

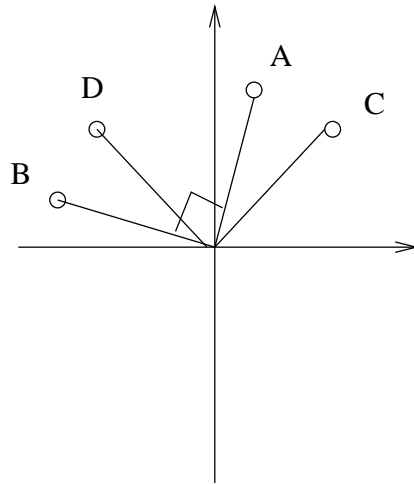


Figure 3.12: 256-state trellis.

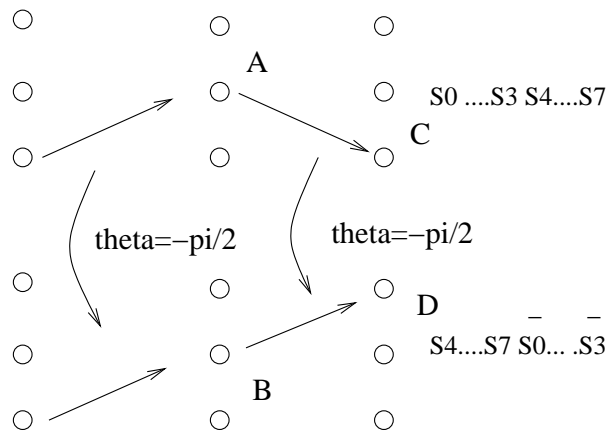


Figure 3.13: 256-state trellis.

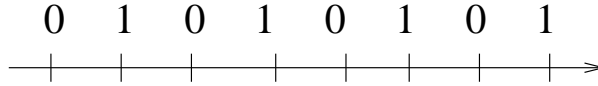


Figure 3.14: Mapping of coded bit along in-phase axis of 64-QAM mapper.

intra-set (within set of 0s or set of 1s) distance is enlarged, which is important considering the spirit of set partitioning. The quadrature axis is the same. Consequently, by the binary convolutional encoding, we can make the coded signals have a larger minimum free distance than the original so as to reduce the symbol error rate at the same SNR.

Now we disregard the coded bits and examine the mapping of the uncoded bits. In the first quadrant of the 64-QAM mapper shown in (3.9), we can discern a pattern of bit mapping consisting of four symbols as a unit and four units organized a quadrant, depicted in Fig. 3.15. As presented in the figure, those uncoded bits are arranged in a regular form which could be used to understand the regularity of the mapping of 256-QAM in Fig. 3.10.

In 256-QAM constellation, a similar mapping rule is used as shown in Fig. 3.16. Different from 64-QAM mapper, the bits arranged in the same unit is not both As or Bs in Fig. 3.5 while in 64-QAM both are.

But how to make the mapping rotationally invariant? For this, if the constellation is rotated by $k\frac{\pi}{2}$, k an integer, the mapping must appear like it is not rotated. For the coded bits, the differential precoder is used to achieve rotational invariance. For the uncoded bits, we just rotate the bits mapping in the first quadrant by $k\frac{\pi}{2}$ counterclockwise to get mapping in the k th quadrant, as shown in Fig. 3.17. (For convenience, we have used decimal numbers in place of binary numbers in this figure, C_5, C_4, C_2, C_1 as the MSB to LSB.) By this, if the constellation is rotated by $k\frac{\pi}{2}$, where k is any integer, the mapping of the uncoded bits are still the same as if it is not rotated.

Note that the mapping in the four-symbol unit follows Gray coding. In addition, each partitioned subset in the second level, or third level in 256-QAM, has Gray-coded bit mapping. Therefore, the intra-set or inter-set symbols errors correspond to a single bit error, which is the minimum amount of bit errors possible.

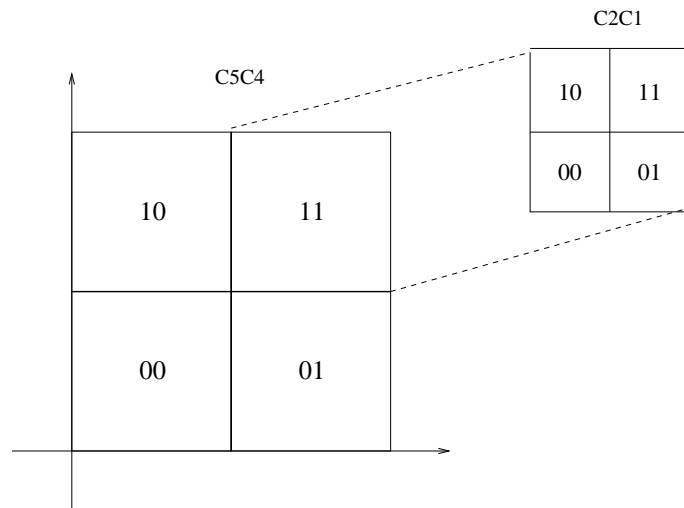


Figure 3.15: Mapping of uncoded bits in the first quadrant of 64-QAM.

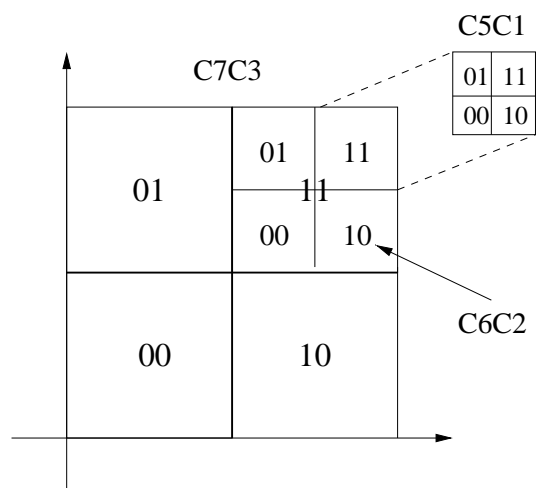
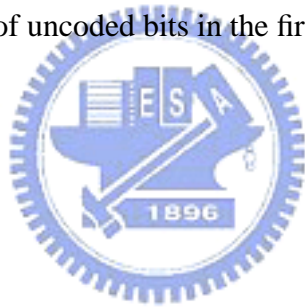


Figure 3.16: Mapping of uncoded bits in the first quadrant of 256-QAM.

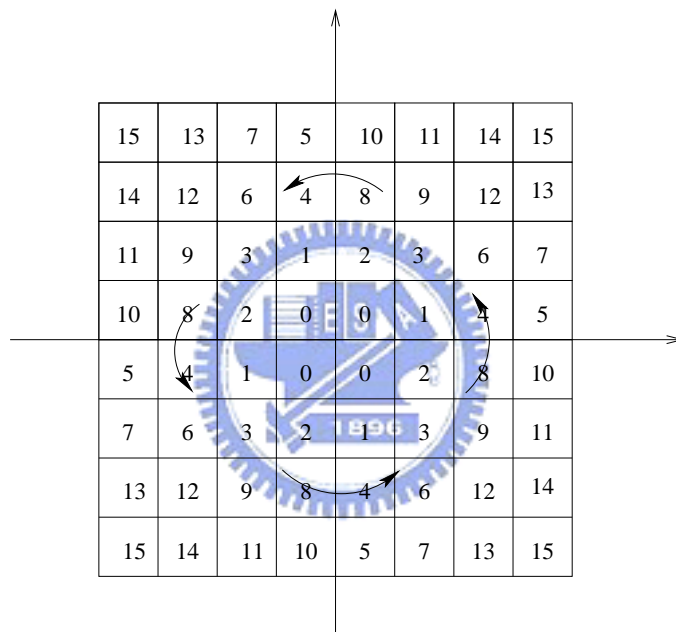


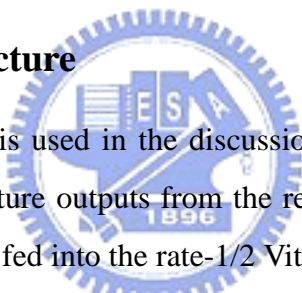
Figure 3.17: Mapping of uncoded bits in other quadrants by rotating the mapping in the first quadrant.

Chapter 4

Algorithm of FEC Decoder

4.1 Decoding of Trellis-Coded Modulation

4.1.1 Decoding Architecture



The TCM decoder in Fig. 4.1 is used in the discussion below. The demodulator produces soft in-phase and quadrature outputs from the received signals through the cable network. Before the signals are fed into the rate-1/2 Viterbi decoder, each branch will be depunctured with depuncturing matrix: [0001:1111] (“0” means no transmission, and “1” denotes transmission). Those decoded bits, the LSBs of the two branches, are encoded and punctured again to map the estimated received symbol through a mapping table. The mapping table uses the punctured signals and (suboptimal) four symbols nearest to the received symbol to decide those uncoded bits. The differential decoder is used to decode those LSBs.

4.1.2 Decoding of Differential Precoder

The differential precoder operates on the pairs (W_j, Z_j) and (X_{j-1}, Y_{j-1}) to obtain an estimate on the transmitted pair (X_j, Y_j) . Therefore, the pairs (X_j, Y_j) and (X_{j-1}, Y_{j-1}) can be used to derive (W_j, Z_j) as (4.1) and (4.2).

$$Z_j = X_j + Y_j + X_{j-1} + Y_{j-1}. \quad (4.1)$$

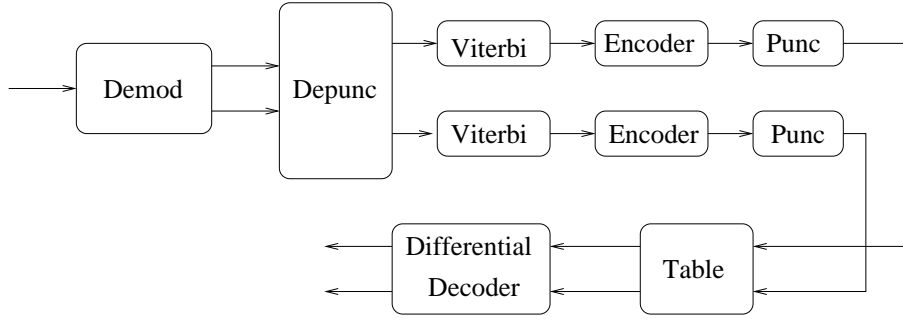
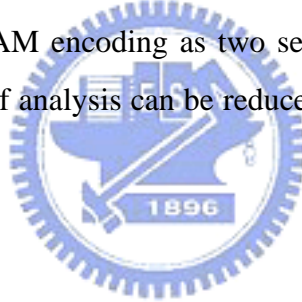


Figure 4.1: TCM decoding architecture.

$$W_j = X_j + Y_{j-1} + (X_j + Y_j)(X + Y_{j-1}). \quad (4.2)$$

4.1.3 Decoding of PBCC

From Figs. 3.4, 3.5, and 3.6, we notice that there are two separate PBCC coding branches. Therefore, we can treat the QAM encoding as two separate PAMs. According to this characteristic, the complexity of analysis can be reduced to two 2^4 -state trellises instead of a 2^8 -state trellis.



Viterbi Decoder

The Viterbi algorithm was proposed in 1967 for decoding convolutional codes. Our application of the Viterbi algorithm to decode TCM is based on the trellis diagram representing the TCM encoder. In the presence of parallel transitions, that is, when more than one signal is associated with the transition from one trellis state to the next, a minor modification should be made in the algorithm. In this situation, the branch metric computation is based on the preliminary detection of the symbol in the branch subconstellation that lies closest to the received signal. Its metric can be used thereafter for that branch, and the Viterbi algorithm can proceed conventionally.

In this situation we have a one-to-one correspondence between the sequence of source symbols and the path through the trellis. Thanks to the set partitioning, the nearest Euclidean distance of the same class is twice the minimum distance of those without set partitioning and hence the degradation of the performance is little compared with the influence of the complexity of convolutional code decoder.

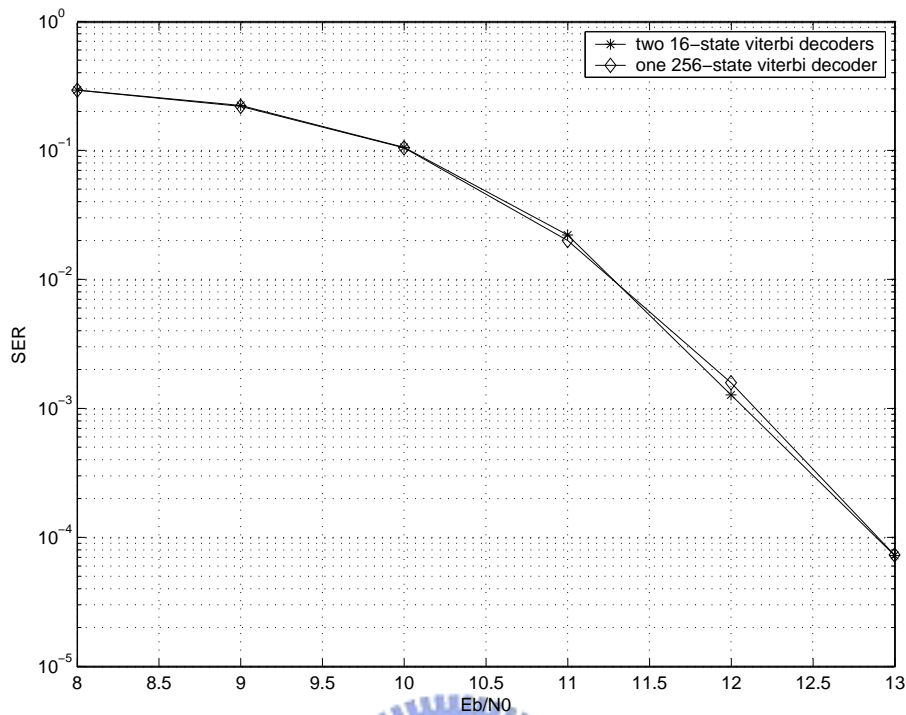


Figure 4.2: Performance of two 16-state decoders and one 256-state viterbi decoder in AWGN.

According to the TCM encoder, the LSBs of the two branches are encoded separately with convolutional encoding, hence we may decode TCM by one Viterbi decoder which decode two branches together. However, decode the two branches takes 256-states Viterbi decoder introducing vast complexity, so we try to decode the two branches with two soft decision Viterbi decoders with 16-states each separately as shown in Fig. 4.1. The simulation results in Fig. 4.2 depict similar performance, which implies the possibility to use two one-dimensional Viterbi decoder with 16 states each instead of one two-dimensional Viterbi decoder with 256 states.

