

國立交通大學

電信工程學系

博士論文

使用訊號空間分割技術之適應性貝氏等化



Adaptive Asymptotic Bayesian Equalization
Using Signal Space Partitioning Techniques

研究生： 陳 仁 智

指導教授： 吳 文 榕

中 華 民 國 93 年 6 月

Adaptive Asymptotic Bayesian Equalization Using Signal Space
Partitioning Techniques

使用訊號空間分割技術之適應性貝式等化

研究生：陳仁智
指導教授：吳文榕 博士

Student: Ren-Jr Chen
Advisor: Dr. Wen-Rong Wu

國立交通大學

電信工程學系博士班



Submitted to Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Chiao-Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
In
Communication Engineering
June 2004
Hsinchu, Taiwan, Republic of China.

中華民國九十三年六月


使用訊號空間分割技術之適應性貝式等化

研究生：陳仁智

指導教授：吳文榕 教授

國立交通大學電信工程學系博士班

摘要



貝氏等化器是符元型等化器中最佳的等化器。然而貝氏等化器的複雜度通常都非常高。最近訊號空間分割的技術已經被應用來降低貝氏等化器的複雜度。這個方法所形成的決策邊界是由一組平面組成，這些平面是由一個系統化的狀態搜尋程序所找到的。然而此方法的缺點有二，第一是，這些平面的個數是由通道特性所決定，而不能由我們控制，另一個缺點是他的搜尋程序不是很有效率。在本論文中，我們提出兩種演算法來解決這些缺點並且探討他的可能應用。對於第一種演算法，我們提出一個近似之貝氏成本函數，使得這些平面的個數是可以自由給定的。這樣所設計出來的等化器可以在複雜度和效能間取得一個平衡。在很多情況下，所提出來的等化器可以犧牲一點效能卻可

以大大的降低複雜度。而這些平面的決定是由一個適應性的演算法所決定。這個適應性的方法非常的強健且複雜度低。

對於第二種演算法，我們將等化器的問題視為一個傳統分類問題。類似的看法最近也有人提出，現存之方法是使用非線性鑑別函數當作分類器。由於使用非線性鑑別器的緣故，往往很難同時達到很好的效能並且享有較低的複雜度。我們提出了一個新的方法來克服這樣的問題。我們的想法是利用比較多組的線性鑑別器來取代比較少組的非線性鑑別器。這樣的想法可以使鑑別器上的參數容易決定而且複雜度較低。而決定這些線性鑑別器的方法也跟第一個演算法類似。由於適應性的做法，所以我們所設計的等化器可以應用於時變的通道中。模擬顯示出我們所提出的方法可以很有效率的逼近貝氏等化器。

我們也同時將我們所提出的等化器應用於陣列天線的通訊系統中。這樣可以形成一個新的非線性空時等化器。這個等化器可以很有效率的逼近空時貝氏等化器。最後，我們針對非線性的最大似然序列估計等化器作了一些探討。眾所週知，在使用最大似然序列估計等化器必須知道通道的響應，然而非線性通道的響應並無一個通用的通道模型。我們提出了一個新的方法來克服這問題，使用這種方法我們完全無需通道模型，而且計算複雜度還可能較低。

Adaptive Asymptotic Bayesian Equalization Using Signal Space Partitioning Techniques

Student: Ren-Jr Chen

Advisor: Dr. Wen-Rong Wu

Institute of Communication Engineering
National Chiao-Tung University
Hsin-Chu, Taiwan 30050

Abstract

The Bayesian equalizer, with or without decision feedback, is known to be optimal for the symbol-by-symbol type of equalizer. However, the computational complexity for the Bayesian equalizer is usually very high. Recently, the signal space partitioning technique has been proposed to solve the problem. It was shown that the decision boundary for the Bayesian equalizer consists of a set of hyperplanes and a systematic state-search process was proposed to find these planes. The main problem of the existing approach is that the number of hyperplanes cannot be controlled. Also, the state-search process is not always efficient. In this dissertation, we propose two new algorithms to remedy these problems and explore their potential applications. For the first algorithm, we propose an approximate Bayesian criterion that allows the number of hyperplanes to be arbitrarily set. As a consequence, a tradeoff can be made between performance and computational complexity. In many cases, the resulting performance loss is small while the computational complexity reduction can be large. An adaptive method using stochastic gradient descent is also developed to identify the functions. The adaptive method is robust and has very low computational complexity.

For the second algorithm, we treat equalization as a classical pattern classification problem. This type of equalization approach has been proposed recently also. Existing algorithms employed nonlinear discriminant functions in the classifier. Due to nonlinear characteristics of the discriminant functions, it is found that the classifier is difficult to save significant computations and at the same time achieve satisfactory results. We propose a new discriminant function approach to overcome this problem. Our idea is to employ a large set of linear discriminant functions instead of a small set of nonlinear functions. By this manner, parameter identification becomes much easier and the computational complexity becomes lower. Similar to the first approach, the number of discriminant functions can be arbitrarily set and an easy trade-off between performance and computational complexity can be made. An adaptive method is developed such that the proposed algorithm is applicable in time-varying environments. Simulations show that our approaches can efficiently approximate the Bayesian equalizer. Also, the low complexity property makes the proposed equalizers suitable for real-world implementation.

We also apply the proposed algorithms to antenna array communication systems. This results in new nonlinear spatio-temporal equalizers. While these algorithms efficiently approximate the spatio-temporal Bayesian equalizers, they inherit other good properties of the temporal counterparts. Finally, we consider the maximum likelihood sequence estimation (MLSE) equalizer for nonlinear channels. It is known that the channel response is required in the MLSE equalizer. However, there does not exist a general model for nonlinear channels. We propose a new MLSE equalizer that does not require any channel modeling and the computational complexity can be much lower than the MLSE equalizer with channel modeling.

誌 謝

對於在艱辛的博士班旅程中所遇到的每個人，我感到非常的幸運跟快樂。首先，我非常感謝我的指導教授 吳文榕博士這五年來的指導跟鼓勵。無論他有多忙，他的辦公室總是開著，讓我們能隨時的找他討論任何我們所遇到的問題。在博士班的旅程遇到挫折的時候，他總是給我信心。非常幸運的我能從老師身上學習到很多特別的想法，很好的工程直覺，以及深入的技術。我希望我的思考和工作態度能塑造的跟老師一樣好。非常非常再次的感謝這樣願意隨時傾聽跟給忠告的指導老師。



我也非常感謝在我博士班的旅程經常和我討論的學長學弟，謝雨滔，楊華龍，許兆元，李彥文。他們經常和我交換他們的心得使我獲益匪淺。我也要感謝那些曾經跟我在同一個實驗室奮鬥的同伴，我不會忘記和她們一起相處快樂的時光，蘇文樹，林秋陪，陳郁夫，王志嘉，張國安，林千惠，崔義明，賴芳信，謝銘哲，吳俊育，蕭詩駿，劉啟帆，林大鈞，蔡國仁，沈英宗，許慶霖，李峰宇，俞丁發，莊秉卓，許獻澤，郭珈汶。

我也要感謝我以前逢甲大學的同學們，有他們的陪伴讓我這旅程不至孤單，危志豪，林偉捷，鄭必章，蔣德興，江盛攸，夏志強，莊承諭，蔣裕和，吳韻宜，潘育正。

我也不會忘記要感謝我的好友，劉欣華，洪楷萱，李明機，柳正忠。要感謝的人真的很多，請原諒我如果我忘記將您的名字寫上。

特別要感謝在我博士旅程的最後一年遇到我的女孩 陳嫻竹，感謝她的愛與支持陪我度過最艱辛的旅程。

最後，但也是最重要的,我要感謝我的父母 陳大福和 邱麗容，沒有他們無私的愛和鼓勵我將無法順利的拿到博士學位，非常感謝我的父母。



Acknowledgements

I feel **extremely** lucky and happy to all the people I have met during the intriguing and appealing journey of my Ph.D.

Foremost, I would like to thank my advisor Prof. Wen-Rong Wu for his guidance and support over the past five years. No matter how busy he was, his office door has always been open. I could always talk to him about any issue. As the frustrations depress me in the journey, I can only feel flattered for the confidence he has always shown in me. He always shown confidence in me and has given me a continuous support which I appreciate. Fortunately, I have benefited from his extraordinary motivation, great intuition, and technical insight. I just hope my thinking and working attitudes have been shaped according to such outstanding qualities. Very special and deep thanks again go to my advisor for such a great human being always ready to listen and give advice.

I will always be indebted to the people who always have discussion with me: Yu-Tao Hsieh, Hua-Lung Yang, Chao-Yuan Hsu, Yinman Lee. They always share with me their interesting comments on my questions.

I would also like to thank the people with whom I have had the pleasure to collaborate: Wen-Shu Su, Chiou-Pei Lin, Yu-Fu Chen, Jr-Chia Wang, Kuo-An Chang, Chian-Huei Lin, Yi-Ming Tsuei, Fang-Shin Lai, Ming-Je Hsieh, Jiun-Yu Wu, Jiun-Shr Shiau, Chi-Fan Liou, Da-Jiun Lin, Kuo-Ren Tsai, Ying-Tzung Shen, Ching-Lin Shiu, Feng-Yu Lee, Ding-Fa Yu, Bing-Juo Chuang, Shian-Tze Hsu, Jia-Wen Guo.

This long journey would have been nothing without all the people I have met. I want to thank my colleagues at Feng Chia University: Jr-Hau Wei, Wei-Jie Lin, Bi-Chang Cheng, Te-Hsing Chiang, Cheng-You Jiang, Jr-Chiang Shia, Cheng-Yu Chuang, Yu-He Chiang, Yun-Yi Wu, Yu-Jeng Pan. I could not possibly forget to thank my good friends: Hsin-Hua Liou, Kai-Shiuan Hung, Ming-Ji Lee, Cheng-Chung Liou. Please forgive me if your name should have been listed above and is missing.

Special thanks have to go to Yen-Ju Chen who I have encountered during my last year in my Ph.D. journey for her love and support.

Last, but not least, I would like to thank my parents, Da-Fu Chen and Li-Rong Chiou, and my whole family for their love and lasting encouragement.