Viterbi Decoder Length

Usually the Viterbi decoder length of 5.8 times the constraint length is enough. However, the punctured convolutional codes may require longer length for its puncture. Therefore, we simulation with different lengths and find a suitable one. Fig. 4.3 shows the error performance of the TCM with 64-QAM using various decoding depths by two one-

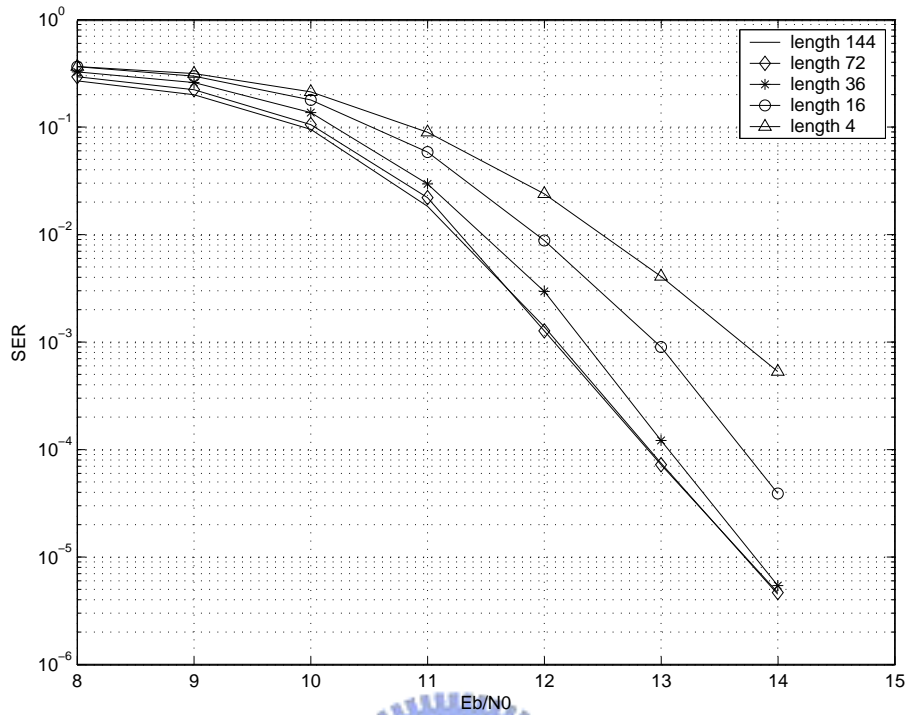


Figure 4.3: Effect on SER of Viterbi decoding depth.

dimensional Viterbi decoders. In the figure, the depth means the numbers of de-punctured symbols pairs with 2 bits in a symbol. For example, since the convolutional encoder is a rate-1/2 coder, there should be 144 symbols in a depth-72 decoder. However, due to the puncturation, there are only 90 received symbols in depth-72 decoder and the rest 54 symbols are inserted with don't cares. Based on the simulation results, a decoder of depth-72 is considered sufficient to achieve good performance.

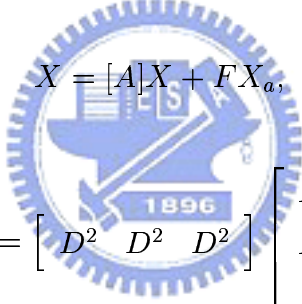
4.1.4 Symbol Error Rate and Bit Error Rate

Since the two LSBs partition the QAM constellation into four subsets, we examine the coding gain of each separately. For memoryless channels an union bound on the bit error probability of a convolutional code can be obtained. The bound could be derived from the transfer function $T(D, B)$ of the code which describes the weight distribution, or wight spectrum, of the incorrect code words and the number of bit errors on the path [11]. Here, D represents the number of "1" source bits and B means the number of "1" coded bits.

Punctured codes are considered high-rate codes and hence their performance can be

derived from the trellis codes of the high-rate code, whose nodes depths are multiple of b for a rate- b/v code [11]. The transfer function of a convolutional code may be evaluated by solving equations describing the transitions between those states. In ITU-T J.83B, the punctured convolutional code is of the rate $4/5$, memory-4 punctured code with generator $G = (25, 37)_{oct}$ and puncture matrix $[P1, P2] = [0\ 0\ 0\ 1, 1\ 1\ 1\ 1]$ (“0” denotes no transmission, “1” denotes transmission). The self-loop at state zero is eliminated by splitting that state into two states X_a and X_b . The rest states are naming X . The exponent of B indicates the number of information bits “1” causing the transition and the exponent of D indicates the Hamming weight associated with the transition. We mention this example for the reason that it has a similiar encoder and most important similiar puncture matrices which can help us understand the trellis expanded by puncturation [11].

The transition behavior could be described by the matrix notation [11]



$$X = [A]X + FX_a,$$

$$[X_b] = \begin{bmatrix} D^2 & D^2 & D^2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix},$$

or

$$X_b = GX.$$

Combine the two equation above, we get a equation

$$T(D, B) = X_b/X_a = G[I - A]^{-1}F.$$

The inverse of matrix $[I - A]$ may be expanded as an infinite series of power of matrices as

$$[I - A]^{-1} = I + [A] + [A^2] + [A^3] + \dots .$$

Substituting into the equation above, we have

$$T(D, B) = GF + G[A]F + G[A^2]F + G[A^3]F + \dots .$$

As a result, the terms a_j and c_j of the weight spectrum from $T(D, B)$ are given by

$$\begin{aligned} T(D, B) \Big|_{B=1} &= \sum_{j=d_{free}}^{\infty} a_j D^j = 2D^3 + 10D^4 + 78D^5 + \dots, \\ \frac{dT(D, B)}{dB} \Big|_{B=1} &= \sum_{j=d_{free}}^{\infty} c_j D^j = 4D^3 + 48D^4 + 528D^5 + \dots, \\ \frac{dT(D, B)}{dD} * D \Big|_{B=1} &= \sum_{j=d_{free}}^{\infty} d_j D^j = 6D^3 + 40D^4 + 390D^5 + \dots. \end{aligned}$$

According to the algorithm above, we obtain the coefficients of a_j , c_j , d_j as shown in Table 4.1.

In the above expressions, d_{free} is the free distance of the code and a_j is the number of incorrect paths or adversaries of Hamming weight j , $j \geq d_{free}$, that split from the correct path and remerge with it sometime later. As for c_j , it is simply the total number of bit errors in the adversaries with Hamming distance j and d_j is the number of bit errors of the source on the branch.

The derivation of the union bound depends on the pairwise error probability P_j associated to the channel condition. Since our simulation channel is an AWGN channel therefore we can write P_j as

$$P_j = Q\left(\frac{\frac{d_j}{2}}{\sqrt{\frac{N_0}{2}}}\right) = Q\left(\sqrt{\frac{j d_{min}^2}{2N_0}}\right) \quad (4.3)$$

where

$$Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz.$$

In eq. (4.3), d_j is the Euclidean distance between the pairwise symbols and d_{min}^2 is the minimum Euclidean distance between two adjacent symbols in the constellation.

Using the weight spectrum in Table 4.1, a union bound of symbol error rate is obtained as

$$P_s \leq \sum_{j=d_{free}}^{\infty} a_j P_j. \quad (4.4)$$

We can derive upper bound of the bit error rate of the punctured convolutional code as

$$P_{b-coded} \leq \frac{1}{b} \sum_{j=d_{free}}^{\infty} c_j P_j \quad (4.5)$$

where b is of the rate- $\frac{b}{v}$ convolutional code.

Similar to (4.5), the bit error rate of the uncoded part is

$$P_{b-uncoded} \leq \frac{1}{v \cdot N} K d_j P_j$$

where N is the number uncoded bits in a symbol.

Combining (4.4) and (4.5), we obtain the bit error rate of one axis as

$$P_b = P_{b-uncoded} + P_{b-coded} \doteq \sum_{j=d_{free}}^{\infty} \left(\frac{1}{v} K d_j P_j + \frac{1}{m} c_j P_j \right). \quad (4.6)$$

In (4.6), K represents the average bit different number of the uncoded bits of two adjacent symbols in the constellation.

Fig. 4.4 illustrates an example of bit errors of an original all-zeros path with an error path diverging from the original path and remerging to it. In this example, there are one 1 and two 0 input bits and therefore cause one bit error in the punctured convolutional code. The number of error symbols, with uncoded bits, is related to the number of 1s in the output of the punctured convolutional code and there are three. If there are average two uncoded-bit errors between two adjacent symbols, the total number of bit errors should be $1 + 2 * 3 = 7$.

The equations above are the union bound for binary convolutional codes. However, there are still parallel transitions in TCM. Therefore, both the parallel transitions of correct path and that of the incorrect path are considered. However, the later is not the dominative term in symbol error rate, so we only analyze approximation of the former as:

$$P_{s-par} \leq 2Q \left(\frac{2d_{min}}{\sqrt{\frac{N_0}{2}}} \right). \quad (4.7)$$

Sum up the above equation and (4.4) we obtain the total symbol error rate as:

$$P_s \leq \sum_{j=d_{free}}^{\infty} a_j P_j + 2Q \left(\frac{2d_{min}}{\sqrt{\frac{N_0}{2}}} \right). \quad (4.8)$$

Table 4.1: Weight Distribution of the PBCC in TCM

d_f	a_j	c_j	d_j
3	2	4	6
4	10	48	40
5	78	528	390
6	528	5123	3168
7	3148	40144	22036
8	19722	3.1102e+5	1.5778e+5
9	1.2227e+5	2.2811e+6	1.1004e+6
10	7.6337e+5	1.6495e+7	7.6339e+6
11	4.7431e+6	1.1634e+8	5.1274e+7
12	2.9417e+7	8.0622e+8	3.53e+8
13	1.8033e+9	5.4327e+9	2.3443e+9
14	1.0833e+9	3.5319e+10	1.5166e+10
15	6.3041e+9	1.4e+11	9.4561e+10

And with (4.6), similar result can be also derived.

Figs. 4.5, and 4.6 depict the simulation results and our analytic results of symbol error rate and bit error rate in AWGN separately. In each plot, E_b is the bit energy in the information source. We have to notice that since there are two independent $\frac{4}{5}$ PBCC in each the TCM scheme. Therefore, the total coding rate is $\frac{28}{30}$ in 64-QAM mode, $\frac{38}{40}$ in 256-QAM mode, and $\frac{48}{50}$ in 1024-QAM mode. In Fig. 4.5, there is a 5 dB gain between classical QAM modulation and TCM. And our simulation shows that in 64-QAM mode, the analytic results are 0.5 dB more than simulation results. In 256-QAM mode and 1024-QAM mode, the simulation results are 0.5 dB more than the analytic results. Similar in Fig. 4.6, we have 4 dB gains between classical QAM modulations and TCM schemes. The analytic results in 64-QAM mode are 0.5 dB more than the simulation results. And in 256-QAM and 1024-QAM modes, the simulation results are 0.25 dB more than the analytic results. In these observation, we also find that the analytic values are very close to the simulation results in high SNR situations.

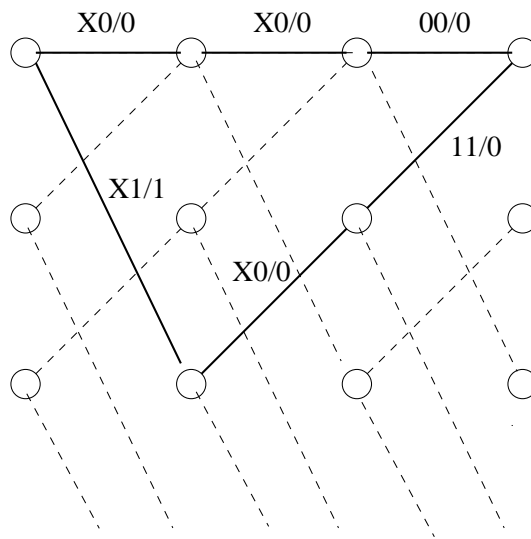


Figure 4.4: State transition diagram of example the PBCC.

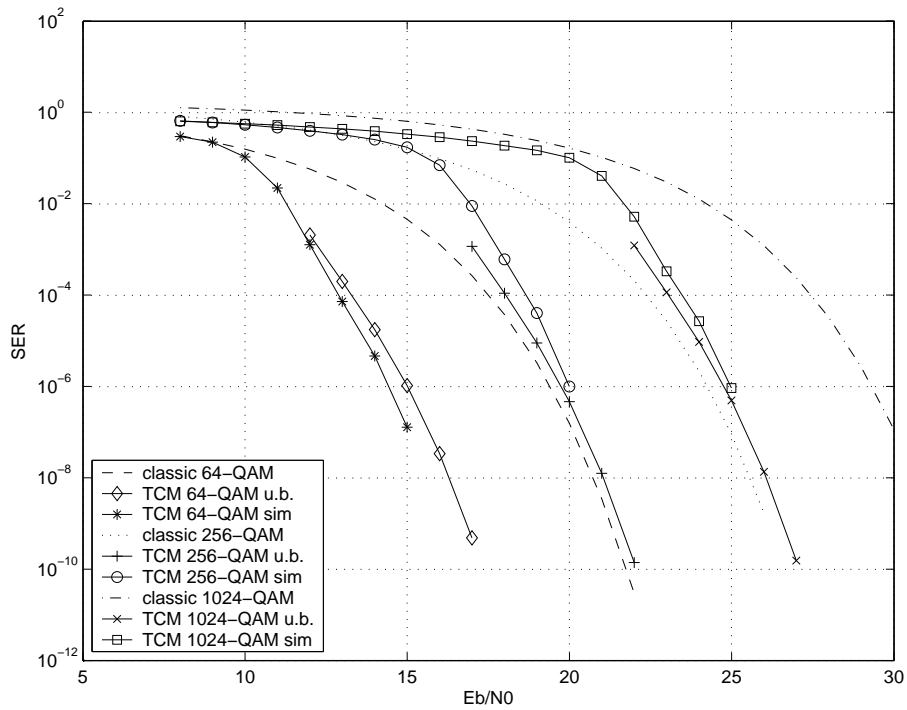


Figure 4.5: SER performance of TCM in 64-QAM, 256-QAM, and 1024-QAM modes.

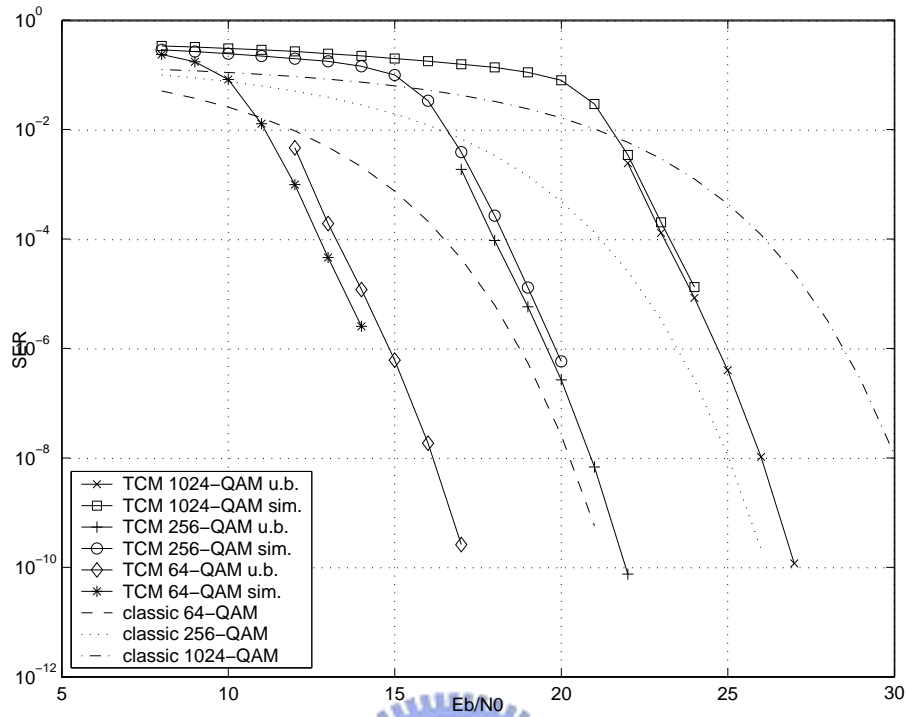


Figure 4.6: BER performance of TCM in 64-QAM, 256-QAM, and 1024-QAM modes.

4.2 De-randomizer

Because the function of the randomizer is to add PN sequence to the FEC frame, the de-randomizer adds an identical PN sequence to recover the original FEC frame.

To generate an identical PN sequence, we use Fig. 3.3 as the de-randomizer. Differentially, “Data In” represents the received symbol and “Data Out” indicates the output symbol to the de-interleaver. The operation is similar to the randomizer. Firstly, the initialization is defined as pre-loading to the “all-ones” state. Secondly, the de-randomizer begins when the first information symbol arrives.

4.3 De-interleaver

As shown in Fig. 3.2, the de-interleaver is the reverse of the interleaver. Besides, the de-interleaver commutator first points to the top-most branch with the longest delay.

This convolutional interleaver and de-interleaver pair are better than conventional block interleaver in two ways: 1) half of the memory units create the same interleaving

distance, and 2) the received information can be outputted continuously after the first valid symbol instead of waiting for the complete arrival of the whole block information.

4.4 Reed-Solomon decoder

Both 64-QAM and 256-QAM mode utilize a (128,122) code over GF(128). The decoding capability is up to $t = 3$ symbol errors per RS block. This code is extended from a (127,122) code by adding a parity check symbol as shown in (3.3).

4.4.1 Extended RS codes

Any narrow-sense $(q - 1, k)$ q -ary Reed-Solomon code C can be extended to form a noncyclic $(q - 1, k)$ q -ary maximum-distance separable (MDS) code by adding a parity check [12, p. 191]. From (3.1) and (3.2), we can further derive a parity-check matrices without and with the single parity check symbol as in (4.9) and (4.10) respectively:

$$H_{(q-1,k)} = \begin{bmatrix} (\alpha^1)^0 & (\alpha^1)^1 & (\alpha^1)^2 & \dots & \dots & (\alpha^1)^{q-2} \\ (\alpha^2)^0 & (\alpha^2)^1 & (\alpha^2)^2 & \dots & \dots & (\alpha^2)^{q-2} \\ \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & & \ddots & & \vdots \\ (\alpha^5)^0 & (\alpha^5)^1 & (\alpha^5)^2 & \dots & \dots & (\alpha^5)^{q-2} \end{bmatrix} \quad (4.9)$$

$$H_{(q,k)} = \begin{bmatrix} 0 & (\alpha^1)^0 & (\alpha^1)^1 & (\alpha^1)^2 & \dots & \dots & (\alpha^1)^{q-2} \\ 0 & (\alpha^2)^0 & (\alpha^2)^1 & (\alpha^2)^2 & \dots & \dots & (\alpha^2)^{q-2} \\ \vdots & \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & (\alpha^5)^0 & (\alpha^5)^1 & (\alpha^5)^2 & \dots & \dots & (\alpha^5)^{q-2} \\ 1 & (\alpha^6)^0 & (\alpha^6)^1 & (\alpha^6)^2 & \dots & \dots & (\alpha^6)^{q-2} \end{bmatrix} \quad (4.10)$$

From (4.9) to (4.10), the number of syndromes is extended from five to six. In (4.10), the first five rows can be used to compute the syndromes S_i for $i = 1, 2, 3, 4, 5$, of the received RS code as follows [7, p. 258]:

$$S_i = r(\alpha^i) = \sum_{j=1}^{127} r_j \alpha^{ij}, \text{ for } i = 1, 2, 3, 4, 5. \quad (4.11)$$

In the equation, r_j is the received j th symbol in a RS block where r_0 is the received extended parity symbol. Furthermore, the sixth syndrome is obtained from the sixth row of (4.10) as:

$$S_6 = r(\alpha^6) = r_0 + \sum_{j=1}^{127} r_j \alpha^{6j}. \quad (4.12)$$

Performance Comparison

The software program for the following simulation is adapted from the public-domain software at the website [8].

In Fig. 4.7(a), the error performance of a two-error correcting RS decoder is compared with that of a three-error decoder for the outer RS(128,122) code. This decoder receives 128 symbols and decodes two random errors in a RS block using first four syndromes in (4.11). In the simulation, symbol error rate after the three-error decoder is almost 1 dB better with respect to the double-error decoder when added symbol error rate is 10^{-3} . In Berlekamp's RS code decoding algorithm, the complexity is $O(t^2)$, where t is the number of error-correcting capability. Therefore, a double-error decoder is $\frac{4}{9}$ (0.49%) less complex than a triple-error decoder. As a result, when complexity becomes an issue, the proposed decoder gives a trade-off between complexity and coding gain.

Ideally, a (128,122) block code can correct up to three errors in a block, and a symbol error bound can be shown as in Fig. 4.7(a) and below:

$$P_s = \sum_{i=4}^{128} \frac{i}{128} \binom{128}{i} P_{as}^i (1 - P_{as})^{128-i}. \quad (4.13)$$

In this equation, P_s is the symbol error probability after decoding and P_{as} is the added random error probability of the received RS symbol before decoding.

However, the simulation result of the triple-error RS code decoder is worse than that of the ideal (128,122) block code. Because once an error occurs in the parity check symbol, the error location or evaluation computation will fail and undermine the performance. The performance bound of the three-error decoder is depicted in Fig. 4.7(a) and can be derived

as:

$$P_s = (1 - P_{as}) \sum_{i=4}^{127} \frac{i}{127} \binom{127}{i} P_{as}^i (1 - P_{as})^{127-i} + P_{as}^2. \quad (4.14)$$

This bound is obtained if all RS block with an erred parity check symbol is detected and then skip the decoding procedure because the decoding procedure must be wrong.

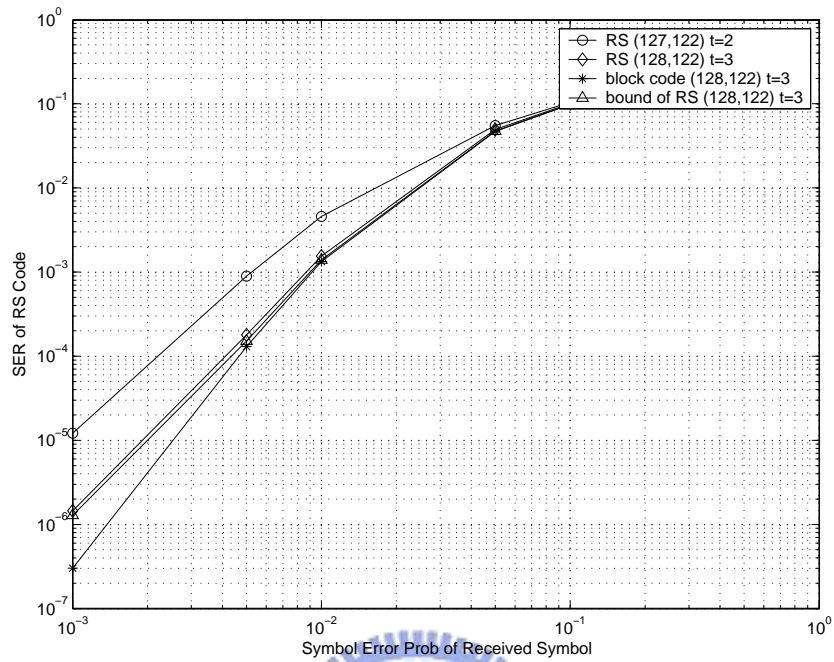
If we force the parity check symbol to be correct, the performance will approach the ideal block code as illustrated in Fig. 4.7(b).

One may wonder whether we can use $t=3$ mode at first and change to $t=2$ mode if the decoder detect the error number is larger than 3. By this idea, the decoder may ignore the parity check symbol error and derives $t=2$ correction ability when errors in other symbols are less or equal to 2. We simulate this idea and find although the parity check symbol error may be avoided, the error detection ability also is reduced. The reduction undermines the performance and therefore do not improve. We conjecture that the decoder might misjudge the error number and causes more error after decoding.

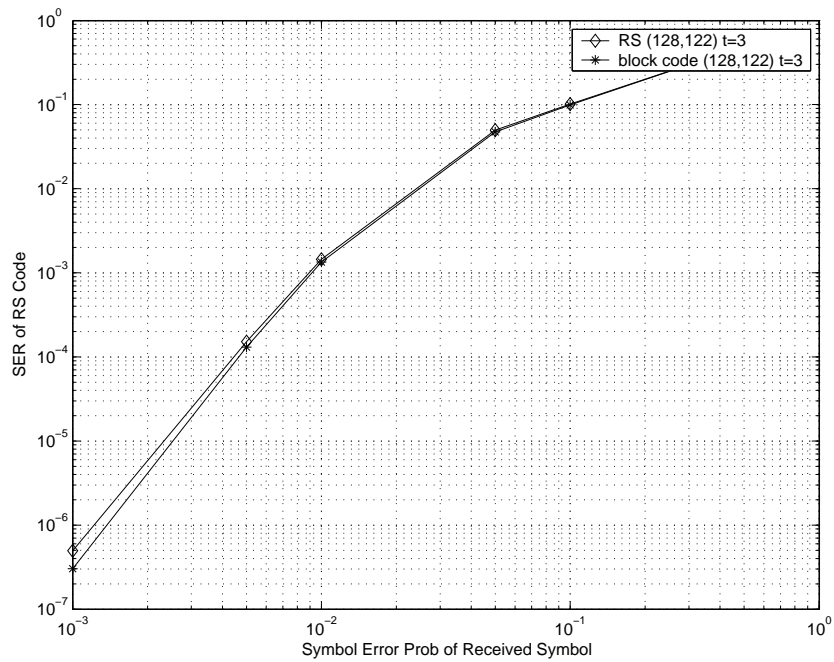
Thesis [24] also elaborates a similar idea about decoding the extended RS code by two possible paths. In one path, the extended parity check symbol is correct and all 6 syndromes are used, while in the other path, the extended parity check symbol is incorrect and only five syndromes are used. This approach first detects the order of the error location polynomial. When the fifth syndrome is used (sixth syndrome is not used), the decoder detects the number. If the order is less than 3 and the error quantity Δ_5 is zero, 5 syndromes are used in decoding. If not, they check at the sixth step (sixth syndrome is used). If the order is 3, 6 syndromes are used in decoding. In that thesis, the author observes that sometimes while 4 symbols are incorrect, one of the error is in extended parity check symbols, the decoder may misjudge that there are only 3 errors. This misjudgement causes more errors after decoding.

Unfortunately, the simulation in that thesis is taken under AWGN while our simulation is under binary symmetric channel (BSC) and hence no comparison could be made directly.

To sum up, the extended RS code can extend the parity check matrix and our decoder provides error correction ability close to $t=3$. However, the complexity is 2.25 times than $t=2$.



(a)



(b)

Figure 4.7: (a) Performance of $t=2$ and $t=3$ for RS code. (b) Performance of $t=3$ with correct parity check symbol and ideal block code.

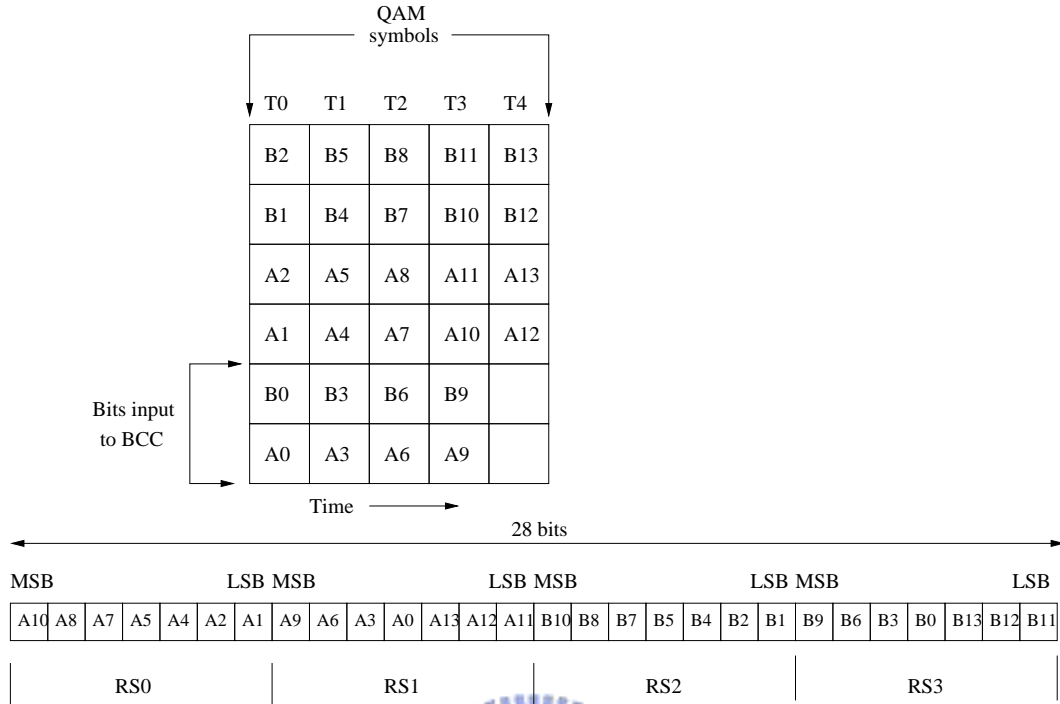


Figure 4.8: 64-QAM trellis group [1, p. 27].

4.5 Concatenated Coding Simulation Results

In this section, we integrate and then simulate the four layers of forward error correction under 64-QAM mode. The trellis group is formed from RS symbols as Fig. 3.4 and Fig. 4.8. Similar to encoding procedure, the decoded bits after TCM is grouped to RS symbols in the same approach. As for the interleaver, parameters $I=128$ and $J=1$ are chosen. Since the 64-QAM mode runs at 5.056941 Mbaud [14] and 4 RS symbols are group into 5 TCM symbols. Therefore, the symbol time of each RS symbol is $0.2472 \mu s$, the burst protection time is $0.2472 \mu s \cdot 128 = 31.6416 \mu s$, and the latency time is $I \cdot (I-1) \cdot J \cdot 0.2472 \mu s = 4.018 \text{ ms}$.

Fig. 4.9 shows the symbol error rate (SER) of the simulation results. The simulation results of 64-QAM TCM from Fig. 4.9 (a) is shown in the '*' line, the simulation results of the overall FEC is in the 'o' line, and the theoretical performance of the overall FEC is in the '+' line. The theoretical performance is derived by the combination of (4.14) and the simulation results of 64-QAM TCM.

The combination is constructed under the assumption that the errors of the RS decoder input are white. This assumption is reasonable because the Euclidean distance of the first

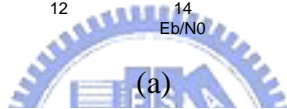
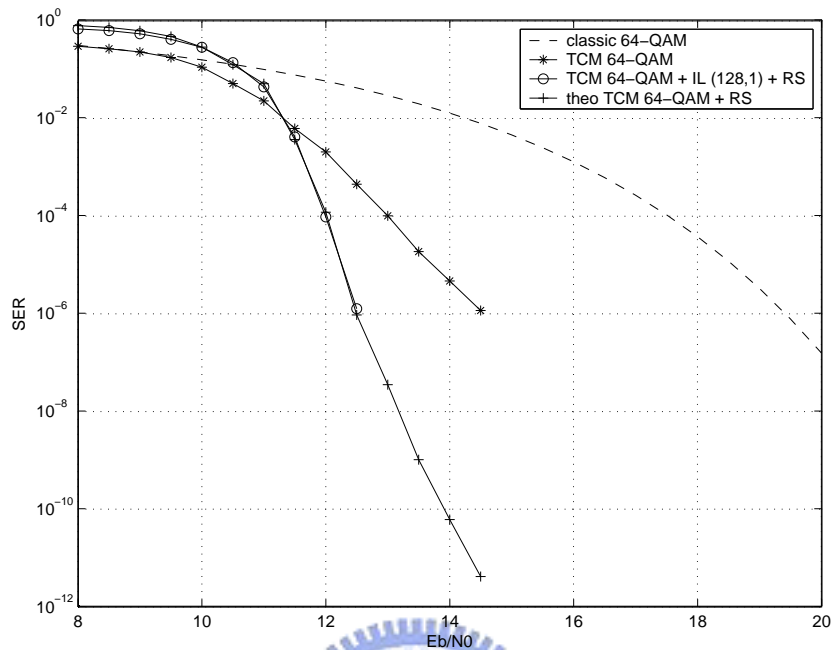
several nearest pairwise error symbol of TCM is not deeper than the depth of the (I=128, J=1) interleaver. The transformation from TCM results to SER of RS code decoder is related to Fig. 4.8. As in the figure, a TCM symbol error can cause the uncoded-bits part error. And on average a single TCM symbol error results 1.2 RS symbols errors. For example, if a single TCM symbol error causes decoding errors of A12 and A13, then RS1 is incorrect. But if the single TCM symbol error causes the incorrection of A10 and A11, both R0 and R1 are erroneous. From Fig. 3.9, however, we observe that the average number of bit difference between two adjacent symbols is 1. Hence, in the last example, only R0 or R1 is erroneous. As a result, on the average, only one symbol will be effected by the uncoded part if a single TCM symbol error happens. Although the error of uncoded-part is related to coded-part, we can assume the interleaver causes the independence of them. To sum up, the SER relationship is as follows:

$$P_{as} = P_{TCM\ SER} + P_{TCM\ coded-part\ BER}.$$

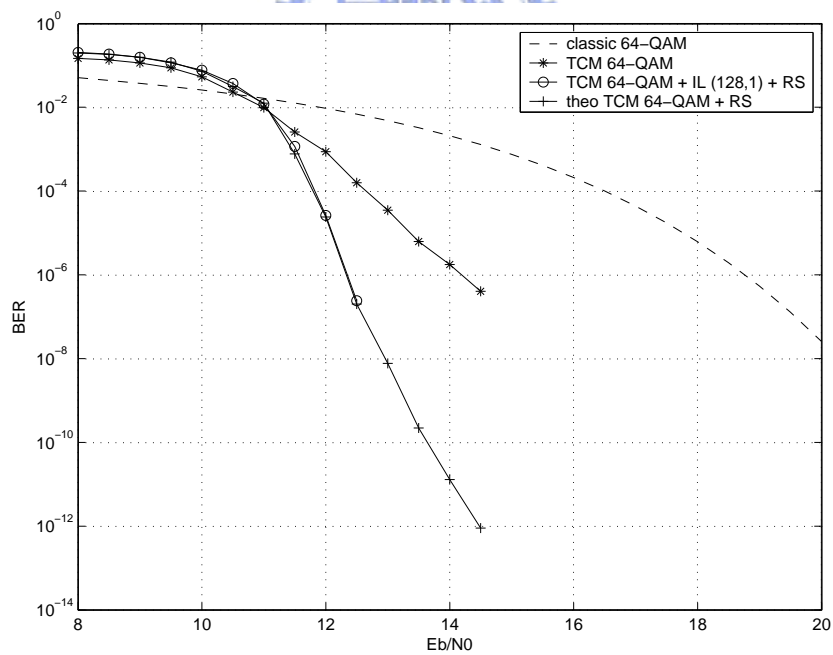
P_{as} is the SER of the input of the RS decoder, $P_{TCM\ SER}$ is the SER of TCM and $P_{TCM\ coded-part\ BER}$ is the BER of the coded part.