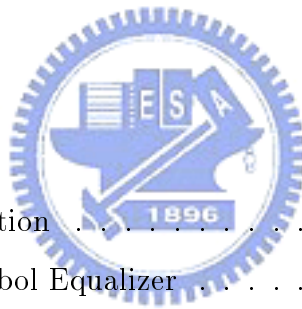
Ren-Jr Chen

May 2004



Contents

Abstract	iii
Acknowledgements	vii
List of Contents	ix
List of Tables	xii
List of Figures	xiv
1 Introduction	1
1.1 Problem of Equalization	1
1.2 The Symbol-by-Symbol Equalizer	2
1.3 New Approach	4
1.4 Organization of the Dissertation	6
2 Optimal Equalizers	8
2.1 Problem Formulation	8
2.2 Maximum Likelihood Sequence Estimation	9
2.3 Optimum Symbol-by-Symbol Equalizer	10
2.4 Comparison of the MLSE and Bayesian Equalizers	15
3 Suboptimal Symbol-by-Symbol Equalizers	17
3.1 Linear Equalizer and DFE	17
3.1.1 Linear Equalizer	17
3.1.2 Decision Feedback Equalizer	19



3.2	Minimum Bit-Error Rate Linear Equalizer	20
3.3	Volterra Equalizers	23
3.3.1	Volterra Series	23
3.3.2	Volterra Equalizer	25
3.3.3	MMSE Coefficient Identification	26
3.3.4	Adaptive Volterra Equalizer	27
3.3.5	Recursive and Decision Feedback Volterra Equalizers	29
3.3.6	Simulations	30
3.4	Neural Equalizers	34
3.4.1	Multi-Layer Perceptron Equalizer	34
3.4.2	Radial Basis Function Equalizers	37
3.5	Signal Space Partitioning Technique	38
3.6	The Discriminant Function Approach	41
4	Adaptive Asymptotic Bayesian Equalization Using a Signal Space Partitioning Technique	45
4.1	An approximate Bayesian Criterion	46
4.2	Equalization Using a Single Hyperplane	47
4.3	Equalization Using Multiple Hyperplanes	52
4.4	Decision Feedback Equalization	59
4.5	Computational Complexity	64
4.6	Simulation Results	65
5	Adaptive Asymptotic Bayesian Equalization Using a set of Discriminant Functions	74
5.1	Bayesian Equalization Using the Linear Discriminant Function Approach	75
5.2	Proposed Equalizer	77
5.3	Discriminative Learning Algorithm	79
5.4	Decision Feedback Equalization	82
5.5	A New cost function	86
5.6	Simulation Results	87

6 Adaptive Nonlinear Spatial and Temporal Equalization 95
6.1 Spatial and Temporal Channel Model 96
6.2 Adaptive Nonlinear Spatial-Temporal Equalization for Structure I 97
6.3 Adaptive Nonlinear Spatial-Temporal Equalization for Structure II 104

7 Maximum Likelihood Sequence Estimation for Nonlinear Channels 114
7.1 Branch Metric Calculation 114
7.2 Branch Metric Calculation Using Signal State Mapping 116
7.3 Simulation Results 118

8 Conclusion 123

References 125



List of Tables

4.1	The adaptive algorithm of proposed DFE	62
4.2	The adaptive algorithm for the proposed reduced storage DFE	63
4.3	Computational complexity comparison for the linear and proposed equalizers in the training phase	64
4.4	Computational complexity comparison for the linear, proposed, and Bayesian equalizers in the decision phase	65
4.5	Computational complexity comparison for the conventional and proposed DFEs in the training phase	65
4.6	Computational complexity comparison for the conventional, proposed, and Bayesian DFEs in the decision phase	65
4.7	Computational complexity comparison for the linear, proposed ($M = 8$), and Bayesian equalizers (for the nonlinear channel in simulations)	72
5.1	The adaptive algorithm of proposed equalizer with decision feedback.	84
5.2	The adaptive algorithm for the proposed reduced storage DFE.	92
5.3	The adaptive algorithm for the proposed fast convergent DFE.	93
5.4	Computational complexity comparison for the DFE and proposed equalizers with decision feedback signal in the training phase.	93
5.5	Computational complexity comparison for the linear, proposed, and Bayesian DFE in the decision phase.	94
5.6	Storage comparison for the linear, proposed, and Bayesian DFEs ($M=8$).	94
6.1	Computational complexity comparison for the linear and the proposed spatio-temporal equalizers	111

7.1 Computational complexity and storage comparison for the Volterra model
and the proposed method 118

7.2 Computational complexity and storage comparison for the Volterra model
and signal state mapping method 121



List of Figures

2.1	A typical digital communication system.	8
2.2	A block diagram of MSLE.	9
2.3	A block diagram of the symbol-by-symbol equalizer (without decision feedback).	11
2.4	A block diagram of the symbol-by-symbol equalizer with decision feedback signals.	12
2.5	Decision boundary of the Bayesian equalizer.	14
2.6	Decision boundary of the Bayesian equalizer.	15
3.1	A typical adaptive equalizer.	18
3.2	A typical linear equalizer with “tanh” function.	22
3.3	Learning curves (BER) for the MMSE, MBER, and LP equalizers.	24
3.4	Structure of a Volterra equalizer.	25
3.5	The adaptive Volterra equalizer.	28
3.6	The structure of VPE.	28
3.7	The structure of the recursive Volterra equalizer.	29
3.8	The structure of a BPDFE.	31
3.9	BER comparison for the linear, Volterra, Bayesian and MLSE equalizers.	32
3.10	BER comparison for the DFE, bilinear DFE, BPDFE, Bayesian DFE and Viterbi equalizers.	33
3.11	A three layers MLP (2,3,3,2).	34
3.12	The architecture of an MLP equalizer.	35
3.13	BER comparison for the linear equalizer, the MLP equalizer (4,9,4,1), and the Bayesian equalizer.	36

3.14	Structure of the RBF network.	37
3.15	Asymptotic Bayesian equalizer using a set of hyperplanes.	40
3.16	BER comparison for the linear, asymptotic Bayesian, and Bayesian equalizers.	41
3.17	BER comparison for the conventional, asymptotic Bayesian, and Bayesian DFEs.	42
3.18	Discriminant functions for equalization.	44
4.1	Decision boundaries for the proposed (dashed line) and the Bayesian (solid line) equalizer with two subsets.	51
4.2	The structure of the proposed nonlinear equalizer.	54
4.3	The decision region of Subset 1, \mathcal{R}_d^1	55
4.4	Decision regions for all subsets, \mathcal{R}_d^i , $i = 1, 2, 3, 4$, and decision boundaries of the proposed (dashed line) and the Bayesian (solid line) equalizer.	56
4.5	The structure of the proposed nonlinear DFE.	61
4.6	BER comparison for the MMSE linear, MBER linear, proposed, and Bayesian equalizers.	66
4.7	BER comparison for the proposed and RS Bayesian equalizers.	67
4.8	BER comparison for the MMSE linear, MBER linear, proposed and Bayesian equalizers.	68
4.9	Decision regions for all subsets, \mathcal{R}_d^i , $i = 1, 2, 3, 4$, and decision boundaries of the proposed equalizer when $M = 4$ (dashed line) and the Bayesian (solid line) equalizer.	69
4.10	Decision regions for all subsets, \mathcal{R}_d^i , $i = 1, 2, \dots, 8$, and the decision bound- aries of the proposed equalizer when $M = 8$ (dashed line) and the Bayesian (solid line) equalizer.	70
4.11	BER comparison for the MMSE linear, MBER linear, proposed, and Bayesian equalizers.	71
4.12	BER comparison for the MMSE linear, proposed, and Bayesian equalizers.	72
4.13	BER comparison for the Conventional DFE, proposed DFE, and Bayesian DFE.	73

5.1	BER comparison for the conventional, proposed (Table 5.1 and Table 5.2), and Bayesain DFEs.	88
5.2	BER comparison for the conventional, proposed (Table 5.1 and Table 5.3), and Bayesain DFEs.	89
5.3	Learning curves for the proposed DFEs in Table 5.1, 5.2 and 5.3 (SNR= 16 dB, $M = 8$).	90
5.4	BER comparison for the proposed DFEs in Table 5.1 and 5.2 (SNR= 16 dB, $M = 8$).	91
5.5	BER comparison for the proposed DFEs in Table 5.1 and 5.3 (SNR= 16 dB, $M = 8$).	94
6.1	Discrete-time spatial and temporal channel model.	96
6.2	Structure I spatial-temporal equalizer.	98
6.3	The signal states ($\theta_{1,0} = 0, \theta_{1,1} = \pi/2, \theta_{2,0} = \pi/2$, and $\theta_{2,1} = \pi/2$).	100
6.4	The signal states ($\theta_{1,0} = 0, \theta_{1,1} = \pi/2, \theta_{2,0} = \pi/10$ and $\theta_{2,1} = \pi/4$).	101
6.5	BER comparison for the linear MMSE and the proposed Structure I spatial-temporal equalizers.	104
6.6	BER comparison for the MMSE linear and the proposed Structure I spatio-temporal equalizers ($U=3, 4$ and 5).	105
6.7	BER comparison for the linear MMSE and the proposed Structure I spatio-temporal DFE ($U=3, 4$ and 5).	106
6.8	Structure II spatial-temporal equalizer.	107
6.9	BER comparison for the linear MMSE (in Fig. 6.8) and the proposed Structure II spatio-temporal equalizers.	111
6.10	BER comparison for the linear MMSE and the proposed spatio-temporal equalizers with Structure I and II (3 antennas).	112
6.11	BER comparison for the linear MMSE and the proposed Structure II spatio-temporal DFEs.	113
7.1	Learning curves for the linear model, the Volterra model, and the cluster means.	119

7.2 BER comparison for the Viterbi equalizer with the linear channel model, with the Volterra channel model, with the perfect channel model, and with the cluster means. 120

7.3 Learning curves for the linear channel model, the Volterra channel model, and the cluster means. 121

7.4 BER comparison of the Viterbi equalizer with the linear channel model, with the Volterra channel model, with the perfect channel model, and with the clustering means. 122



Chapter 1

Introduction

1.1 Problem of Equalization

Many digital communication channels suffer from intersymbol interference (ISI) due to their band-limited channel characteristics. Commonly used channels are often considered as linear. However, there exist some applications where the channels are nonlinear. Digital satellite and magnetic recording channels are two examples. In satellite communications, the satellite transponder and earth station amplifier usually operate near the saturation region [1], [2] and that cause the nonlinear effects. For magnetic recording devices, the problem arises during the writing process. This is because transitions written previously often cause the next transition to shift in position, and adjacent transitions also partially erase with each other. This results in a nonlinear amplitude distortion when the signal is read back [3]–[5]. Equalizers are the commonly used devices to compensate for these channel effects. Conventionally, equalizers are divided into two categories; one is the sequence estimation equalizer, and the other is the symbol-by-symbol equalizer.

The sequence estimation type of equalizers, as it named, estimates the whole sequence of the transmit symbols. During past years, several structures have been proposed. Chang and Hancock [6] proposed a structure whose computational complexity grows linearly with the sequence length and it may provide an optimum performance under a specific condition. Abend and Fritchman [7] proposed a recursive structure whose computational complexity does not grow with the received signal sequence. Finally, the well known maximum likelihood sequence estimation (MLSE), implemented by the Viterbi algorithm, was proposed by Forney [8]. Owing to its efficiency, this Viterbi-based MLSE becomes the

most popular algorithm for the sequence estimation equalizer. Although the MLSE is known to provide nearly optimal equalization performance, there are some concerns in real-world applications. The MLSE equalizer is computationally expensive and requires a sufficiently long decision delay. Also, the MLSE equalizer provides the optimal performance only in stationary channels. In nonstationary channels, the channel estimation errors may accumulate and this may seriously affect its performance. Also, application of MLSE equalizer in nonlinear channels may be troublesome. This is due to the lack of a unique nonlinear channel model.