The BER of the overall simulation is shown in Fig. 4.9 (b). The analysis of RS decoder is similar to above. However, the direct transformation from SER to BER is complex. Therefore we derive the theoretical performance of BER by dividing the SER theoretical performance by 3.6. Here 3.6 is an experience parameter.

From our simulation results in Fig. 4.9 (b), the coding gain of overall FEC at BER= 10^{-4} is 4.7 dB. In thesis [24], the author derived simulation coding gain by 4.3 dB. The difference may be caused by the different truncation length of the Viterbi decoder, RS decoder, and interleaver. The author used truncation length of 32 and his proposed RS decoder, and interleaver of I=16 and J=8.



(a)



(b)

Figure 4.9: (a) SER of 64-QAM FEC overall simulation results. (b) BER of 64-QAM FEC overall simulation results.

Chapter 5

Turbo Equalization

The increasing demand for high spectral efficiency modulation introduces intersymbol interference (ISI) which deteriorates the received signal. From discussion in chapter 2, we realize that the cable modem receiver suffers from serious intersymbol interference (ISI). As a result, to deal with this phenomenon is necessary for the optimal receiver to accomplish acceptable performance.

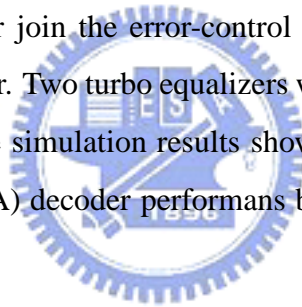
For many years, equalization has been considered as a powerful method in ISI suppression. Optimal equalization methods for minimizing the bit error rate and the sequence error rate are nonlinear and are based on maximum-likelihood (ML) estimation, which turns into maximum *a posteriori* probability (MAP) estimation. The computational complexity of the trellis-based approaches is determined by the number of trellis states, equal to 2^L , that grows exponentially with the number L . This problem is exacerbated in high-order trellis states. For complexity reason, typical equalizer consists of linear processing of the received signal (linear equalizer) and possibly past symbol estimates (decision feedback equalizer). The parameters of these filter can be selected according to various criteria, such as zero forcing (ZF) or minimum mean squared error (MMSE) criteria.

Significant improvements in BER performance are possible with coded data transmission using and error correction code (ECC). Communicating soft information between the equalizer and the decoder, instead of hard information (symbol estimates only), improves the BER performance but usually requires more complex decoding algorithms. A variety of systems employ convolutional codes and ML equalizers together with an interleaver

after the encoder and a deinterleaver before the decoder [19]. Interleaving shuffles the encoded symbols and thus decorrelates error events introduced by the equalizer. These bursty errors are hard to deal with using a convolutional decoder.

Conventional solutions generally involve both equalization and channel coding separately. However, jointly consider the equalization and the decoding processing is usually impossible due to complexity reason. A number of iterative receiver algorithms repeat the equalization and decoding on the same received symbols. The feedback information from the decoder is used by the equalizer and the information from equalizer is also utilized by the decoder. This method is called turbo equalization and is originally developed for turbo coding. Now, this method is adapted into various application such as trellis-coded modulation (TCM), code division multiple access (CDMA) [20], and enhanced general packet radio system (EGPRS).

In this chapter we consider join the error-control coding scheme in existing cable modem standards with equalizer. Two turbo equalizers with different channel decoder are considered and compared. The simulation results shows that turbo equalizer with soft-output Viterbi algorithm (SOVA) decoder performs better than that with conventional Viterbi decoder.



5.1 Principle of the Transmission Scheme

The transmission scheme portrayed in Fig. 5.1. As discussed in chapter 3, the channel encoder and a M-ary mapper consist of trellis-coded modulation. The channel encoders formed by two rate- $\frac{4}{5}$ punctured convolutional code encoder and a differential pre-coder. The cable channel is made by the cable scheme in chapter 2, and w is additive white Gaussian Noise (AWGN).

In Fig. 5.1, the TCM is fed by independent binary data x . The independence of input data is caused by the previous layer, a randomizer. Those data are group in M-ary symbols $d = (a + jb)$ where a and b take equiprobable values in the set $\pm 1, \pm 3, \dots, \pm(\sqrt{M-1})$ with $\sqrt{M} = 2^m$.

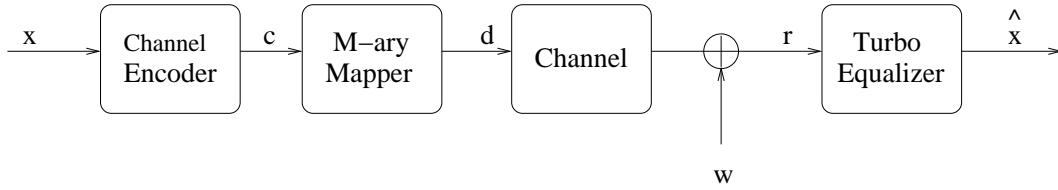


Figure 5.1: Principle of the transmission scheme.

5.2 Turbo-Equalizer with Viterbi Decoder

In this section, we discuss a turbo equalization approach to recover coded data transmission over the channel. The main idea of turbo equalization is to iteratively equalize and decode the received data. In addition, the equalizer and the decoder may provide their information to each other to improve the performance.

A natural starting point for the development is to achieve better SNR performance of the equalizer. Once we provide higher equalizer output SNR, the decoder receives more reliable data and thus reduce the symbol error rate. From experience the more reliability the data used in feedback path of decision feedback equalizer, the higher SNR the equalizer would achieve. When we accomplish higher SNR, the decoder may receive more reliable information and thus reduce its symbol error rate. These reduced symbol error rate data could be transmitted to next stage equalizer in feedback branch as more reliable data. Than the next stage equalizer can start a new iteration.

Each iteration p , $p = 1, \dots, P$, is carried out by a module fed by both the received symbols r^{p-1} and decoded data d^{p-1} from module $p - 1$. This turbo equalizer scheme is drawn in Fig. 5.2. In this figure, the delay is the latency of each module. As depicted in Fig. 5.3, each module consists of an equalizer, a symbol-to-binary converter (SBC), a soft-input hard-output (SIHO) decoder, a binary to symbol converter (BSC).

Fig. 5.4 illustrates our turbo equalization structure. In this structure, we use DFE as our equalizer structure in each stage. In the first stage, the equalizer starts up with MMA algorithm to achieve certain SNR and then switch to LMS algorithm with decision directed mode. The soft output of the equalizer is transmitted to TCM decoder. The TCM decoder uses conventional soft-input hard-output Viterbi decoding algorithm. In later stages, the equalizers use information from last stage TCM output as training sequence.

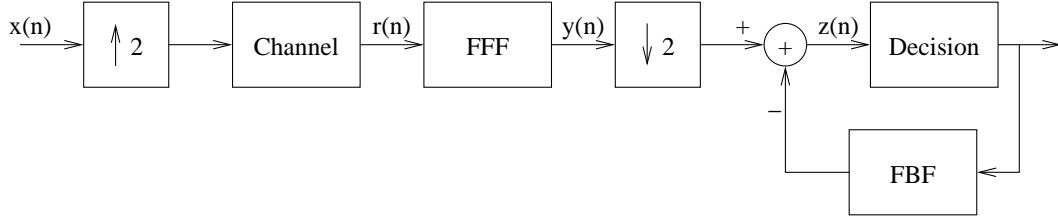


Figure 5.5: DFE structure.

5.2.1 Equalizer Structure

In the cable modem system, the 64-QAM, and 256-QAM (even 1024-QAM) modulation complicates the receiver algorithms due to the multilevel transmission symbols. Since the number of trellis states in the equalizer grows exponentially, the full maximum likelihood sequence estimation (MLSE) cannot be implemented, but suboptimum decision feedback technique like decision feedback equalizer (DFE) is chosen. Consequently, iterative turbo equalization approach is attractive to make up the performance loss.

The decision-feedback equalizer (DFE) is a nonlinear equalizer. Its structure is depicted in Fig. 5.5. A DFE consists of three parts: the feedforward filter (FFF), the feedback filter (FBF), and the decision circuit. The function of FFF is to shorten the time spread of the transmitted signal due to the channel response, and FBF is to subtract out that portion of intersymbol interference produced by previously detected symbols [13, p. 30]. A DFE yields good performance in the presence of severe intersymbol interference.

To illustrate Fig. 5.5, we define the following notations [14]: L_h : length of channel impulse response, in number of symbols, L_f : length of FFF, in number of symbols, L_b : length of FBF, L : oversampling factor at receiver input, equal to 2 in our case, $x(n)$: source symbol at time n , symbol spaced,

To operate under minimum mean-squared error (MMSE) criterion, we can obtain the optimal DFE coefficients and output SNR under the minimum mean-square error (MMSE) criterion [14] as

$$F = [H_1^H H_1 + H_3^H H_3 + \frac{S_n}{S_x} I]^{-1} R_D^H \triangleq C R_D^H \quad (5.1)$$

and

$$B = H_2 F, \quad (5.2)$$

where S_x is the signal power, S_n is the noise power, I is the identity matrix, R_D is the D th row in the Toeplitz-structured channel matrix H (of dimension $(L_f + L_h) \times (L_f L)$), and H_1 and H_3 are submatrices of H composed of the first $D + 1$ rows and the last $L_f + L_h - D - 1 - L_b$ rows, respectively, with L_h being the length of the channel impulse response, L the oversampling factor at receiver input, L_f the length of the FFF in number of symbols, and L_b the length of the FBF in number of symbols. The resulting MMSE is given by [14]

$$J_{min} = S_x(1 - R_D C R_D^H) \quad (5.3)$$

and the associated output SNR given by [14]

$$SNR_D = \frac{S_x}{J_{min}} = \frac{1}{1 - R_D C R_D^H}. \quad (5.4)$$

5.2.2 Adaptatiive Algorithm

Next, we consider which blind algorithm should be choosen. (We will discuss these algorithms and their properties later.) According to the purpose of our study, we choose MMA algorithm instead of CMA because MMA has automatic phase recovery (APR) property which CMA does not have. Therefore the MMA algorithm automatically rotates the constellation to a phase $\frac{\pi}{2}k$, k is an integer, to the transmitted phase. This property just fits the rotationally invariant character of the TCM discussed in last chapter. Recall that the rotationally invariant TCM can decode with phase rotated by $\frac{\pi}{2}k$, k is an integer. In this point of view, we choose MMA as the adaptive algorithm in the first stage.

LMS Adaptation

Practically, a transmission system usually uses an adaptive algorithm to find the proper coefficients. A training-sequence-based adaptation is introduced in the study as Fig. 5.6.

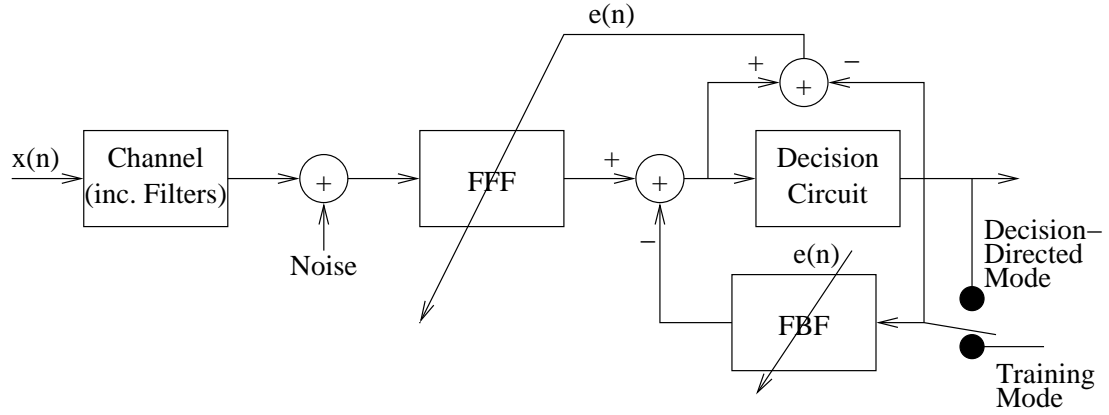


Figure 5.6: Adaptive DFE structure.

Since the main limitation of method of steepest decent is it needs exact computation of a (deterministic) gradient vector $\nabla \Psi(n)$ where $\Psi(n)$ is the cost function. We employ instantaneous estimates to compute the common stochastic gradient vector (SGA) (i.e., the LMS algorithm) for coefficient adaptation. The filter updating method can be described as

$$\begin{aligned} \underline{w}(k+1) &= \underline{w}(k) - \nabla_{\underline{w}(k)} \{[y(k) - d(k)]^2\} \\ &= \underline{w}(k) - \mu e(k) \underline{x}^*(k) \end{aligned} \quad (5.5)$$

where $y(k)$ is the output of the filter, $d(k)$ is the desired signal, $e(k) = y(k) - d(k)$ is the error, $\underline{x}(k)$ is a vector of received signal, and μ is the adaptation step size. This formular can also be described as

$$\begin{aligned} \underline{w}(k+1) &= \underline{w}(k) - \nabla_{\underline{w}(k)} \Psi(y) \\ &= \underline{w}(k) - \mu \frac{\partial}{\partial y(k)} [\Psi] \nabla_{\underline{w}(k)} y(k) \\ &= \underline{w}(k) - \mu \psi(k) \underline{x}^*(k) \end{aligned} \quad (5.6)$$

where Ψ is the cost function, ψ is the differentiation of Ψ with respect to $y(k)$ and is called the error function with respect to Ψ . Therefore, LMS algorithm is SGA with cost function $\Psi = |[y(k) - d(k)]^2| = |e(k)|$. With the cost function, the LMS adaptation algorithm for DEF can be written as

$$F_{n+1} = F_n - \mu \nabla_F |e(n)|^2 = F_n + 2\mu_F e(n) \hat{y}^H(n) \quad (5.7)$$

Table 5.1: Conditions of Simulation 0.1

Constellation	64QAM
Channel Reponse	CH0
Equalizer Length	$L_f = 12, L_b = 20$
Startup Procedure	LMS, training sequence
Step Size	2e-4, 1e-4, 5e-5, 2.5e-5
Simulation Condition	Input SNR = 25 dB

$$B_{n+1} = B_n - \mu \nabla_B |e(n)|^2 = B_n - 2\mu_B e(n) y \hat{x}_{L_b}^H(n-1) \quad (5.8)$$

where μ is the adaptation step size.