The symbol-by-symbol type of equalizers estimate a single transmit symbol instead of the whole sequence. The main difference between the symbol-by-symbol equalizer from the sequence estimation equalizer is that the output of the symbol-by-symbol decision only depends on the a finite set of observations (finite memory); while the output of the MLSE equalizer depends on all past observations (infinite memory). To improve the performance, decisions can be feedback in the symbol-by-symbol equalizer. This type of equalizer can be linear or nonlinear. In general, nonlinear equalizers perform better, but their computational complexities are higher. One exception is the decision feedback equalizer (DFE) [9]. Being a nonlinear equalizer, The DFE enjoys low computational complexity and good performance. In many scenarios (nonlinear channels for example), however, the linear equalizer or the DFE cannot give satisfactory results and a more sophisticated nonlinear symbol-by-symbol equalizer is required. It has been shown that the optimum symbol-by-symbol equalizer is the Bayesian equalizer [10], and the optimum symbol-by-symbol DFE is the Bayesian DFE [11]. The computational complexity of these two optimal equalizers is considerably more than the linear equalizer and DFE, even more than the MLSE equalizer. Thus, many nonlinear equalizers with lower computational complexity have been proposed to approximate Bayesian equalizer or Bayesian DFE.

1.2 The Symbol-by-Symbol Equalizer

Nonlinear algorithms approximating the Bayesian equalizer generally have structures allowing a tradeoff between performance and computational complexity. These algorithms include the polynomial based nonlinear equalizer and the artificial neural network. The

polynomial based nonlinear equalizer have been reported in [1], [2], [12]–[18]. This approach uses the Volterra series to expand the input-output relationship of the Bayesian equalizer. It turns out that the output signal is a polynomial function of the input signal. To well approximate the Bayesian equalizer, the polynomial equalizer generally requires a large amount of parameters. In [19], [20], [21], a bilinear recursive polynomial equalizer was proposed to alleviate the problem. It was shown that this type of polynomial equalizer can be much more efficient. The neural network is known to be powerful in modeling nonlinear systems. In [22]–[27], multilayer-layer perceptrons (MLP) were proposed to serve as nonlinear equalizers. Another type of neural network applying to equalization is the radial basis function (RBF) network [28]–[32]. It has been shown that the RBF network is better than the MLP in the equalization problem because the structure of the RBF network has a closer relationship to the Bayesian equalizer. Disadvantages of these approaches are long training time and lack of methodologies for architecture selection. Also, their computational complexities may still be too high for many applications. Thus, other methods with reduce complexity were proposed [33]–[36].

Efficient algorithms for solving the Bayesian equalization problem were developed recently. The support vector machine (SVM) approach [37]–[38], a nonlinear modeling tool, was recently applied to nonlinear equalization. It was shown that the computational complexity of the SVM can be much lower than polynomial and neural network based equalizers. However, the learning algorithm in the SVM needs to solve a quadratic programming problem. The optimization method is somewhat computationally intensive. Another approach to the Bayesian equalizer uses the signal state partitioning technique. By treating the equalization problem as the classification problem, it has been shown that the Bayesian decision boundary consists of a set of hyperplanes when the signal to noise ratio (SNR) is infinite [33]. Y. Kim, Moon and S. Chen [39]–[42] employs the signal space partitioning technique to approach the Bayesian equalizer by a set of hyperplanes. The works in [40] and [41] used a combinatorial search and optimization process to find these planes. Despite its high computational complexity, this method does not guarantee to obtain the asymptotic Bayesian solution. It has been shown [39] that the hyperplanes can be formed by so called dominant signal state pairs. A simpler method was proposed in

[42] to search for these dominant pairs. This design guarantees to achieve the Bayesian solution asymptotically. The signal space partitioning techniques mentioned above all require channel information for signal state calculation. The number of hyperplanes and dominant states depends on channel characteristics. If the channel response changes, the hyperplanes must be re-calculated. Thus, these approaches are inefficient for time-varying channels. Another problem is that when decisions are included, the original signal space must be translated into a new space. If the channel is nonlinear, this is not an easy work.

Yet, there is another approach that treats equalization as a classical pattern classification problem [43], [44]. The equalizer was considered as a classifier with predefined discriminant functions. In [43], [44], neural networks were used as the discriminant functions and a discriminative learning algorithm [45] yielding the minimum classification rate was employed to train the discriminant functions. In the classical classification approach, the number of discriminant functions is set equal to the number of object classes. Thus, [43], [44] define N discriminant functions where N is the size of the symbol alphabet. The performance of this approach can approach that of the optimal Bayesian equalizer. However, since the optimal decision boundary is often highly nonlinear, the number of network layers required is large. It suffers the similar problems as those in the [33]–[36].

1.3 New Approach

From the above discussion, we know that the computational complexity is the central issue in the Bayesian equalization. Many studies have been devoted to the compromise the performance with complexity. As we can see, most of them are either computational not efficient or not applicable in time-varying channels. In this dissertation, we will consider the equalization problem from two perspectives and propose simple and effective algorithms to solve the problem. We consider the problem from the the signal space partitioning and from discriminant function point of views. Although these two approaches yield the similar results, their implications are different.

We first propose an adaptive asymptotic Bayesian equalizer using a signal space partitioning technique. In conventional signal space partitioning technique, the received signal space is seen as the union of two subsets when the transmitted signal is binary. Each sub-

set corresponds to the transmitted signal $+1$ or -1 . Hyperplanes are then searched and used to partition the received signal space. In our approach, we first divide the received signal space into M subsets, $M > 2$, and then merge them into two subsets. Based on this idea, we propose a new method that can provide an efficient approximation to the Bayesian equalizer. The resulting equalizer was similar to the equalizer in [39]–[42] which consists of a set of hyperplanes; however, the implication of these planes and the method for finding them are quite different. In existing methods, the number of hyperplanes is determined by channel responses. In our approach, the number of hyperplanes can be arbitrarily set. The hyperplanes found by the proposed algorithm are generally different from those found by the method in [39]–[42]. Our method allows an easy tradeoff between complexity and performance. In many cases, we can make the performance loss small while the computational complexity reduction is large. Another feature is that the parameters of these hyperplanes can be adaptively identified using a stochastic gradient descent (SGD) method. As a result, the proposed equalizer can be effectively applied to time-varying channels. Signal detection is performed using a set of parallel linear discriminant functions followed by a maximum operation. The computational complexity of the proposed equalizer is low and suitable for real-world implementation. Using the similar idea, we also extend our approach to approximate the Bayesian DFE.

We then propose adaptive asymptotic Bayesian equalizers using the discriminant function approach. Due to nonlinear characteristics of discriminant functions, the classifiers in [43], [44] are difficult to save significant computations and at the same time achieve satisfactory results. In [43], [44], observations for a possible transmit symbol are mapped to a class and a nonlinear discriminant function (neural network) function is developed for the class. In the proposed method, we let observations for a possible transmit symbol be mapped to multiple classes and multiple linear discriminant functions are derived for the class. In other words, we employ a large set of linear discriminant functions instead of a small set of nonlinear functions in the equalizer. We develop a mapping method that is independent of channel responses and can arbitrarily set the number of discriminant functions. This allows a easy trade-off between performance and computational complexity. We also develop an adaptive method that can identify the linear discriminant functions

and make the proposed algorithm applicable in time-varying environments. Except for the feedforward equalization algorithm, we have also consider efficient and fast convergent decision feedback algorithms. The strategy to use the decision feedback signal is different from that in [39] and [41]. The choice of discriminant functions does not depend on channel responses and the equalizer design is the same for linear and nonlinear channels.

All the equalizer we have discussed are all in the temporal domain. Introducing antenna arrays in communication systems, we can extend equalization into spatial domain and this can dramatically increase system performance. Spatio-temporal equalization algorithms have been reported in literature [46]–[50]. However, they are confined in simple structures and the performance is not always satisfactory. Applying the proposed equalization idea in spatio-temporal systems is straightforward; however, the performance enhancement is significant. Finally, we discuss the MLSE equalization in nonlinear systems. As we known, the MLSE equalizer requires the channel response and this is simple for linear channels. However, it may be problematic for nonlinear channels. This is because there does not exist a general model for nonlinear channels. Even we can have a model, determination of the number of parameters becomes a next problem. We propose a method that completely remove these problems. This method does not require any channel modeling and the computational complexity can be lower compared to the the one with channel modeling.

1.4 Organization of the Dissertation

This dissertation contains seven chapters in addition to this introductory chapter. In Chapter 2, we formulate the equalization problem and review the MLSE and Bayesian equalizers. We also briefly discuss the classical pattern recognition problem and the SGD training algorithm in adaptive signal processing.

In Chapter 3, we discuss existing approaches in symbol-by-symbol equalizers. First, we describe the conventional linear equalizer and the DFE and show that the MMSE criterion is not adequate for coefficient identification. We then illustrate how the the cost function can be modified to achieve the minimum bit error rate (MBER). After that, we describe the Volterra equalizer, the neural network equalizer, and the signal space

partitioning technique, and the discriminant function approach.

In Chapter 4, we develop the adaptive asymptotic Bayesian equalization using a signal space partitioning technique in detail. We will show how the Bayesian equalization can be approximated and how the computations can be saved. A new cost function is also proposed such that the proposed equalizer can be adequately identified. The identification is implemented using an adaptive manner. The computational complexity of the proposed adaptive algorithm is also evaluated.

In Chapter 5, we develop the adaptive asymptotic Bayesian equalizer using the discriminant function approach. As mentioned, we attack the equalization problem from the classical pattern recognition point of view. We use the cost function in [45] to identify the parameters of the discriminant functions and propose efficient learning algorithms for reducing the computational complexity and storage.

In Chapter 6, we extend the algorithm developed in Chapter 5 to antenna array communication systems and this results in spatio-temporal nonlinear equalizers. Two structure are considered; one has better performance but the computational complexity is higher, and the other has opposite characteristics. We show that our spatio-temporal equalizers can significantly outperform the conventional ones.

In Chapter 7, we proposed an efficient MLSE equalizer for nonlinear channels. As described, the main advantage of this equalizer is that it does not require any channel modeling. This is particular useful when the nonlinear characteristics of the channel is not well known. We also show that not only the performance can be improved, but also the computational complexity can be reduced.

In Chapter 7.3, we draw some concluding remarks and suggest the potential research topics for further research.

Chapter 2

Optimal Equalizers

2.1 Problem Formulation

A typical digital communications system is shown in Fig. 2.1, where $x(n)$ denotes the

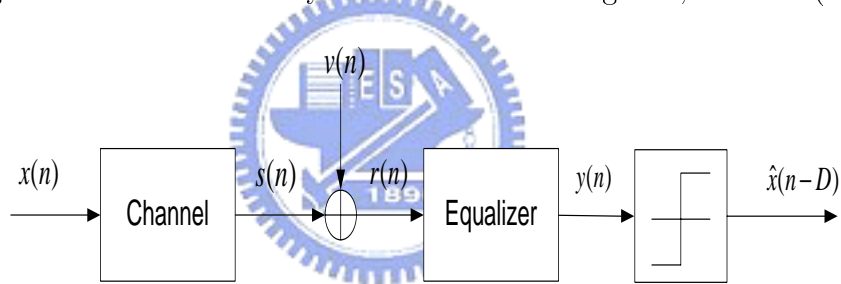


Figure 2.1: A typical digital communication system.

transmitted symbol, $s(n)$ the channel output, $v(n)$ the channel noise, $r(n)$ the received signal, $y(n)$ the equalizer output. Let L_c be the memory length of the channel, $\mathbf{x}'(n) = [x(n), x(n-1), \dots, x(n-L_c+1)]^T$ be the channel input vector, and the channel input and output be characterized by a mapping function $\psi(\cdot)$. We then have the following relationship:

$$\begin{aligned} r(n) &= s(n) + v(n) \\ &= \psi(\mathbf{x}'(n)) + v(n). \end{aligned} \tag{2.1}$$

To simplify the problem, we assume that $x(n)$ is a binary phase shift keying (BPSK) signal, i.e., $\{x(n) = \pm 1\}$, and $v(n)$ is an additive white Gaussian noise (AWGN) with

variance σ_v^2 . The mapping function $\psi(\cdot)$ can be linear or nonlinear. If the channel is linear, $\psi(\mathbf{x}'(n))$ can be expressed as

$$\psi(\mathbf{x}'(n)) = \sum_{k=0}^{L_c-1} c_k x(n-k), \quad (2.2)$$

where $\{c_k\}$ is the channel impulse response. The mission of the equalizer is to estimate the transmitted signal $x(n)$. As mentioned in Chapter 1, the equalizer could be divided into two types; one is the sequence estimation type and the other is the symbol-by-symbol type. In this chapter, we discuss the optimal sequence estimation and symbol-by-symbol equalizers, namely, the MLSE and the Bayesian equalizer.

2.2 Maximum Likelihood Sequence Estimation

Let \mathbf{r}_a be a received signal sequence corresponding to a transmitted signal sequence \mathbf{x}_a , where $\mathbf{r}_a = [r(0), r(1), \dots, r(L_x - 1)]^T$, $\mathbf{x}_a = [x(0), x(1), \dots, x(L_x - 1)]^T$ and L_x is the number of the transmitted signal symbols. Note here that we let the length of \mathbf{r}_a and \mathbf{x}_a be equal for notational simplicity. This implicitly assumes that the first $L_c - 1$ elements of \mathbf{x}_a have zero values. Let \mathcal{X} be a set of all possible combinations of $\mathbf{x}'(n)$ and Θ be a set of all possible combinations of \mathbf{x}_a . In absence of noise, $\psi(\mathbf{x}'(n))$ has only finite possible values ψ_i , $0 \leq i \leq 2^{L_c} - 1$. Each ψ_i corresponds to an element in \mathcal{X} . Let Φ denote the set containing all possible ψ_i 's. The MLSE equalizer estimates the transmitted signal sequence \mathbf{x}_a by maximizing the probability $P(\mathbf{r}_a|\mathbf{x})$, $\mathbf{x} \in \Theta$. Fig. 2.2 depicts the block diagram of the MLSE equalizer. Thus, the decision rule for the MLSE equalizer can be



Figure 2.2: A block diagram of MSLE.

express as

$$\hat{\mathbf{x}}_a = \arg \max_{\mathbf{x} \in \Theta} P(\mathbf{r}_a | \mathbf{x}) \quad (2.3)$$

$$= \arg \max_{\mathbf{x} \in \Theta} \prod_{n=0}^{L_x-1} \frac{1}{\sqrt{2\pi}\sigma_v} \exp\left(-\frac{(r(n) - \psi(\mathbf{x}'(n)))^2}{2\sigma_v^2}\right) \quad (2.4)$$

$$= \arg \min_{\mathbf{x} \in \Theta} \sum_{n=0}^{L_x-1} (r(n) - \psi(\mathbf{x}'(n)))^2 \quad (2.5)$$

$$= \arg \min_{\mathbf{x} \in \Theta} \sum_{n=0}^{L_x-1} \sum_{i=0}^{2^{L_c}-1} (r(n) - \psi_i)^2. \quad (2.6)$$

The above decision rule can be efficiently implemented by the Viterbi algorithm. The Viterbi algorithm utilizes a trellis diagram to compute the path metrics. Each path metric corresponds to a distance value (for a ψ_i) in (2.6). A possible pattern of $\mathbf{x}'(n)$ is called a state. Each state in the trellis diagram is assigned a value, called the partial path metric. The partial path metric is determined from a beginning state at time $n = 0$ to a particular ending state at time $n = k$. At each ending state, the “best” partial path metric is chosen from the paths terminated at that state. The selected metric represents the survivor path and the remaining metrics represent the nonsurvivor paths. The survivor paths are stored while the nonsurvivor paths are discarded in the trellis diagram. The Viterbi algorithm selects the signal survivor path left at $n = L_x - 1$. The path is the ML path. Trace-back of the ML path on the trellis diagram would then provides the ML estimated sequence.

2.3 Optimum Symbol-by-Symbol Equalizer

We first discuss the optimum symbol-by-symbol equalizer without decision feedback. Fig. 2.3 shows the block diagram of the symbol-by-symbol equalizer (without decision feedback). The optimum symbol-by-symbol equalizer is known as the Bayesian equalizer. The symbol-by-symbol equalizer processes a block of received signals $\mathbf{r}(n) = [r(n), r(n-1), \dots, r(n-L_e+1)]^T$, where L_e is the memory size of the equalizer, and produce an estimate of $\hat{x}(n-D)$ (D is the desired output delay). From (2.1), we have

$$\mathbf{r}(n) = \mathbf{s}(n) + \mathbf{v}(n), \quad (2.7)$$

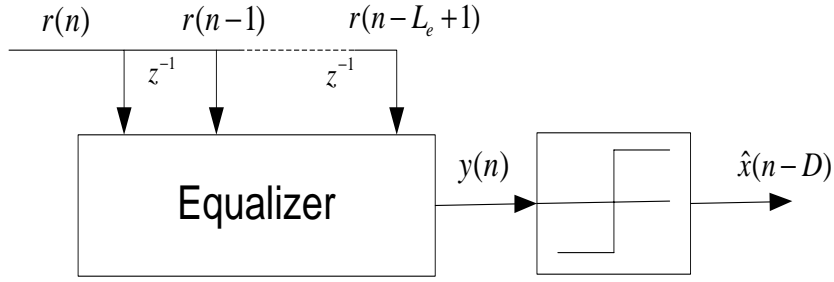


Figure 2.3: A block diagram of the symbol-by-symbol equalizer (without decision feedback).

where $\mathbf{v}(n) = [v(n), v(n-1), \dots, v(n-L_e+1)]^T$ is a noise vector, and $\mathbf{s}(n) = [s(n), s(n-1), \dots, s(n-L_e+1)]^T$ is the received signal vector. If noise is absent, we can have the received signal as

$$\mathbf{s}(n) = \mathbf{h}(\mathbf{x}(n)), \quad (2.8)$$

where

$$\mathbf{x}(n) = [x(n), x(n-1), \dots, x(n-D+1), x(n-D), x(n-D-1), \dots, x(n-L_c-L_e+2)]^T, \quad (2.9)$$

is a data vector and $\mathbf{h}(\cdot)$ denotes a vector mapping function.

We can then see that there are N_a , which is $2^{L_c+L_e-1}$, possible combinations of $\mathbf{s}(n)$. We call $\mathbf{s}(n)$ the signal vector and its possible values signal states, which are denoted as \mathbf{s}_i , $1 \leq i \leq N_a$. Let \mathcal{R}^{L_e} be the L_e dimension Euclidean space. The set of all possible signal vectors is defined by

$$\mathcal{S} \triangleq \{\mathbf{s}_i \in \mathcal{R}^{L_e}, 1 \leq i \leq N_a\}. \quad (2.10)$$

We call \mathcal{S} the signal state set. Two subsets of \mathcal{S} are defined by

$$\begin{aligned} \mathcal{S}^\pm &\triangleq \{\mathbf{s}_i \in \mathcal{S} : x(n-D) = \pm 1\}, \\ \mathcal{S} &= \mathcal{S}^+ \cup \mathcal{S}^-. \end{aligned} \quad (2.11)$$

The equalizer, which can be seen as a classifier, is then used to classify the received signal space (in \mathcal{R}^{L_e}) into two regions and assign each region a decision value. Under the AWGN scenario, it is shown in [31] that the optimum Bayesian equalizer is

$$\hat{x}(n-D) = \begin{cases} +1, & f_B(\mathbf{r}(n)) > 0 \\ -1, & f_B(\mathbf{r}(n)) \leq 0 \end{cases}, \quad (2.12)$$

where

$$f_B(\mathbf{r}(n)) = \chi^{(+)}(\mathbf{r}(n)) - \chi^{(-)}(\mathbf{r}(n)), \quad (2.13)$$

and

$$\chi^{(\pm)}(\mathbf{r}(n)) = \sum_{\mathbf{s}_i \in \mathcal{S}^{\pm}} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma_v^2}\right). \quad (2.14)$$

Next, we take the decision feedback signal into consideration. The block diagram of the symbol-by-symbol equalizer with decision feedback is shown in Fig. 2.4. As shown

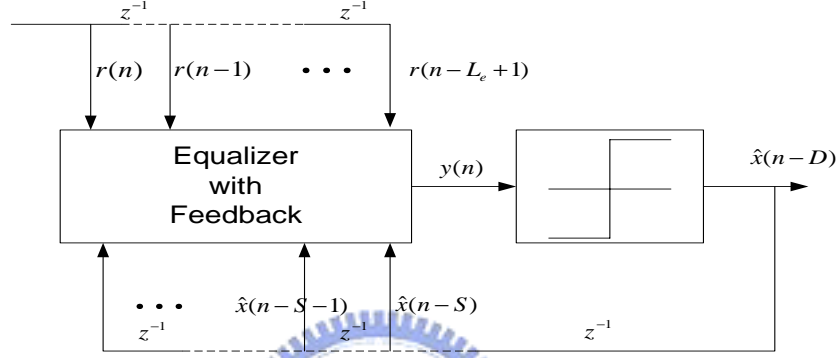


Figure 2.4: A block diagram of the symbol-by-symbol equalizer with decision feedback signals.