Simulation Results of LMS algorithm

We now simulate the LMS algorithm in training sequence mode to obtain the relationship between step size, output SNR, as well as convergence rate. The parameter of this simulation is in Table 5.1. Using the results taken from [14], we set $L_f = 20$ and $L_b = 12$.

The first thing that one notices is that after 2 times downsampling at the receiver, the additive noise should be 2 times lower, which is 3 dB gain in power. Consequently, if the equalizer is ideal, the SNR performance should be 28 dB.

Secondly, from Fig. 5.7, the output SNR after convergence is similar when step sizes are 2.5e-5 and 5e-5. And the convergence rate of step size 5e-5 is twice as fast as step size 2.5e-5. However, there is only 0.1 dB difference between their output SNRs. There is a suggestion that if the convergence rate is concerned, step size 5e-5 can be chosen instead of 2.5e-5. Observe from the figure that, when step size is 2e-4, the SNR performance is 2-dB worse than when step size is 1e-4. Hence, in the view of SNR performance, we should avoid using step size larger than 1e-4. And it would be clear that using step size 1e-4 will reduce output SNR about 0.5 dB. From the above observation, step size 5e-5 would be a choice if a trade-off between convergence rate and SNR performance. According to this choice, the difference the ideal equalizer output SNR and the experimental value is 2 dB.

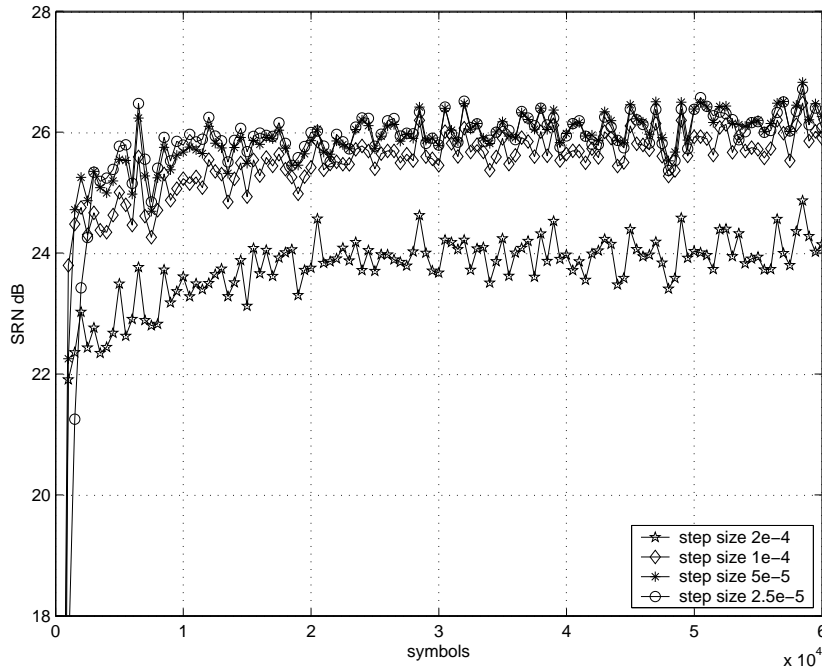


Figure 5.7: Simulation 0.1. LMS with variable step size.

Blind Equalization

According to the broadcasting feature of cable modem, the training sequence is unavailable at all. Therefore, a blind algorithm is used to start and when it converges to a certain SNR, the adaptive algorithm turns to training-based adaptive algorithm to improve the SNR performance.

Unlike conventional trained equalization, blind equalizers do not require a training sequence to start up or restart. Instead, blind equalization use the statistical properties of transmitted sequence as *a priori* knowledge. This blind start-up character enable them to be usefully applied in broadcast and point-to-multipoint networks.

The concept of blind equalization has been known since the publication of Sato's original work on this subject in 1975 [16]. Besides the early contribution, the two best known blind equalization algorithms for two-dimensional modulation schemes are the reduced constellation algorithm (RCA) and the constant modulus algorithm (CMA). Yang *et al.* [16] also proposed the multi-modulus algorithm (MMA).

The Multi-Modulus Algorithm and Its Properties

The blind adaptation algorithms are almost all based on the SGA as (5.6). The design of blind algorithms mostly becomes a design of the cost function, and seeks to minimize a cost function, which is not necessarily the mean square error.

The MMA is also a kind of SGA and its cost function is given by

$$\Psi = E [(y.re^p - R_M^p)^2 + (y.im^p - R_M^p)^2] \quad (5.9)$$

where y is the equalizer output and R_M is called the constraint value of the algorithm. From appendix of [16], y_n would converge to the symbol a_n . Taking the differentiation of the cost function with respect to y , we can obtain the error function of the MMA as

$$\psi = y.re \cdot |y.re|^{p-2} (y.re^p - R_M^p) + j \cdot y.im \cdot |y.im|^{p-2} (y.im^p - R_M^p). \quad (5.10)$$

In practice, a value $p = 2$ usually provides the best compromise between performance and complexity of implementation as [16]

$$\psi = y.re \cdot (y.re^p - R_M^p) + j \cdot y.im \cdot (y.im^p - R_M^p). \quad (5.11)$$

Based on the derivation in [16], when the constellation is symmetric to the real and the imaginary axis, the optimal constrain value is given by

$$R_M^p = \frac{E [|a_n|^{2p}]}{E [|a_n|^p]} = \frac{E [a.re^{2p}]}{E [a.re^p]}, \quad (5.12)$$

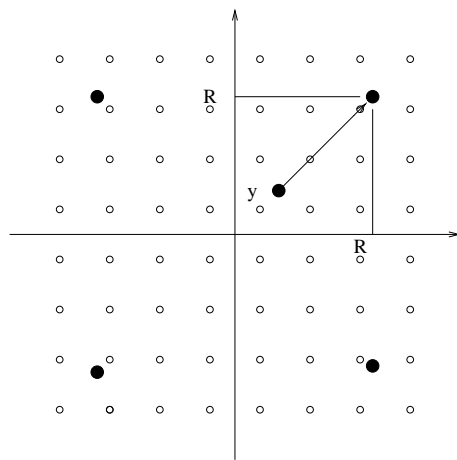
and when $p = 2$, the constraint value is given by

$$R_M^p = \frac{E [a.re^4]}{E [a.re^2]}. \quad (5.13)$$

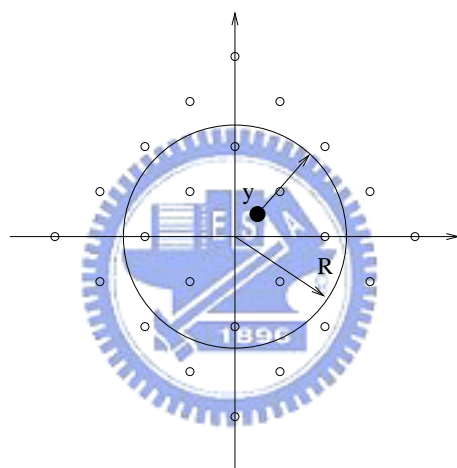
From [16], we can compare the three blind algorithms as shown in Fig. 5.8. From the figure, four points constitute the constraint values of RCA, a circle constitutes that of CMA, and four lines constitute that of MMA. Table 5.2 summerizes the three algorithms.

Automatic Phase Recovery (APR) Property of MMA

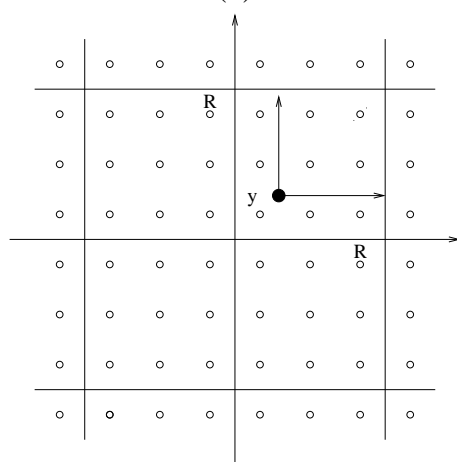
From Table 5.2, the constraint values of MMA form orthogonal sets in two-dimensional space. Therefore, under some non-circularly symmetric constellations, take QAM for instance, the equalized symbols will settle to a phase of an integer multiple of $\frac{\pi}{2}$. Reference [15] derives this property mathematically as follows:



(a)



(b)



(c)

Figure 5.8: Principle of blind algorithms. (a) RCA. (b) CMA. (c) MMA.

Table 5.2: Summary of Blind Algorithms

Algorithm	Cost Function	Error Function $\psi(y)$	Constraint value R
RCA	$E[y - R_R \text{csgn}(y) ^2]$	$y - R_R \text{csgn}(y)$	$R_R = \frac{E[a \cdot re^2]}{E[a \cdot re]}$
CMA	$E[(y ^2 - R_C^2)^2]$	$y \cdot (y ^2 - R_C^2)$	$R_C^2 = \frac{E[a ^4]}{E[a ^2]}$
MMA	$E[(y \cdot re^2 - R_M^2)^2 + (y \cdot im^2 - R_M^2)^2]$	$y \cdot re \cdot (y \cdot re^2 - R_M^2) + j \cdot y \cdot im \cdot (y \cdot im^2 - R_M^2)$	$R_M^2 = \frac{E[a \cdot re^4]}{E[a \cdot re^2]}$

Let x and y be the real and imaginary coordinates of a rotated received QAM constellation point, and let a and b be the real and imaginary coordinates of the original QAM constellation point. Their relationship can be shown as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \quad (5.14)$$

From Table 5.2, the cost function is given by

$$\begin{aligned} \Psi(\theta) &= E \left[(x^2 - R_M^2)^2 + (y^2 - R_M^2)^2 \right] \\ &= E \left[(x^2 - y^2)^2 \right] - 2E \left[(xy)^2 \right] + 2R_M^4 - 2R_M^2 \cdot E \left[x^2 + y^2 \right] \\ &= E \left[(a^2 - b^2)^2 \right] - 2E \left[(xy)^2 \right] + 2R_M^4 - 2R_M^2 \cdot E \left[a^2 + b^2 \right] \\ &= 2E \left[a^4 \right] + 2 \left(E \left[a^2 \right] \right)^2 - 2E \left[(xy)^2 \right] - 2R_M^4 - 2R_M^2 \cdot E \left[a^2 \right]. \end{aligned} \quad (5.15)$$

Since both a and R_M^2 fixed, to minimize the cost function is to maximize $E[(xy)^2]$.

We can include (5.14) to derive

$$E \left[x^2 y^2 \right] = \frac{1}{2} \left(E \left[a^4 \right] - 3 \left(E \left[a^2 \right] \right)^2 \right) \sin^2 2\theta + \frac{1}{4} \left(E \left[a^2 \right] \right)^2 \leq \frac{1}{4} \left(E \left[a^2 \right] \right)^2. \quad (5.16)$$

Due to the uniform distribution of the constellation, $E \left[a^4 \right] - 3 \left(E \left[a^2 \right] \right)^2$ is smaller than zero. Hence the equality of (5.16) holds if and only if $\sin^2 2\theta = 0$. Consequently, the cost function is minimum when when the phase is rotated to an integer multiple of $\frac{\pi}{2}$.

Simulation Results of Blind Algorithms

In this work, we employ blind adaptation as Table 5.2. The structure of the adaptive DFE is shown in Fig. 5.9.

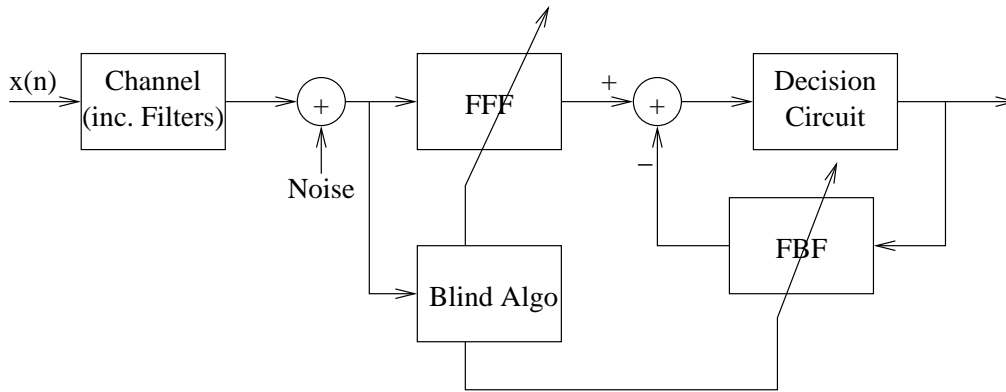


Figure 5.9: MMA adaptive filter using DFE.

MMA and CMA in 64-QAM

Recall that the constraint values of MMA constitute orthogonal sets in two-dimensional space. Therefore, if the algorithm converges, then it would settle at a certain phase. For typical QAM constellations, the settled phase will be an integer multiple of $\frac{\pi}{2}$. However, the constraint value of CMA does not form an orthogonal sets and does not have this property.

We consider simulation 1.1 in Table 5.3. In this simulation, two results are shown. Firstly, we demonstrate the various steps of MMA and CMA start up. Secondly, the automatic phase recovery (APR) property of the MMA is shown while the CMA does not have this character. This simulation is taken under 64-QAM constellation with the same received symbols.

Fig. 5.10 plots 5,000 received symbols before DFE. In this figure, these symbols are scattered before entering the equalizer. Fig. 5.11 depicts the various steps of MMA start up. Fig. 5.11(a) is 5,000 output symbols of DFE with MMA algorithm before the algorithm converges. These symbols are taken after 12,000 iterations in start up. In this figure, the output symbols gather to certain groups but are still quite close. Fig. 5.11(b) shows the output symbols of DFE with MMA algorithm after the algorithm converges. Like in Fig. 5.11(a), 5,000 symbols are taken after 30,000 iterations in start up. Compared with Fig. 5.11(a), symbols in Fig. 5.11(b) are closer to certain groups than they are in Fig. 5.11(a). Therefore, those output symbols are more and more closely gathered in the process of

Table 5.3: Conditions of Simulation 1.1

Constellation	64QAM	
Input SNR	25 dB	
Channel Reponse	CH0	
Equalizer Length	$L_f = 12, L_b = 20$	
Step Size	$1*10e-7$	
Startup Procedure	MMA	CMA

convergence. We can also discover that the constellation rotates to an integer multiple of $\frac{\pi}{2}$, which demonstrates the APR. Using the same received symbols, the same number of output symbols, as well as the same number of iterations after the start up as in Fig. 5.11, we simulate again with CMA and show the results in Fig. 5.12. After comparison of Fig. 5.12(a) and Fig. 5.12(b), the progress of convergence is depicted. But the phase of constellation doesn't rotate to an integer multiple of $\frac{\pi}{2}$ as Fig. 5.11.

To recapitulate, when MMA is convergent, APR will provide a phase rotation of an integer multiple of $\frac{\pi}{2}$. Recall from chapter 4, the TCM in J.83b is $\frac{\pi}{2}k$ rotationally invariant, where K is an integer. Therefore, the phase rotation caused by MMA can be solved by the TCM.

Simulation Results of MMA for 64-QAM

Now we simulate several cases for the suitable step sizes for the tradeoff between convergence rate and converged SNR. These results will be used in further study.

Firstly, we apply the MMA algorithm to 64-QAM as shown in simulation 2.1 in Table 5.4. In this simulation, CH0 in 2.3 is used. Three different step sizes is considered in two noise conditions: with 25 dB input SNR and noise-free. Here, "noise-free" is used to simulate the extremely high SNR situation while "25 dB" is used for normal situation. Furthermore, $4e4$ transmitted symbols are used in the simulation.