in the figure, the inputs of the equalizer are $\mathbf{r}(n)$ and the decision feedback vector $\hat{\mathbf{x}}_b(n)$, where $\hat{\mathbf{x}}_b(n) = [\hat{x}(n-S), \hat{x}(n-S-1), \dots, \hat{x}(n-L_c-L_e+2)]^T$. The instant $n-S$ denotes the starting time instant of the feedback signal. Thus, $S \leq D+1$. Let the length of the decision feedback vector $\hat{\mathbf{x}}_b(n)$ be L_b . Then, we have $N_b = 2^{L_b}$ possible feedback signal patterns. We call each pattern as a decision feedback state. Let $\hat{\mathbf{x}}_j$ be one possible decision feedback state, $1 \leq j \leq N_b$. Also, we collect all possible decision feedback vectors into the set

$$\mathcal{B} \triangleq \{\hat{\mathbf{x}}_j \in \mathcal{R}^{L_b}, 1 \leq j \leq N_b\}, \quad (2.15)$$

and call \mathcal{B} the decision feedback state set. Given a decision feedback state, we can then divide the set \mathcal{S} into N_b subsets

$$\mathcal{S}_j \triangleq \{\mathbf{s}_{i,j} \in \mathcal{S}, 1 \leq i \leq N_d : \hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j\}, \quad (2.16)$$

$$\mathcal{S} = \bigcup_{1 \leq j \leq N_b} \mathcal{S}_j. \quad (2.17)$$

where $\mathbf{s}_{i,j}$ denote one possible signal vector when $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$. In each \mathcal{S}_j , there are $N_d = N_a/N_b$ states. Each \mathcal{S}_j can further be divided into two subsets according to the value of $x(n - D)$

$$\mathcal{S}_j^\pm \triangleq \{\mathbf{s}_{i,j} \in \mathcal{S}_j : x(n - D) = \pm 1\}, \quad (2.18)$$

$$\mathcal{S}_j = \mathcal{S}_j^+ \cup \mathcal{S}_j^-. \quad (2.19)$$

If all decisions are correct and noise is Gaussian, it was shown that [11] the optimum Bayesian DFE is

$$\hat{x}(n - D) = \begin{cases} +1, & f_B(\mathbf{r}(n), \hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j) > 0 \\ -1, & f_B(\mathbf{r}(n), \hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j) \leq 0 \end{cases}, \quad (2.20)$$

where

$$f_B(\mathbf{r}(n), \hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j) = \chi_j^{(+)}(\mathbf{r}(n)) - \chi_j^{(-)}(\mathbf{r}(n)), \quad (2.21)$$

and

$$\chi_j^{(\pm)}(\mathbf{r}(n)) = \sum_{\mathbf{s}_{i,j} \in \mathcal{S}_j^\pm} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_{i,j}\|^2}{2\sigma_v^2}\right). \quad (2.22)$$

Here, we use two simple examples to illustrate the nonlinear behavior of the Bayesian equalizer.

Example 1:

Consider a linear channel with memory size $L_c = 2$. Let the channel responses be $c_0 = 1$ and $c_1 = 0.5$. Also, let the noise variance σ_v^2 be 0.05 and the memory size of the equalizer be $L_e = 2$ and $D = 0$. Thus, $\mathbf{x}(n) = [x(n), x(n - 1), x(n - 2)]^T$ and eight signal states result. The signal state set can then be divided into two subsets according to $x(n) = 1$ and $x(n) = -1$.

$$\mathcal{S}^+ = \{\mathbf{s}_1 = [1.5, 1.5]^T, \mathbf{s}_2 = [1.5, 0.5]^T, \mathbf{s}_3 = [0.5, -0.5]^T, \mathbf{s}_4 = [0.5, -1.5]^T\},$$

$$\mathcal{S}^- = \{\mathbf{s}_5 = [-0.5, 1.5]^T, \mathbf{s}_6 = [-0.5, 0.5]^T, \mathbf{s}_7 = [-1.5, -0.5]^T, \mathbf{s}_8 = [-1.5, -1.5]^T\}.$$

The decision boundary of the Bayesian equalizer (2.12) is shown in Fig. 2.5. In the figure, the symbol ‘o’ denotes the signal state corresponding to $x(n) = +1$ and ‘x’ denotes the signal state corresponding to $x(n) = -1$. \mathcal{R}_d^+ and \mathcal{R}_d^- are the decision regions corresponding

to $x(n) = 1$ and $x(n) = -1$, respectively. They can be expressed

$$\mathcal{R}_d^+ = \{\mathbf{r}(n) \in \mathcal{R}^{L_e} : f_B(\mathbf{r}(n)) > 0\}, \quad (2.23)$$

$$\mathcal{R}_d^- = \{\mathbf{r}(n) \in \mathcal{R}^{L_e} : f_B(\mathbf{r}(n)) \leq 0\}. \quad (2.24)$$

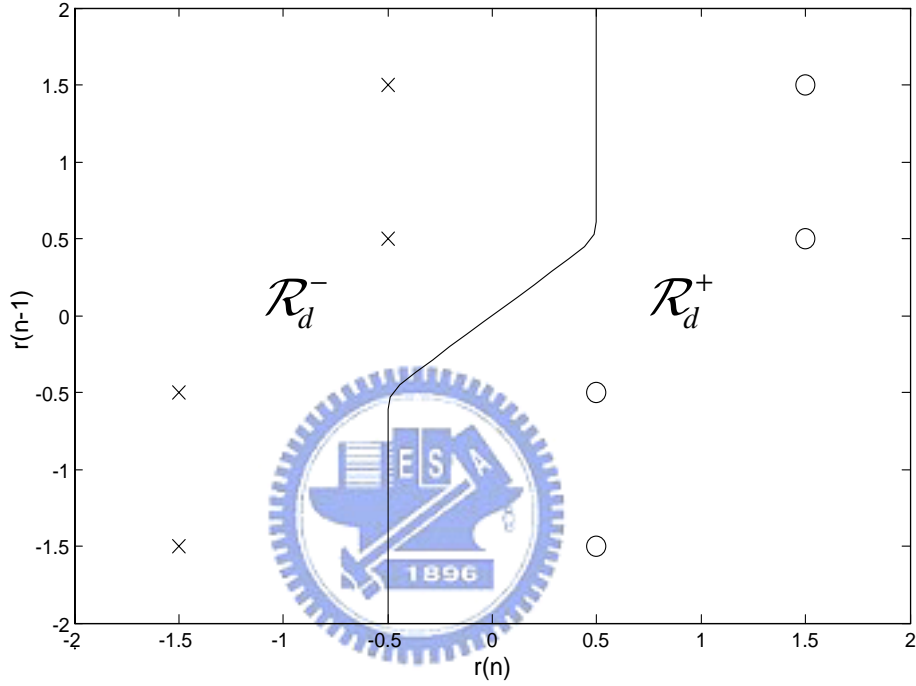


Figure 2.5: Decision boundary of the Bayesian equalizer.

Example 2:

Consider a linear channel with memory length $L_c = 2$ and let the channel responses be $c_0 = 0.5$ and $c_1 = 1$. Also, let the memory size of the equalizer be $L_e = 2$ and $D = 0$. In this scenario, the received signal space \mathcal{R}^{L_e} is not linearly separable. The decision boundary of the Bayesian equalizer (2.12) is shown in Fig. 2.6.

From both figures, we can clearly see that the equalization problem indeed can be considered as a classification problem (i.e., to classify the received signal space into two regions). The other thing we can observe is that the decision boundary of the Bayesian equalizer can be highly nonlinear. It can be easily verified that if the equalizer is linear, the decision boundary is linear too. Thus, in the scenario considered in Example 2, a

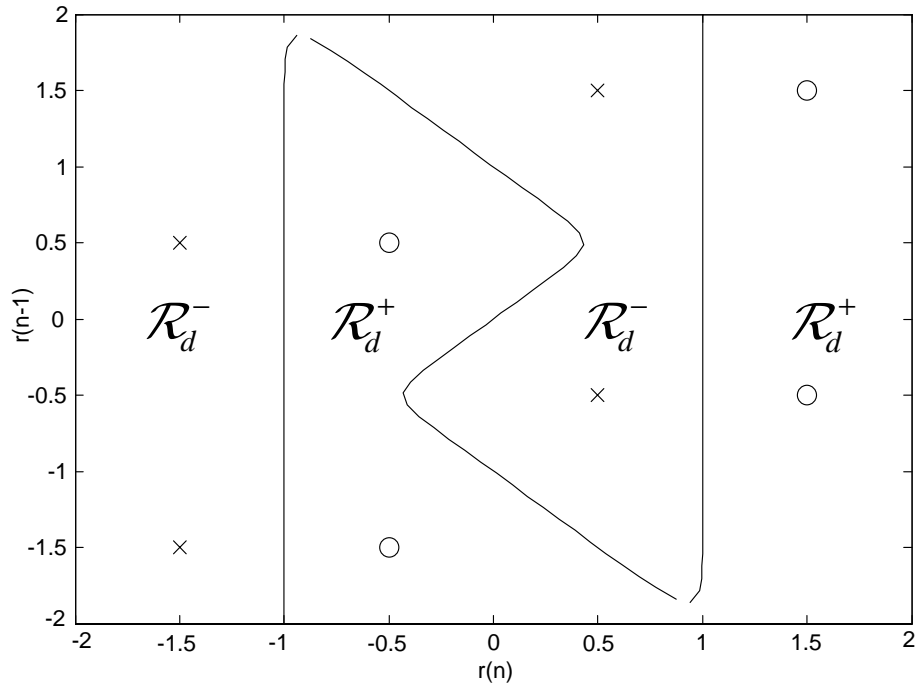
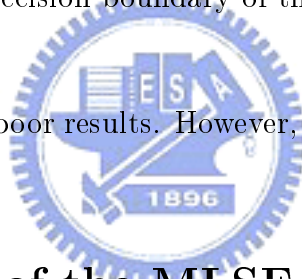


Figure 2.6: Decision boundary of the Bayesian equalizer.

linear equalizer will result in poor results. However, the Bayesian equalizer is not affected at all.



2.4 Comparison of the MLSE and Bayesian Equalizers

The performance of the Bayesian equalizer can asymptotically converge to the MLSE as L_e is increased. However, the computational complexity of the Bayesian equalizer will grow higher than the MLSE equalizer. So, what is the advantage of the Bayesian equalizer? In [51], the authors have discussed this problem in detail. There are at least two advantages of the Bayesian equalizer. The first one is that there are many suboptimal algorithms that can well approximate the Bayesian equalizer. It is not necessary to use the Bayesian equalizer with its original form. These suboptimal approaches are able to trade performance for computational complexity. It is often possible that a suboptimal algorithm can significantly reduce the computational complexity and still maintain good performance. In the following chapters, we will review some existing approximations and

develop new approaches. The tracking capability to nonstationary channels is another advantage of the Bayesian equalizer. S. Chen et al. [11] have conducted computer simulations to compare the performance of an radial basis DFE (which is an approximation to the Bayesian DFE) and an MLSE equalizer. The nonstationary channel considered was a mobile radio channel. The results showed that the radial basis DFE outperforms the MLSE equalizer. The degradation of the MLSE equalizer is due to the accumulation of channel tracking errors.



Chapter 3

Suboptimal Symbol-by-Symbol Equalizers

In this chapter, we briefly describe some existing methods that approximate the Bayesian equalizer.

3.1 Linear Equalizer and DFE

3.1.1 Linear Equalizer

Using the structure in Fig. 2.3, we can express the output of a linear equalizer as

$$y(n) = \mathbf{f}^T \mathbf{r}(n), \quad (3.1)$$

where $\mathbf{f} = [f_0, f_1, \dots, f_{L_e-1}]^T$ is the equalizer coefficient vector. As we can see, the output of the linear equalizer is a linear combination of the received signal $\mathbf{r}(n)$. To determine \mathbf{f} , a minimum mean-square error (MMSE) criterion is usually used. The cost function can then be expressed as

$$J(n) = E[e^2(n)] = E[(x(n-D) - y(n))^2], \quad (3.2)$$

where $E[\cdot]$ denotes the statistical expectation. With this cost function, it is apparent that the equalizer tries to estimate the desired signal $x(n-D)$.

It is well-known [17] that the optimal \mathbf{f} minimizing $J(n)$ is the Wiener solution given by

$$\mathbf{f}_{mmse} = \mathbf{R}_{\mathbf{r}(n)\mathbf{r}(n)}^{-1} \mathbf{p}_{x(n-D)\mathbf{r}(n)}, \quad (3.3)$$

where

$$\mathbf{R}_{\mathbf{r}(n)\mathbf{r}(n)} = E[\mathbf{r}(n)\mathbf{r}^T(n)], \quad (3.4)$$

$$\mathbf{p}_{x(n-D)\mathbf{r}(n)} = E[x(n-D)\mathbf{r}(n)], \quad (3.5)$$

are the correlation matrix of the received vector $\mathbf{r}(n)$ and the cross-correlation vector of the desired response signal $x(n-D)$ and received vector $\mathbf{r}(n)$, respectively. Substituting (3.3) into (3.2), we can then obtain the MMSE value as

$$\begin{aligned} \xi_{mmse} &= E[(x(n-D) - \mathbf{f}_{mmse}^T \mathbf{r}(n))^2] \\ &= E[x^2(n-D)] - 2\mathbf{f}_{mmse}^T \mathbf{p}_{x(n-D)\mathbf{r}(n)} + \mathbf{f}_{mmse}^T \mathbf{R}_{\mathbf{r}(n)\mathbf{r}(n)} \mathbf{f}_{mmse} \\ &= E[x^2(n-D)] - \mathbf{f}_{mmse}^T \mathbf{p}_{x(n-D)\mathbf{r}(n)}. \end{aligned} \quad (3.6)$$

As we can see, the Wiener solution involves matrix inverse and multiplication operations which are computational extensive. Also, the channel may be time-varying. Thus, adaptive equalizer are more useful in real-world applications. Fig. 3.1 shows a typical adaptive equalizer architecture. An adaptive filter uses the steepest-descent method to

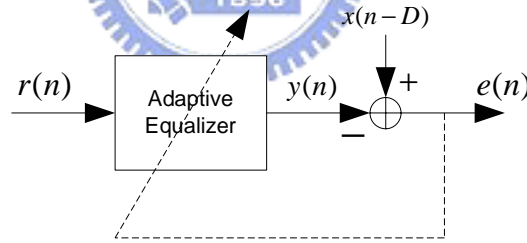


Figure 3.1: A typical adaptive equalizer.

adjust its tap weights iteratively. Let the cost function to minimize be

$$J(n) = E[\beta(\mathbf{f}(n))], \quad (3.7)$$

where $\beta(\cdot)$ is a function of $\mathbf{f}(n)$.

In the steepest descent method, the coefficients are updated using the following strategy:

$$\mathbf{f}(n+1) = \mathbf{f}(n) - \mu \nabla_{\mathbf{f}(n)} E[\beta(\mathbf{f}(n))], \quad (3.8)$$

where $\nabla_{\mathbf{f}}$ is the gradient vector with respect to \mathbf{f} , and μ is a step size that controls the convergence, tracking, and steady-state properties of the filter. In reality, however, exact measurement of the gradient vector is not possible. The gradient vector must be estimated from the available data. The simplest estimate is the stochastic gradient in which the expectation operations in (3.8) is ignored. Then, we have

$$\mathbf{f}(n+1) = \mathbf{f}(n) - \mu \nabla_{\mathbf{f}(n)} \beta(\mathbf{f}(n)). \quad (3.9)$$

Even though this gradient estimate is rather simple, the adaptive algorithm works reasonably well. The resultant algorithm is referred to as least mean square (LMS) adaptive algorithm. Back to the equalization problem, the cost function is defined as

$$J(n) = E[\beta(\mathbf{f}(n))] = E[e^2(n)]. \quad (3.10)$$

Substituting (3.10) in (3.8) and performing some algebra simplification, we then obtain the LMS adaptive algorithm as

$$e(n) = x(n-D) - \mathbf{f}^T(n) \mathbf{r}(n), \quad (3.11)$$

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu e(n) \mathbf{r}(n). \quad (3.12)$$

3.1.2 Decision Feedback Equalizer

As shown in Fig. 2.4, the DFE output (with $S = D - 1$) is a linear combination of the received signals and feedback decisions.

$$y(n) = \mathbf{f}^T \mathbf{r}(n) + \mathbf{b}^T \hat{\mathbf{x}}_b(n), \quad (3.13)$$

where $\mathbf{b} = [b_0, b_1, \dots, b_{L_b-1}]^T$ and $\hat{\mathbf{x}}_b(n) = [\hat{x}(n-D-1), \hat{x}(n-D-2), \dots, \hat{x}(n-D-L_b)]^T$.

Similar to the linear equalizer, the DFE coefficients can be identified by minimizing the MSE.

$$J(n) = E[(x(n-D) - y(n))^2]. \quad (3.14)$$

The Wiener solution for the DFE is then

$$\tilde{\mathbf{f}}_{mmse} = \mathbf{R}_{\mathbf{r}(n)\mathbf{r}(n)}^{-1} \mathbf{p}_{x(n-D)\mathbf{r}(n)}, \quad (3.15)$$

where

$$\mathbf{R}_{\tilde{\mathbf{r}}(n)\tilde{\mathbf{r}}(n)} = E[\tilde{\mathbf{r}}(n)\tilde{\mathbf{r}}^T(n)], \quad (3.16)$$

$$\mathbf{p}_{x(n-D)\tilde{\mathbf{r}}(n)} = E[x(n-D)\tilde{\mathbf{r}}^T(n)], \quad (3.17)$$

$$\tilde{\mathbf{f}} = [\mathbf{f}^T, \mathbf{b}^T]^T, \quad (3.18)$$

$$\tilde{\mathbf{r}}(n) = [\mathbf{r}(n), \hat{\mathbf{x}}_b(n)]^T. \quad (3.19)$$

And, the LMS adaptive algorithm is given by

$$e(n) = x(n-D) - y(n), \quad (3.20)$$

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu e(n)\mathbf{r}(n), \quad (3.21)$$

$$\mathbf{b}(n+1) = \mathbf{b}(n) + \mu e(n)\hat{\mathbf{x}}_b(n). \quad (3.22)$$

3.2 Minimum Bit-Error Rate Linear Equalizer

In this section, we consider a linear equalizer that minimizes the bit-error rate criterion. The MBER equalizer first appeared in [52]. This work showed that the continuous-time linear equalizer that minimizes BER can be represented as a matched filter followed by a tapped-delay line filter. The first adaptive algorithm for the MBER equalizer was proposed in [53]. Here, we mainly discuss the adaptive MBER equalizer proposed in [54] for its simplicity.

Consider the linear channel with coefficients c_i for $0 \leq i \leq L_c - 1$. The received signal can be expressed as

$$r(n) = \sum_{k=0}^{L_c-1} c_k x(n-k) + v(n). \quad (3.23)$$

where $v(n)$ is an AWGN with zero mean and variance σ_v^2 . Thus, the received signal vector $\mathbf{s}(n)$ in (2.7) becomes

$$\mathbf{s}(n) = \mathbf{C}\mathbf{x}(n), \quad (3.24)$$

where \mathbf{C} is a Toeplitz convolution matrix containing the channel coefficients c_i 's.

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots & 0 \\ 0 & c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{L_c-1} \end{bmatrix}. \quad (3.25)$$

The Wiener solution for the linear equalizer \mathbf{f} is

$$\mathbf{f}_{mmse} = (\mathbf{C}\mathbf{C}^T + \sigma_v^2\mathbf{I})^{-1}\mathbf{c}_{D+1}, \quad (3.26)$$

where \mathbf{c}_{D+1} is the $(D+1)$ th column of \mathbf{C} and \mathbf{I} is an $L_e \times L_e$ identity matrix. As mentioned, the LMS adaptive algorithm is

$$e(n) = x(n-D) - \mathbf{f}^T(n)\mathbf{r}(n), \quad (3.27)$$

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu e(n)\mathbf{r}(n). \quad (3.28)$$

The decision is then $\hat{x}(n-D) = \text{sign}(y(n))$, where

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad (3.29)$$

Now, we use the MBER criterion to determine \mathbf{f} . The BER for a detected symbol can be found as

$$\begin{aligned} P(\hat{x}(n-D) \neq x(n-D)) &= P(x(n-D)y(n) < 0) \\ &= P(x(n-D)\mathbf{f}^T(n)\mathbf{C}\mathbf{x}(n) + x(n-D)\mathbf{f}^T\mathbf{v}(n) < 0) \\ &= E[P(x(n-D)\mathbf{f}^T\mathbf{C}\mathbf{x}(n) + x(n-D)\mathbf{f}^T\mathbf{v}(n) < 0 | \mathbf{x}(n))] \\ &= E\left[Q\left(\frac{\mathbf{f}^T\mathbf{C}\mathbf{x}(n)x(n-D)}{\|\mathbf{f}\|\sigma_v}\right)\right] \\ &= \frac{1}{N_a} \sum_{i=1}^{N_a} Q\left(\frac{\mathbf{f}^T\mathbf{s}_i}{\|\mathbf{f}\|\sigma_v}\right), \end{aligned} \quad (3.30)$$

where the expectations are over the N_a equally likely $\mathbf{x}(n)$ binary vectors, $\|\mathbf{f}\|^2 = \mathbf{f}^T\mathbf{f}$, $Q(\cdot)$ is the Gaussian error function, \mathbf{s}_i is the signal state, and $\mathbf{v}(n) = [v(n), v(n-1), \dots, v(n-L_e+1)]^T$. Thus, the cost function for the MBER criterion can be written as

$$J(n) = E\left[Q\left(\frac{\mathbf{f}^T\mathbf{C}\mathbf{x}(n)x(n-D)}{\|\mathbf{f}\|\sigma_v}\right)\right]. \quad (3.31)$$

The solution for the optimal MBER equalizer is given by

$$\mathbf{f}_{mber} = \arg \min_{\mathbf{f}} E\left[Q\left(\frac{\mathbf{f}^T\mathbf{C}\mathbf{x}(n)x(n-D)}{\|\mathbf{f}\|\sigma_v}\right)\right]. \quad (3.32)$$

As shown in [54], the adaptive algorithm for the MBER equalizer is

$$e(n) = x(n-D) - \mathbf{f}^T(n)\mathbf{r}(n),$$

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu 1(n)\text{sign}(e(n))\mathbf{r}(n), \quad (3.33)$$

where $1(n)$ is an error indicator function that is zero or one, depending on whether a decision error occurs at time n or not, i.e.,

$$1(n) = \begin{cases} 0, & \text{if } \text{sign}(\mathbf{f}^T \mathbf{r}(n)) = x(n-D) \\ 1, & \text{if } \text{sign}(\mathbf{f}^T \mathbf{r}(n)) \neq x(n-D) \end{cases}. \quad (3.34)$$