The simulation 2.1 results of case 1 is in Fig. 5.13(a). As depicted in the figure, the convergence rate of step size $5e-7$ and $2.5e-7$ are close, and that of step size $1.25e-7$ and $6.25e-8$ is much slower. Sketchily, 5,000 symbols would be needed to converge with step

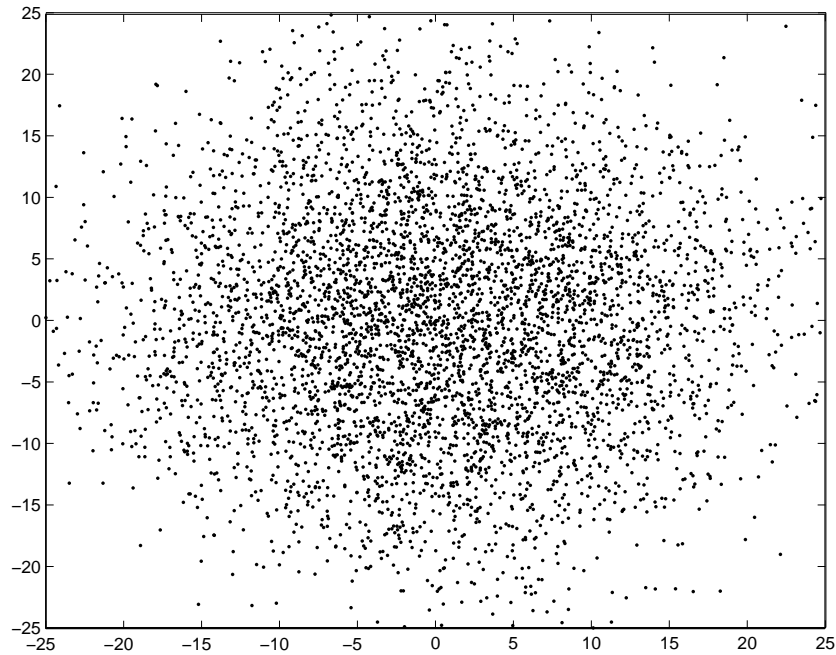


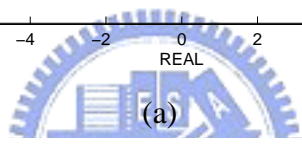
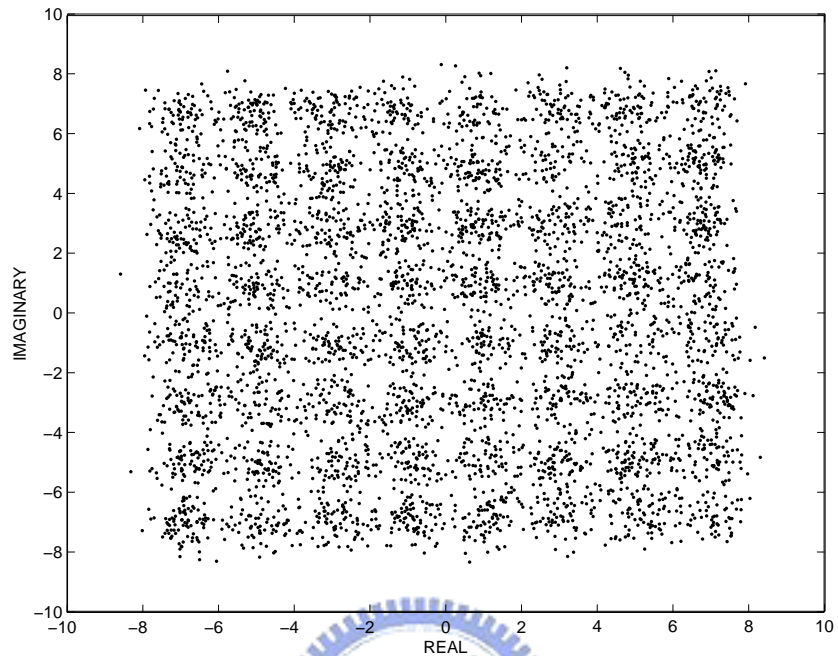
Figure 5.10: Simulation 1.1 Received symbols.

size $5e-7$ and 7,000 symbols for $2.5e-7$. Besides, 15,000 symbols for step size $1.25e-7$ and 30,000 symbols for step size $6.25e-8$. Moreover, the SNRs converge to 17.5 dB with step size $5e-7$, 21 dB with step size $2.5e-7$, and 22.5 dB with step size $1.25e-7$. In the view of trade off between convergent rate and SNR performance, $2.5e-7$ would be a reasonable choice.

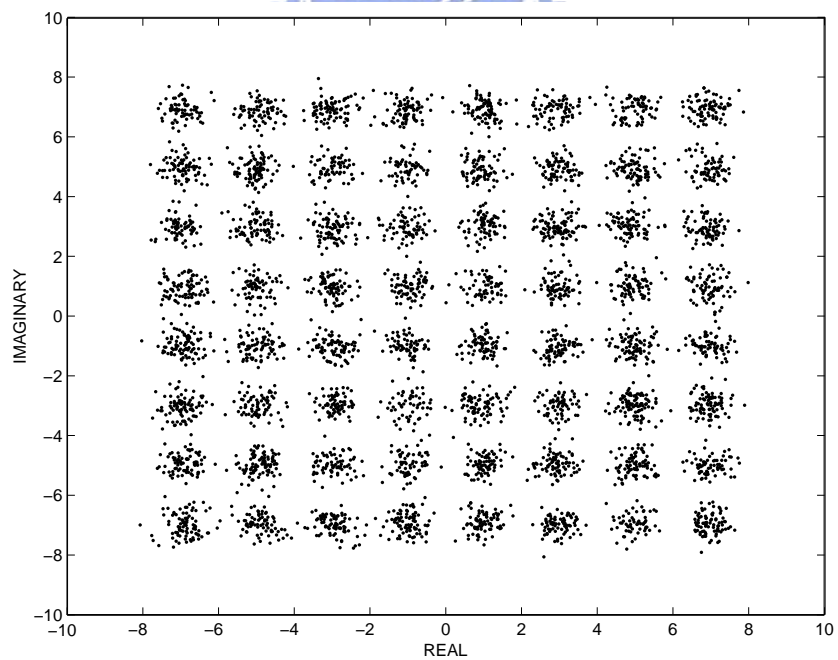
The simulation 2.1 results of case 2 is in Fig. 5.13(b). Similar to case 1, the convergence rate is fastest with step size $1e-6$, then is $5e-7$, and the slowest is $6.25e-8$. Both in case 1 and case 2, doubling the step size causes a 2 to 3 dB loss in SNR. Besides, the smaller the step size, the slower the convergence rate the equalizer achieves.

5.2.3 Channel Decoder

In the decoding part, the reason that TCM is chosen instead of RS code is as follows. We will begin by considering where to feedback the information from the decoder. From chapter 3, the encoder are composed of Reed-Solomon encoder, interleaver, randomizer, and trellis encoder. Relatively, trellis decoder, de-randomizer, de-interleaver, and Reed-Solomon decoder constitute a complete decoder. Now we consider the latency of each

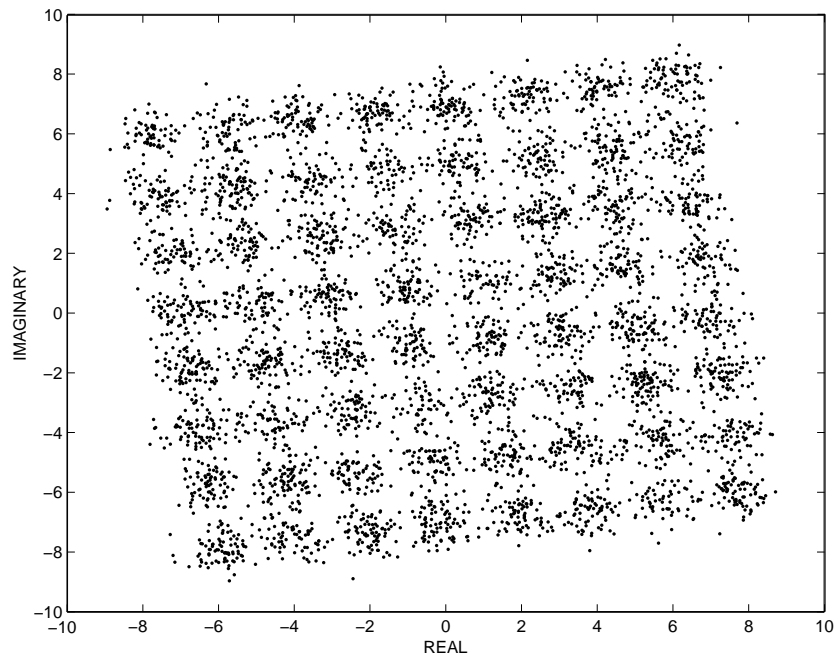
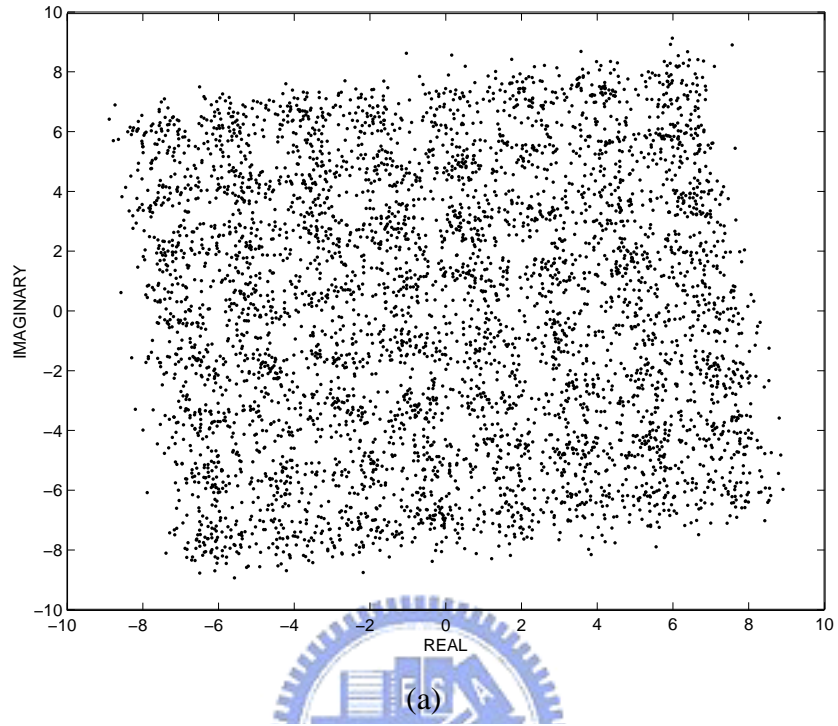


(a)



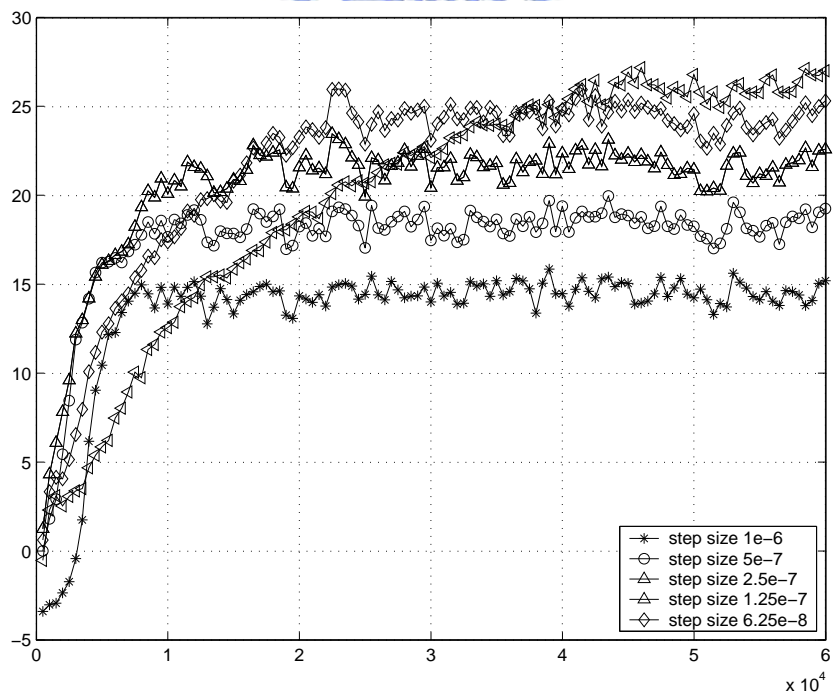
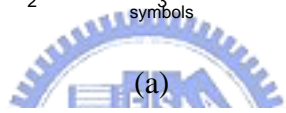
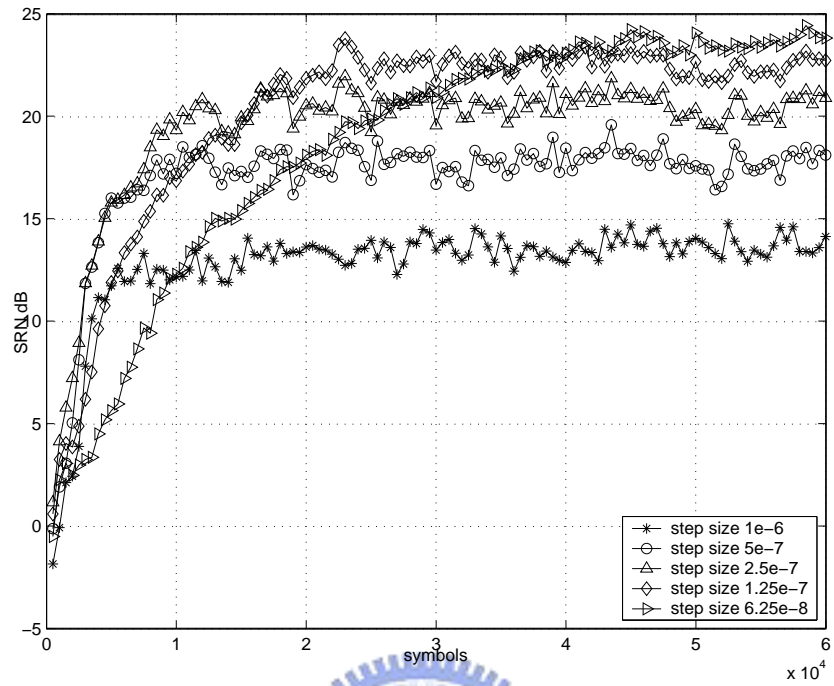
(b)

Figure 5.11: Simulation 1.1 of MMA. (a) Before convergence. (b) After convergence.



(b)

Figure 5.12: Simulation 1.1 of CMA. (a) Before convergence. (b) After convergence.



(b)

Figure 5.13: Simulation 2.1 of MMA. (a) Case 1, input SNR = 25 dB. (b) Case 2, noise-free.

Table 5.4: Conditions of Simulation 2.1

Constellation	64QAM	
Channel Reponse	CH0	
Equalizer Length	$L_f = 12, L_b = 20$	
Startup Procedure	MMA	
Step Size	1e-6, 5e-7, 2.5e-7, 1.25e-7, 6.25e-8	
Simulation Condition	Case 1: Input SNR = 25 dB	Case 2: noise-free

component and do the trade off between the latency and performance. As shown in Fig. 4.3, the Viterbi decoder length is 72 (requiring 90 input symbols). From [1, pp. 22–23], interleaver contains several modes, including $(I,J) = (128,1), (128,2), (128,3), (128,4), (128,5), (128,6), (128,7), (128,8), (64,2), (32,4), (16,8),$ and $(8,16)$. Among these modes, the $(128,8)$ mode will yield the longest delay, of 65016 symbols, which is 722.4 times the latency of Viterbi decoder. According to this observation, the feedback path should come from the output of Viterbi decoder length.

5.2.4 Simulation Results of Turbo Equalizer with Viterbi Decoder

In this simulation, we try to find the performance of our proposed turbo equalizer. Ideally, the symbol error rate will be better through several iterations and finally the ISI will be completely cancelled and transmitted symbols are only corrupted by a correlated noise. Because by iterative decoded data, the DFEs may escape the local minima and reach more accurate coefficients. And the decoders may find different decoding path through those iteratively equalized symbols.