Next, we consider an architecture shown in Fig. 3.2. In this architecture, the linear equalizer is followed by a hyperbolic tangent function. In the function, ε is the parameter controlling the degree of nonlinearity. We call this structure as a linear perceptron equal-

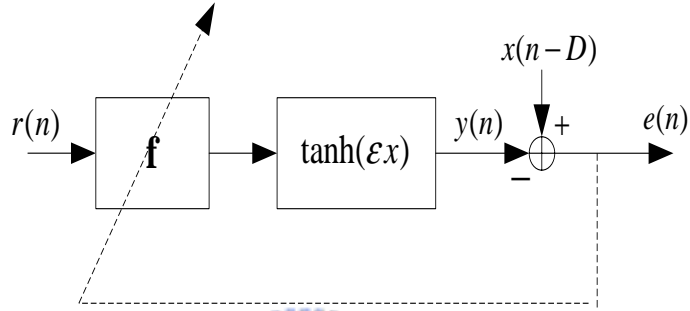


Figure 3.2: A typical linear equalizer with “tanh” function.

izer (LPE). We still use the MMSE criterion to identify the equalizer. The cost function is then

$$J(n) = \frac{1}{4} E[e^2(n)], \quad (3.35)$$

where

$$e(n) = x(n-D) - y(n), \quad (3.36)$$

and

$$y(n) = \tanh(\varepsilon \mathbf{f}^T \mathbf{r}(n)). \quad (3.37)$$

Note $1/4$ is just a scalar for cost function normalization. It is straightforward to derive the LMS-type of adaptive algorithm as

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu(1 - y^2(n))e(n)\mathbf{r}(n). \quad (3.38)$$

It is interesting to examine the cost with larger ε . If the decision is correct, i.e., $x(n-D) = 1$ and $\mathbf{f}^T \mathbf{r}(n) \geq 0$, or $x(n-D) = -1$ and $\mathbf{f}^T \mathbf{r}(n) < 0$, the cost function $J(n)$ will approach

to zero. On the other hand, if the decision is wrong, i.e. $x(n - D) = 1$ and $\mathbf{f}^T \mathbf{r}(n) < 0$, or $x(n - D) = -1$ and $\mathbf{f}^T \mathbf{r}(n) \geq 0$, the cost $J(n)$ approaches to one. We can then say that $J(n)$ is a “zero-one” like cost function. This cost function has a significant implication as shown below.

Now, we summarize three algorithms as follows:

$$\mathbf{f}_{mmse}(n + 1) = \mathbf{f}_{mmse}(n) + \mu e(n) \mathbf{r}(n), \quad (3.39)$$

$$\mathbf{f}_{mber}(n + 1) = \mathbf{f}_{mber}(n) + \mu 1(n) \text{sign}(e(n)) \mathbf{r}(n), \quad (3.40)$$

$$\mathbf{f}_{tanh}(n + 1) = \mathbf{f}_{tanh}(n) + \mu (1 - y^2(n)) e(n) \mathbf{r}(n).. \quad (3.41)$$

Comparing (3.40) with (3.41), we can find that $(1 - y^2(n))e(n)$ will approach to $1(n)\text{sign}(e(n))$ as $\varepsilon \rightarrow \infty$. We can then conclude that the cost function with zero-one characteristic will give the MBER solution. We will have more discussion regarding this subject in the later chapters.

We use an example to compare these three algorithms. We consider a linear channel shown below:

$$r(n) = 1.2x(n) + 1.1x(n - 1) - 0.2x(n - 2). \quad (3.42)$$

We define SNR as the ratio of σ_s^2 and σ_v^2 , where σ_s^2 is the variance of $s(n)$ and σ_v^2 is that of $v(n)$. We set SNR = 27dB, $L_e = 3$, and $D = 2$. The step size μ in (3.39) was set as 0.01, in (3.40) was set 0.2, and in (3.41) was set 0.1 (with $\varepsilon = 10$). All these three equalizers were initialized with zero values. We show the learning curves (with the BER) in Fig. 3.3. The dashed-lines correspond to the BER bounds for the MBER and the MMSE equalizers. We can see that the MBER and LP equalizers perform similarly; both outperform the conventional MMSE equalizer. Also note that the adaptive MBER and LP equalizers converges slower.

3.3 Volterra Equalizers

3.3.1 Volterra Series

It has been shown that a nonlinear system can be modeled with the Volterra series [55]. Let $x(n)$ and $y(n)$ be the input and the output of a system, respectively. Using the

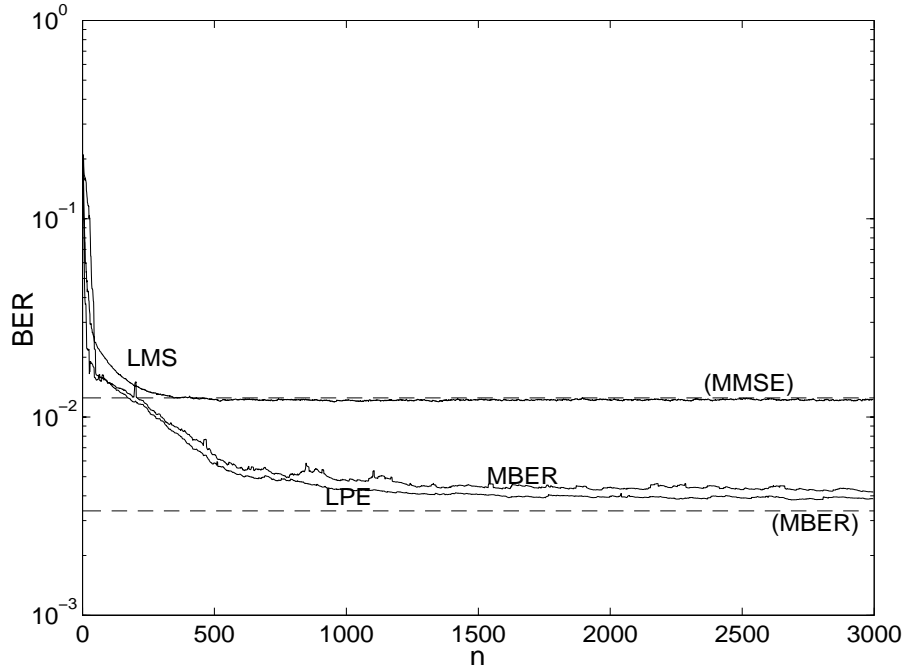


Figure 3.3: Learning curves (BER) for the MMSE, MBER, and LP equalizers.

Volterra series, we can have the following relationship:

$$y(n) = h_0 + \sum_{p=1}^{\infty} \bar{h}_p[x(n)], \quad (3.43)$$

where

$$\bar{h}_p[x(n)] = \sum_{m_1=-\infty}^{\infty} \cdots \sum_{m_p=-\infty}^{\infty} h_p(m_1, \dots, m_p) x(n - m_1) \cdots x(n - m_p), \quad (3.44)$$

The function $h_p(m_1, \dots, m_p)$ in (3.44) is called the p th-order Volterra kernel of the system. In most cases, the nonlinearity of a system has finite dimension and the memory is also finite. We can truncate the Volterra series expansion to a P th order with the memory size L . Then, (3.43) can be rewritten as

$$y(n) = h_0 + \sum_{p=1}^P \bar{h}_p[x(n)], \quad (3.45)$$

where

$$\bar{h}_p[x(n)] = \sum_{m_1=0}^L \cdots \sum_{m_p=0}^L h_p(m_1, \dots, m_p) x(n - m_1) \cdots x(n - m_p). \quad (3.46)$$

From (3.45), we can see that h_0 characterize an constant offset, $h_1(m_1)$ the linear response, and $h_p(m_1, \dots, m_p)$ the p -th order nonlinear behavior of the system. We also assume that $h_p(m_1, \dots, m_p) = 0$ for all $m_i \leq 0$ and $1 \leq i \leq p$.

3.3.2 Volterra Equalizer

In [1], [2], [12]–[15], Volterra equalizers were proposed to equalize the satellite signal. The structure of the Volterra equalizer is shown in Fig. 3.4. The output of the Volterra

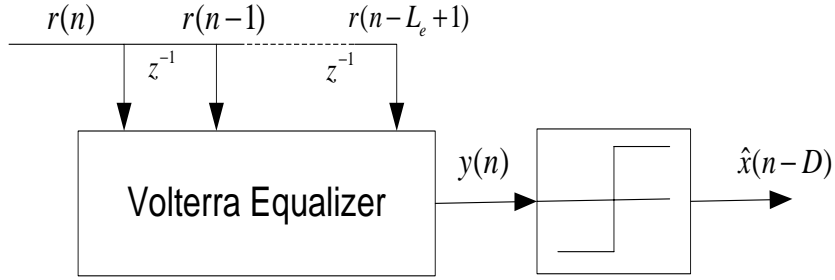


Figure 3.4: Structure of a Volterra equalizer.

equalizer, used to approximate a desired signal $x(n - D)$, can be written as

$$y(n) = h_0 + \sum_{p=1}^P \bar{h}_p[r(n)], \quad (3.47)$$

where

$$\bar{h}_p[r(n)] = \sum_{m_1=0}^{L_e-1} \cdots \sum_{m_p=0}^{L_e-1} h_p(m_1, \dots, m_p) r(n - m_1) \cdots r(n - m_p). \quad (3.48)$$

As previously defined, L_e is the length of received signal. For convenience, we use a vector representation for Volterra kernels (with finite support). Define the vector $\vec{\mathbf{r}}_1(n)$ and $\vec{\mathbf{h}}_1$ as

$$\vec{\mathbf{r}}_1(n) = [r(n), r(n-1), \dots, r(n-L_e+1)]^T, \quad (3.49)$$

and

$$\vec{\mathbf{h}}_1 = [h_1(0), h_1(1), \dots, h_1(L_e-1)]^T, \quad (3.50)$$

respectively. Then, the output of the first kernel is given by

$$\bar{h}_1[r(n)] = \vec{\mathbf{h}}_1^T \vec{\mathbf{r}}_1(n). \quad (3.51)$$

We further define a recursive representation for the input vector $\vec{\mathbf{r}}_p(n)$ of the p th order kernel as

$$\vec{\mathbf{r}}_p(n) = \vec{\mathbf{r}}_1(n) \otimes \vec{\mathbf{r}}_{p-1}(n), \quad (3.52)$$

where the symbol \otimes indicates the Kronecker product operation. The Kronecker product of an $L_1 \times M_1$ matrix \mathbf{A} with an $L_2 \times M_2$ matrix \mathbf{B} results in an $L_1 L_2 \times M_1 M_2$ matrix $\mathbf{A} \otimes \mathbf{B}$ expressed as