In the first stage, the start up procedure includes two steps. Firstly, the equalizer starts up with MMA blind algorithm. When the equalizer achieves certain SNR, the second step is to switch the equalizer to LMS adaptive algorithm with decision directed mode. Fig. 5.14 depicts the output SNR of the first stage. In the figure, the equalizer starts with MMA algorithm and converges to 18 dB when input SNR is 18 dB. As long as MMA is converged, we switch the adaptive algorithm to LMS at about the 20,000th symbol. LMS algorithm provides 3 dB improvement to 21 dB level. According to the statistical

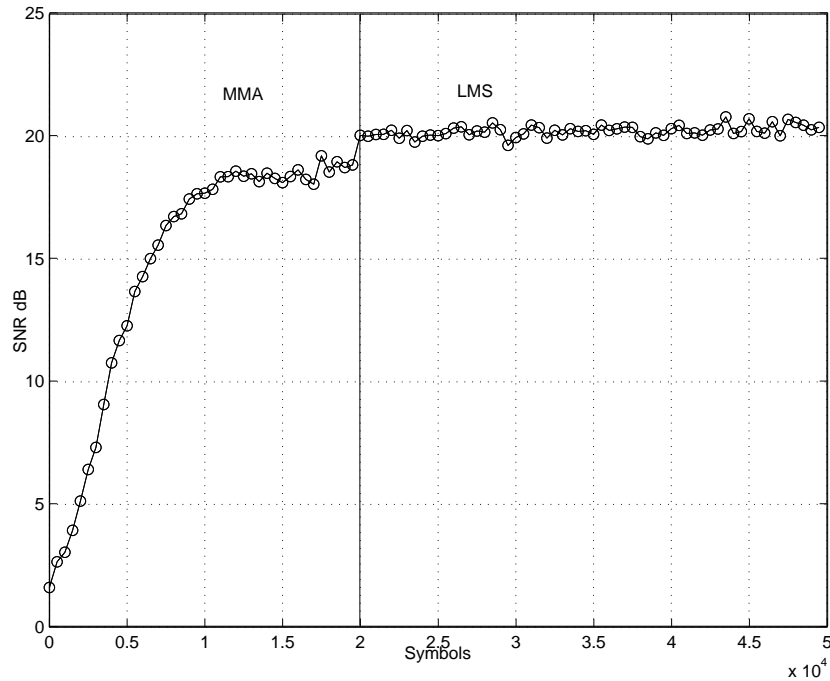


Figure 5.14: SNR rise between MMA and LMS algorithms.

characteristics of cable modem systems, it is reasonable to use LMS after MMA without worrying about channel might dynamically change.

When the equalizer is converged, the output from the equalizer is sent to a TCM decoder. Within the decoder, two conventional Viterbi decoders are used as shown in chapter 4. The decoded bits are then re-encoded and mapped to as estimated data and transmitted to the feedback branch of the next stage equalizer. The next stage equalizer uses LMS adaptive algorithm with the feedback information from last stage. Then again, the equalizer output is sent to next stage TCM decoder and starts the next stage.

The parameter of this simulation is the following. As before, 64-QAM modulation is chosen, CH0 is used. Then in start-up procedure, the blind equalization uses MMA algorithm and then uses LMS algorithm as in Fig. 5.4. From Fig. 5.4(a), we choose step size $2.5e-7$ for MMA algorithm, and from Fig. 5.7, we choose step size $2e-5$ for LMS algorithm. The equalizer length $L_f = 12$, and $L_b = 20$ are chosen through [14] and [15]. In the simulation, it is important to transmit the coefficients from the current stage equalizer to the next stage. Without this transmission, the next stage equalizer has to start

Table 5.5: Conditions of Simulation 3.1—Turbo Equalizer

Constellation	64QAM
Channel Reponse	CH0
Equalizer Length	$L_f = 12, L_b = 20$
Startup Procedure	MMA and LMS
Iterative Algo.	LMS
Decoder	Viterbi Decoder
Step Size of MMA	$2.5e-7$
Step Size of LMS	$2e-5$ (1st stage) $2e-6$ (later stages)

up from the initial value and the convergence time will be lengthened.

From Fig. 5.15, we can obtain about 3.2 dB in SER about 10^{-4} when turbo equalization is compared with DFE with MMA algorithm.

In figure, classic 64-QAM is also employed to get the total coding gain as 3.8 dB. Here the SER of TCM in 64-QAM mode from chapter 4 is introduced. This is because a perfect turbo equalizer can eliminate ISI and therefore the SER can approach the SER of TCM under AWGN. So we can compare our equalizer with a perfect equalizer and obtain there is still 1.1 dB loss in our turbo equalizer.

The performance comparison with DFE with LMS (decision directed mode) and turbo equalizer is also shown in the figure. The performance of DFE with LMS corresponds to the first iteration ($p = 1$) of the turbo equalizer. It can be shown that the turbo equalizer obtains 0.2 dB gain in the second stage and totally 0.21 dB in the third stages.

From the above observation, Viterbi algorithm itself may obtain 0.21 dB by two times iterations (the third stage). This is because it makes hard symbol estimates resulting in performance loss in multistage detection.

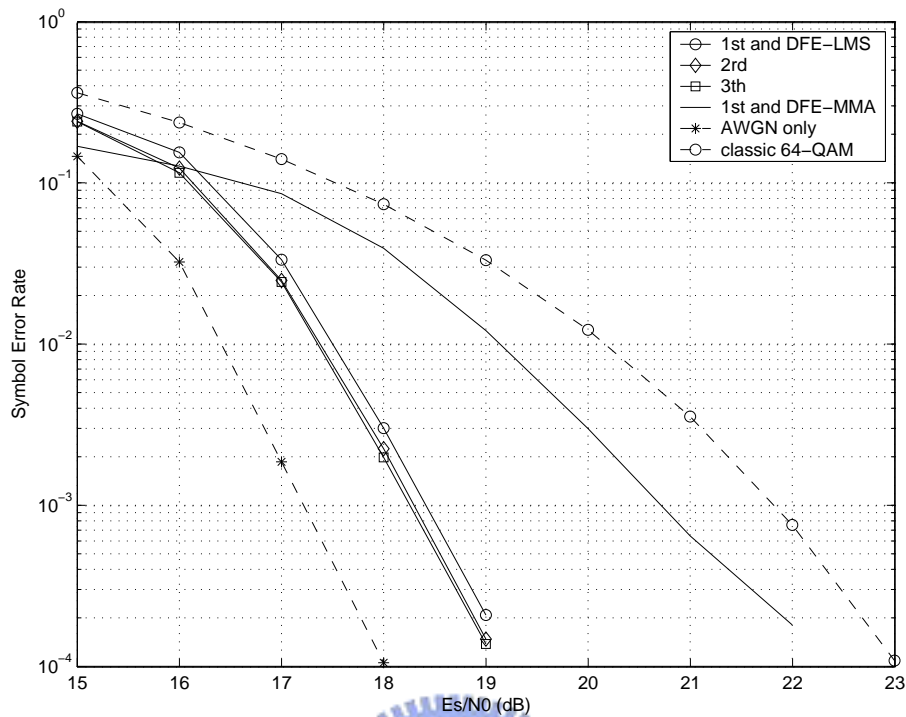


Figure 5.15: Simulation 3.1. Simulation results of turbo equalizer with Viterbi decoder.

5.3 Turbo Equalizer with Soft-input Soft-output Decoder

The hard decision of Viterbi decoder undermines the performance of turbo equalizer. Therefore we use soft-input soft-output M-ary decoder instead of conventional Viterbi decoder. It is organized as in Fig. 5.16.

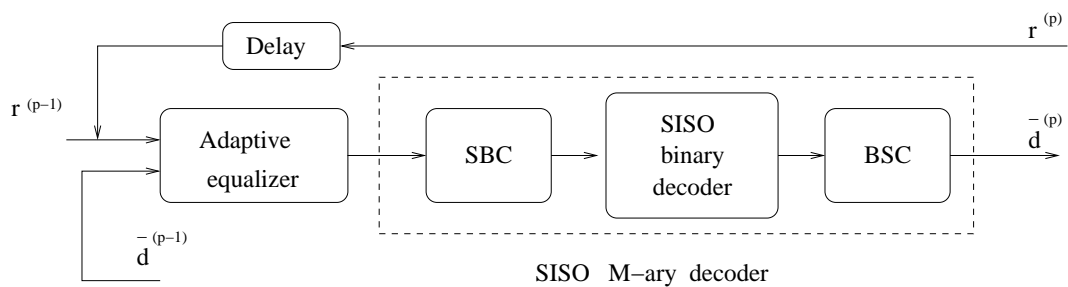


Figure 5.16: Schematic diagram of module $p - 1$.

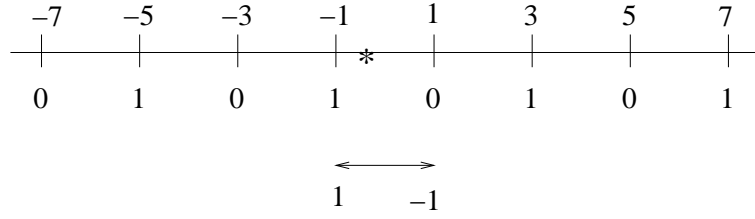


Figure 5.17: Example of symbol to binary converter for 64-QAM.

5.3.1 Symbol to Binary Converter (SBC)

This function enables the same binary channel decoder to be used regardless of the state number of multilevel M -ary modulation. The SBC associates values $\Lambda(c_{k,i})$. Values of $\Lambda(c_{k,i})$ are defined as the LLR or binary coded data $c_{k,i}$; $i = 1, 2, \dots, 2m$, at each sample $s_k = d_k + e_k$. Here m represents the number of bits per axle per symbol contains [21]. Detail mathematical derivation from LLR to received symbol mapping is shown in [21].

Here we use the physical meaning of LLR to obtain our SBC. Also from above discussion, we can decode these two axis independently. As shown in chapter 3, the coded part of TCM is 0 and 1 mutually arranged in both I and Q axis. Therefore, we can use the distance between the received point and a reference point as the reliability. Fig. 5.17 shows an example of a received symbol located between $(-1,1)$ in -0.7 . Since we represents 1 and -1 instead of 1 and 0 in SOVA, the reliability can be linearly transform from to a value between 1 and -1. In this example, the reliability is 0.85.

5.3.2 Soft-input Soft-output (SISO) Channel Decoder

From the above discussion, Viterbi algorithm (VA) itself could not be very successfully implemented in the iterative decoding schemes because it makes the hard symbol estimates resulting in performance loss in multistage detection. A modification of Viterbi algorithm, named soft output Viterbi algorithm, gives soft outputs in the form of log-likelihood function [23]. At the final iteration, SOVA makes the hard decision by comparing the outputs with the zeros thresholds. SOVA is a bi-direction algorithm, which means that it consists of two sets of calculations, forward and backward recursions.

MAP algorithm calculates probabilities of 1s and 0s of each particular bits. Its soft

Table 5.6: Complexity Comparison of MAP and SOVA Algorithms Implemented in Turbo Equalization (from [22])

	MAP	SOVA
Additions	$2 \cdot 2^k \cdot 2^v + 6$	$2 \cdot 2^k \cdot 2^v + 9$
Multiplications	$5 \cdot 2^k \cdot 2^v + 8$	$2^k \cdot 2^v$
Max ops.		$2^k \cdot 2^v - 1$
Exp.	$2 \cdot 2^k \cdot 2^v$	

output is calculated as the log-likelihood ratio. Although MAP algorithm is more accurate, its complexity is higher than SOVA. Table 5.6, from [22], compares the complexity of these two algorithms. Algorithms are represented by the number of computation operations for an (n, k) convolutional code of v memory registers. Clearly, SOVA has the advantages of less multiplications and no exponential operations.

5.3.3 Introduction to SOVA Algorithm

In this section, we describe SOVA algorithm. The soft output of it contains two pieces of information. Firstly, the sign represents the conventional hard decision decoded information. Secondly, the magnitude shows the probability of a correct decision. SOVA delivers soft decisions

$$\hat{\Lambda}_k = \hat{u}_k \hat{L}_k. \quad (5.17)$$

The LLRs are the logarithm of the ratio of two probabilities. For example, as a given decoded bit u_k , its LLR is given by

$$\hat{L}_k = \log \frac{P(u_k = +1)}{P(u_k = -1)} \quad (5.18)$$

where $P(u_k = +1)$ is the probability that $u_k = +1$, and so does $P(u_k = -1)$. Notice that we use +1 and -1 instead of 1 and 0.

To give soft output, we modify classical VA as follows. The algorithm starts with conventional VA to decode \hat{U}_k . With VA we have

$$Prob(path\ m) \sim e^{M_m}, m = 1, 2. \quad (5.19)$$

Here we label the path m and metric of it as M_m . In a binary trellis, there are two paths reaching state $S_k = s$ at stage k in the trellis. As traditional VA, the metrics of these two paths are calculated. If two path \underline{s}_k^s and $\hat{\underline{s}}_k^s$ reaching state $S_k = s$ have metrics $M(\underline{s}_k^s)$ and $M(\hat{\underline{s}}_k^s)$, and the path \underline{s}_k^s is selected as the survivor because its metric is higher, then we define the metric difference as

$$\Delta_k^s = M(\underline{s}_k^s) - M(\hat{\underline{s}}_k^s). \quad (5.20)$$

The probability that the decision is correct is,

$$P(\text{correct decision at } S_k = s) = \frac{P(\underline{s}_k^s)}{P(\underline{s}_k^s) + P(\hat{\underline{s}}_k^s)}. \quad (5.21)$$

Substitute with metrics, we derive

$$\begin{aligned} & P(\text{correct decision at } S_k = s) \\ &= \frac{M(\underline{s}_k^s)}{M(\underline{s}_k^s) + M(\hat{\underline{s}}_k^s)} = \frac{e^{\Delta_k^s}}{1 + e^{\Delta_k^s}}. \end{aligned} \quad (5.22)$$

Through some approximation [23], [17], we derive the LLR as

$$L(\hat{u}_k | y) \approx u_k \min_{\substack{i=k \dots k+\delta \\ \hat{u}_k \neq \hat{u}_k^i}} \Delta_k^{s_i} \quad (5.23)$$

where u_k is the value of the bit given by the ML path and u_k^i is the value of this bit for the path which merged with the ML path and was discarded at trellis stage i . Parameter δ is chosen by the following reasons. Observations of VA have shown that all survival paths at stage l in the trellis normally come from the same path at some point before l . This point is considered δ transitions from l , and δ is usually five times the constraint length of the convolutional codes. Thus the LLR of SOVA must consider the probability that the paths merging with ML path from stage k to stage $k + \delta$ were incorrectly discarded. This value is calculated by considering the metric difference $\Delta_k^{s_i}$ for all stages s_i along the ML path from transition k to stage $k + \delta$. Thus the minimization in Eq. (5.23) is achieved by saving those metric difference of the path merging with the ML path which would have different value from the decoded bit u_k . Those path with the same decoded bit would not affect the reliability of decision of u_k . In other explanation, the reliability, LLR, of the decoded bit is about the probability it would be replaced by different decoded result. In Viterbi decoder, the probability could be transform to the distance difference, or metric difference.

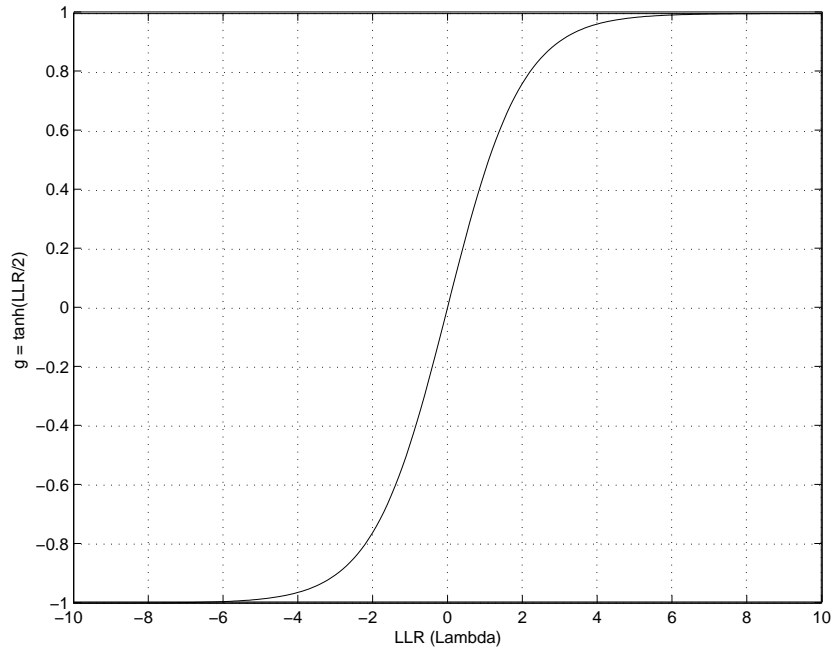


Figure 5.18: Transfer function from LLR to estimated symbol.

5.3.4 Binary to Symbol Converter (BSC)

To feed the equalizer feedback filter. It is necessary for the transmitted symbols to be known or estimated. Provided by previous module decoder, the LLR can guide to a estimated value \bar{d} as shown in Fig. 5.16.

The range of LLR from the decoder varied in vast area, but unfortunately the estimated symbols values are limited. Consequently, we use a mathematical equation to restrain the LLR from +1 to -1 as

$$\bar{g} = \tanh\left(\frac{\hat{\Lambda}(c)}{2}\right). \quad (5.24)$$

And Fig. 5.18 shows the relationship between LLR from last module and the estimated \bar{g} . However, g is just the reliability of decoded bits not the estimation of the transmitted symbol. We still needs to encode the decoded bit \bar{u}_k to get the estimated symbol sign. With its sign and reliability from g we obtain estimated \bar{d} through reference points as shown in Fig. 5.17.

Table 5.7: Conditions of Simulation 4.1.—Turbo Equalizer.

Constellation	64QAM
Channel Reponse	CH0
Equalizer Length	$L_f = 12, L_b = 20$
Startup Procedure	MMA and LMS
Iterative Algo.	LMS
Decoder	SOVA Decoder
Step Size of MMA	$2.5e-7$
Step Size of LMS	$2e-5$ (1st stage) $2e-6$ (later stages)

5.3.5 Simulation Results of Turbo Equalizer with SOVA Decoder

For simulation, the parameters from last simulation are continuously used. The equalizer in the first iteration is the same as last simulation which also represents the performance when only MMA and LMS are used without any turbo iteration. Again, we use DFE in all stages as our equalizer structure. Here we simulation two possible cases.

In the first case, MMA algorithm is used to converge to a certain SNR and then switch to LMS with decision directed mode in the first stage. And LMS alorithm are employed in later stages as Table 5.7. As shown in Fig. 5.19, the turbo equalizer obtain 3.4 dB in high SNR when compared with one DFE with MMA. When we comapre turbo equalizer with DFE with LMS algorithm, 1.2 dB gain is obtained in low SNR situation and 0.43 dB is achieved in high SNR. In the simulation, it is important to shrink the step size after first stage, or we may not have much gain in the second stage. It is shown that in the second stage, the turbo equalizer provides 1.1 dB in low SNR and 0.36 dB in high SNR. And the performance saturates in the fifth stage. The total coding gain between classic 64-QAM and 64-QAM mode TCM with turbo equalizer is 4.2 dB in SER 10^{-4} . The figure also depicts there is also 0.8 dB gain ideal turbo equalizer and our proposed turbo equalizer in SER 10^{-4} .

In the second case, MMA is used in the first stage and the later stages as Table 5.8. This turbo equalizer is useful when the channel changes frequently because its first stage

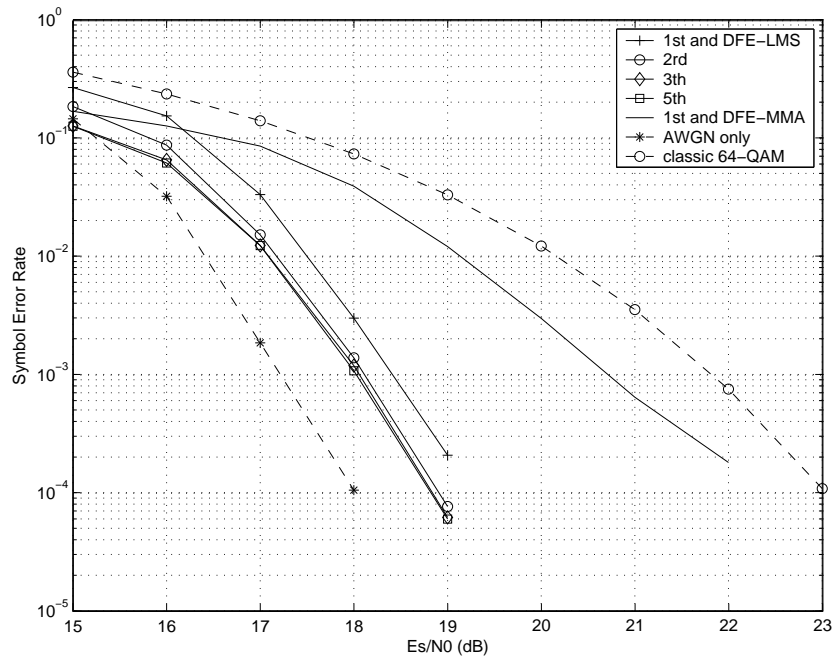


Figure 5.19: Simulation 4.1. Simulation results of turbo equalizer with SOVA decoder.

use blind algorithm only. As depicted in Fig. 5.20, the turbo equalizer obtains 3.5 dB gain between the first stage and the 5th stage in SER 10^{-4} . However, there is a 1 dB difference between this turbo equalizer and the ideal turbo equalizer.

5.4 Turbo Equalizer Conclusion

In this chapter, we have demonstrated two different turbo equalizers structure. Compared with DFE with MMA algorithm only, the one with traditional VA decoder improves 3.2 dB and the other one with SOVA decoder improves 3.4 dB in SER 10^{-4} . When it comes to DFE with LMS algorithm, the first turbo equalization algorithm reaches 0.21 dB gain while the second algorithm acquire 0.43 dB. From the observation, VA produces hard estimated symbol that undermines the performance while SOVA introduces soft output, reliability, which provides more information.

Table 5.8: Conditions of Simulation 4.2.—Turbo Equalizer.

Constellation	64QAM
Channel Reponse	CH0
Equalizer Length	$L_f = 12, L_b = 20$
Startup Procedure	MMA
Iterative Algo.	LMS
Decoder	SOVA Decoder
Step Size of MMA	$2.5e-7$
Step Size of LMS	$2e-6$ (later stage)

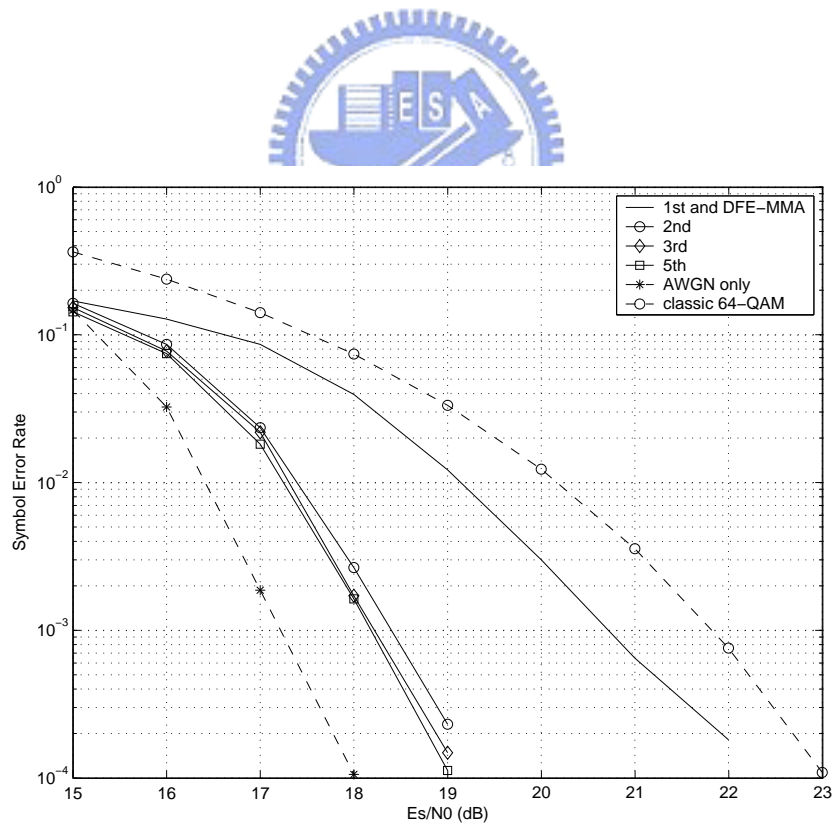


Figure 5.20: Simulation 4.1. Simulation results of turbo equalizer with SOVA decoder.

Chapter 6

Conclusion and Future Work

This work studied two parts of J.83B: one is the forward error correction and the other is the turbo equalizer applied. And we focused on the error correction properties and the impact of turbo equalization in this specification.

The forward error correction in J.83B employs four layers including Reed Solomon coding, interleaving, randomizer, and trellis-coded modulation. In the RS code part, we analyzed the property of the extended RS code and derived its single extended parity-check matrix. Also, we simulated the decoder and the results show that its error correction ability is between $t=2$ and $t=3$. In the trellis-coded modulation part, we presented the rotational invariance characteristic of the trellis-coded modulation. Furthermore, the weight distribution of the punctured convolutional code in TCM was derived. And through the distribution, the theoretic symbol error rate and bit error rate bound of the TCM was obtained. For decoding complexity, we compared the performance of several Viterbi decoder depths. Besides, due to the independence of the two branches punctured convolutional codes in TCM, the Viterbi decoder complexity can be reduced from one 256 states decoder to two 16 states decoders. Lastly, we simulated the decoding architecture in AWGN channel and compared results with the analytic results obtained previously. Those simulation results show that the TCM schemes have 5 dB gains when compared with classical QAM modulation in symbol error rate, and in bit error rate, there are 4 dB gains. Furthermore, the difference between the analytic value and the simulation results are less than 0.5 dB.

For equalization in J.83B, we simulated the performance of decision feedback equalizer with LMS algorithm and two blind algorithms—CMA and MMA. Then we proposed two different turbo equalizer algorithms based on the previous results. The first equalizer employs conventional Viterbi algorithm while the second equalizer involves soft-output Viterbi algorithm. When we compared DFE with MMA algorithm with SER 10^{-4} , the first turbo equalization structure derives 3.2 dB gain while the second algorithm obtains 3.4 dB gain. When it came to the comparison between DFE with LSM algorithm and turbo equalizers. The first algorithm acquires 0.21 dB gain. The second structure reaches 1.2 dB gain in low SNR and 0.43 dB gain in high SNR. And there is 0.8 dB gain between this turbo equalizer and ideal equalizer.

From these observations, turbo equalizer may not be essential in television entertainment. However, when those transmitted data can not be lost or when a short converge time is required, turbo equalizer is quite useful.

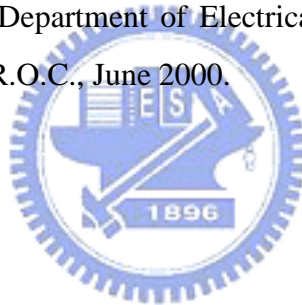
In future work, the performance improvement could be discussed. We could employ the MAP algorithm not only in the equalizer structure but also in the decoder. And the relationship between converters and noise variance could be derived. For further study, the interleaver gain and the theoretical performance of each iteration could be discussed. And the performance study under different channel condition could also be considered.

Bibliography

- [1] ITU-T Recommendation J.83, *Digital Transmission of Television Signals—Digital Multiprogramme Systems for Television, Sound and Data Services for Cable Distribution*. 1997.
- [2] S. Ovadia, *Broadband Cable TV Access Networks— From Technologies to Applications*, Prentice-Hall. 2001.
- [3] W. Ciciora, James Farmer, and David Large, *Modern Cable Television Technology*. San Francisco, California: Morgan Kaufmann, 1999.
- [4] A. Dutta-Roy, “An overview of cable modem technology and market perspectives”, *IEEE Commun. Mag.*, vol. 39, no. 6, June 2001.
- [5] E. R. Bartlett, *Cable Television Technology & Operations—HDTV and NTSC Systems*. McGraw-Hill, 1990.
- [6] E. R. Evans, *Digital Telephony Over Cable—The PacketCable Network*. Addison-Wesley, 2001.
- [7] I. S. Reed and X. Chen, *Error-Control Coding For Data Networks*. Kluwer Academic, 1999.
- [8] <http://www.eccpage.com/>.
- [9] L. F. Wei, “Rotationally invariant convolutional channel coding with expanded signal space—part I: 180°,” *IEEE J. Select. Areas Commun.*, vol. 2, no. 5, pp. 672–686, Sep. 1984.

- [10] L. F. Wei, "Rotationally invariant convolutional channel coding with expanded signal space—part II: nonlinear codes," *IEEE J. Select. Areas Commun.*, vol. 2, no. 5, pp. 659–671, Sep. 1984.
- [11] D. Haccoun, and G. Begin, "High—rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1113–1125 Nov. 1989.
- [12] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, 1995.
- [13] S. Haykin, *Adaptive Filter Theory*. Prentice-Hall, 2002.
- [14] D. W. Lin (principal investigator), K.-C. Hung, and M.-W. Liu, "Research in cable modem transceiver technology under high-order QAM," CCL project report no. T2-91033-1, Nov. 2002.
- [15] D. W. Lin (principal investigator), K.-C. Hung, and M.-W. Liu, "Advanced cable modem transceiver technology research," CCL project report no. T1-92019-6, Nov. 2003.
- [16] J. Yang, J Werner, and G. A. Dumont, "The multimodulus blind equalization and its generalized algorithms," *IEEE J. Select. Areas Commun.*, vol. 20, no. 5, pp. 997–1015, Jun. 2002.
- [17] J. P. Woodard and L. Hanzo, "Comparative study of turbo Decoding techniques: an overview," *IEEE Trans. Vehicular Technology*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.
- [18] R. Koetter, A. C. Singer, and M. Tuchler, "Turbo equalization," *IEEE Signal Processing Mag.*, vol. 21, no. 6, pp. 67–80, Jan. 2004.
- [19] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 41, pp. 704–713, May 1995.

- [20] M. Tuchler, R. Koetter, and A. C. Singer, "Turbo equalization: principles and new results," *IEEE Trans. Commun.*, vol. 501, no. 5, pp. 754–767, May 2002.
- [21] C. Laot, A. Glavieux, and J. Labat, "Turbo equalization: adaptive equalization and channel decoding jointly optimized," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, Step. 2001.
- [22] B. Vucetic, and J. Yuan, *Turbo Codes: Principles and Applications*. Boston, Mass.: Kluwer Academic, 2000.
- [23] J. Hagenauer, and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," *IEEE Global Telecommun. Conf. Rec.*, 1989, pp. 1680–1686.
- [24] C.-Y. Chien, "Software implementation and performance analyses for cable modem FEC code," M.S. thesis, Department of Electrical Engineering, National Chi Nan University, Puli, Taiwan, R.O.C., June 2000.



作者簡歷

劉明璋，民國六十九年出生於台中縣。民國九十一年六月畢業於國立交通大學電子工程學系，同年九月進入國立交通大學電子所就讀，從事通訊系統相關研究。民國九十三年六月取得碩士學位，碩士論文題目為『纜線數據機的更誤解碼與渦輪等化器研究』。研究範圍與興趣包括：通道編碼、等化器及通訊系統。