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{0,0}\mathbf{B} & a_{0,1}\mathbf{B} & \cdots & a_{0,M_1-1}\mathbf{B} \\ a_{1,0}\mathbf{B} & a_{1,1}\mathbf{B} & \cdots & a_{1,M_1-1}\mathbf{B} \\ \vdots & \vdots & \vdots & \vdots \\ a_{L_1-1,0}\mathbf{B} & a_{L_1-1,1}\mathbf{B} & \cdots & a_{L_1-1,M_1-1}\mathbf{B} \end{pmatrix}, \quad (3.53)$$

where $a_{i,j}$ is the ij -th entry of \mathbf{A} . The Kronecker product operation gives a well-defined ordering of the elements in $\vec{\mathbf{r}}_p(n)$. The coefficients of the kernel $h_p(m_1, m_2, \dots, m_p)$'s are arranged into a vector $\vec{\mathbf{h}}_p$ along with the elements of $\vec{\mathbf{r}}_p(n)$ such that they can match the received signal product $r(n - m_1)r(n - m_2) \cdots r(n - m_p)$ in $\vec{\mathbf{r}}_p(n)$. Then output of the p th kernel is given by

$$\bar{h}_p[r(n)] = \vec{\mathbf{h}}_p^T \vec{\mathbf{r}}_p(n). \quad (3.54)$$

where $\bar{h}_p[r(n)]$ is the coefficient vector for the p -th order kernel. Thus, the (3.47) can be rewritten as

$$y(n) = \vec{\mathbf{h}}^T \vec{\mathbf{r}}(n), \quad (3.55)$$

where

$$\vec{\mathbf{r}}(n) = [1, \vec{\mathbf{r}}_1^T(n), \vec{\mathbf{r}}_2^T(n), \dots, \vec{\mathbf{r}}_P^T(n)]^T, \quad (3.56)$$

and

$$\vec{\mathbf{h}} = [h_0, \vec{\mathbf{h}}_1^T, \vec{\mathbf{h}}_2^T, \dots, \vec{\mathbf{h}}_P^T]^T. \quad (3.57)$$

3.3.3 MMSE Coefficient Identification

The next step is to estimate the coefficients in $\vec{\mathbf{h}}$. As previous scenarios, we use the MMSE criterion given by

$$E[e^2(n)] = E[(x(n - D) - \vec{\mathbf{h}}^T \vec{\mathbf{r}}(n))^2]. \quad (3.58)$$

The Wiener solution can be obtained as previously [17].

$$\vec{\mathbf{h}}_{mmse} = \mathbf{R}_{\vec{\mathbf{r}}(n)\vec{\mathbf{r}}(n)}^{-1} \mathbf{P}_{x(n-d)\vec{\mathbf{r}}(n)}, \quad (3.59)$$

where

$$\mathbf{R}_{\vec{\mathbf{r}}(n)\vec{\mathbf{r}}(n)} = E[\vec{\mathbf{r}}(n)\vec{\mathbf{r}}^T(n)], \quad (3.60)$$

is the correlation matrix of the received signal vector $\vec{\mathbf{r}}(n)$ and

$$\mathbf{p}_{x(n-D)\vec{\mathbf{r}}(n)} = E[x(n-D)\vec{\mathbf{r}}(n)], \quad (3.61)$$

is the cross-correlation vector of the desired signal $x(n-D)$ and the received signal vector $\vec{\mathbf{r}}(n)$. Here, we implicitly assume that the correlation matrix is invertible and $x(n-D)$ and $r(n)$ are jointly stationary.

Substituting (3.59) into (3.58), we can obtain the MMSE value as

$$\begin{aligned} \xi_{mmse} &= E[(x(n-D) - \vec{\mathbf{h}}_{mmse}^T \vec{\mathbf{r}}(n))^2] \\ &= E[x^2(n-D)] - 2\vec{\mathbf{h}}_{mmse}^T \mathbf{p}_{x(n-D)\vec{\mathbf{r}}(n)} + \vec{\mathbf{h}}_{mmse}^T \mathbf{R}_{\vec{\mathbf{r}}(n)\vec{\mathbf{r}}(n)} \vec{\mathbf{h}}_{mmse} \\ &= E[x^2(n-D)] - \vec{\mathbf{h}}_{mmse}^T \mathbf{p}_{x(n-D)\vec{\mathbf{r}}(n)}. \end{aligned} \quad (3.62)$$

The Wiener solution in (3.59) is very similar to those in linear scenarios. However, the Volterra equalizer requires a large number of coefficients. Consequently, the computational complexity for the Wiener solution can be huge. Another difficulty is that the correlation matrix contains higher-order statistics of the received signals. It is well known that higher-order statistics are noise sensitive. We then need a large amount of data to obtain reliable estimates.

3.3.4 Adaptive Volterra Equalizer

The computational complexity problem of the Wiener solution can be overcome by using adaptive processing. This approach is also similar to that in linear equalizers. Fig. 3.5 depicts the structure of the adaptive Volterra equalizer.

Extending the LMS adaptive algorithm in linear equalizers, we can easily obtain the adaptive algorithm for the Volterra equalizer. The adaptive algorithm is summarized as follows:

$$e(n) = x(n-D) - \vec{\mathbf{h}}^T(n)\vec{\mathbf{r}}(n), \quad (3.63)$$

$$\vec{\mathbf{h}}(n+1) = \vec{\mathbf{h}}(n) + \mu e(n)\vec{\mathbf{r}}(n), \quad (3.64)$$

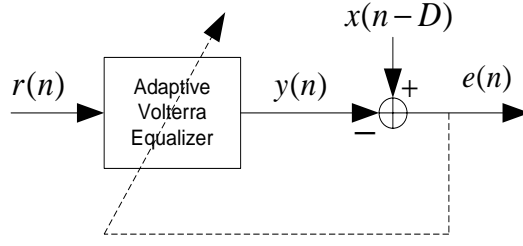


Figure 3.5: The adaptive Volterra equalizer.

where μ is the step size and $\vec{\mathbf{h}}(n)$ is the coefficients vector at time n . Although the adaptive algorithm for the Volterra equalizer is similar to that for the linear equalizer, its behavior is different. It has been shown that [9] the convergence rate of the LMS algorithm is inversely proportional to the eigenvalue spread of the input correlation matrix. It was found that the eigenvalue spread in the Volterra equalizer is larger than that in linear equalizers. Even when the input signal is white, the nonlinear entries in the input vector causes the eigenvalue spread in the Volterra equalizer to be more than one. This implies that the convergence rate for the Volterra equalizer will be slower.

Two works related the Volterra equalizers worth mentioning [16], [18]. In these works, a Volterra-perceptron structure, shown in Fig. 3.6, was proposed. The difference between the Volterra and Volterra-perceptron equalizers (VPE) is the presence of the hyperbolic tangent function $\xi(\cdot)$. From Fig. 3.6, the VPE can be expressed as

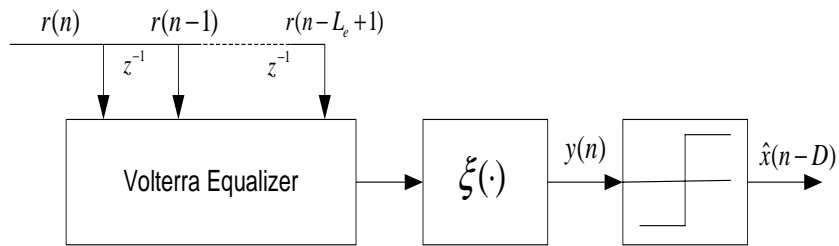


Figure 3.6: The structure of VPE.

$$y(n) = \xi(\vec{\mathbf{h}}^T(n)\vec{\mathbf{r}}(n)), \quad (3.65)$$

where

$$\xi(x) = \tanh(\varepsilon x/2), \quad \varepsilon > 0. \quad (3.66)$$

The adaptive algorithm can be derived by the SGD shown below

$$y(n) = \xi(\vec{\mathbf{h}}^T(n)\vec{\mathbf{r}}(n)), \quad (3.67)$$

$$e(n) = x(n - D) - y(n), \quad (3.68)$$

$$\vec{\mathbf{h}}(n + 1) = \vec{\mathbf{h}}(n) + \mu e(n)(1 - y^2(n))\vec{\mathbf{r}}(n). \quad (3.69)$$

As discussed in the previous section, if a cost function has the “zero-one” characteristics, the equalizer obtained by minimizing the function will achieve the MBER performance. It is apparent that the $\tanh(\cdot)$ function can change the MSE cost function to have this property. We then conclude that the adaptive VPE, which can achieve the MBER, will outperform the original Volterra equalizer (with MMSE criterion).

3.3.5 Recursive and Decision Feedback Volterra Equalizers

An inherent problem associated with the Volterra equalizer is that the number of coefficients required is usually large. In this section, we discuss the recursive Volterra equalizers. Using this structure, the number of equalizer coefficients can be significantly reduced. A general recursive Volterra equalizer is shown in Fig. 3.7. The output of the Volterra equal-

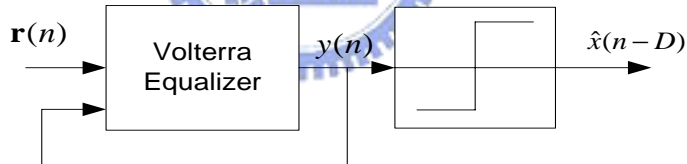


Figure 3.7: The structure of the recursive Volterra equalizer.

izer can be expressed as

$$y(n) = \sum_{i=0}^P \kappa_i(y(n-1), y(n-2), \dots, y(n-M), r(n), r(n-1), \dots, r(n-L_e+1)). \quad (3.70)$$

where $\kappa_i(y(n-1), y(n-2), \dots, y(n-M), r(n), r(n-1), \dots, r(n-L_e+1))$ is an i th-order polynomial with variables inside the parentheses. As it is known, the linear IIR filter can model linear systems more efficiently than the linear FIR filter. Similarly, the recursive Volterra filter can model nonlinear systems more efficiently than the FIR Volterra filter.

If decisions are correct, it will be better to use decisions as feedback since they do not have noise. Here, we discuss a special type of decision feedback Volterra equalizer called

the bilinear DFE [19], [20], [56]. The bilinear DFE has a low computational complexity and good performance. Its input-output relationship is given by

$$y(n) = \sum_{i=0}^{L_e-1} f_i r(n-i) + \sum_{j=1}^{L_b-1} b_j \hat{x}(n-D-j) + \sum_{i=0}^{L_e-1} \sum_{j=1}^{L_b-1} c_{i,j} r(n-i) \hat{x}(n-D-j). \quad (3.71)$$

where L_b denotes the number of feedback decisions, and \hat{x} the feedback signal. Although the bilinear model is relatively simple, it possesses some universal modeling properties [57]. It has been shown that under fairly mild conditions that a bilinear system with finite number of coefficients can be used to approximate any Volterra system with arbitrary precision.

Using the SGD method, we can obtain the adaptive algorithm for the bilinear DFE.

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \mu e(n) \mathbf{r}(n), \quad (3.72)$$

$$\mathbf{b}(n+1) = \mathbf{b}(n) + \mu e(n) \hat{\mathbf{x}}(n-D-1), \quad (3.73)$$

$$\mathbf{C}(n+1) = \mathbf{C}(n) + \mu e(n) \mathbf{r}(n) \hat{\mathbf{x}}^T(n-D-1). \quad (3.74)$$

where

$$e(n) = x(n-D) - y(n), \quad (3.75)$$

$$\mathbf{f}(n) = [f_0(n), f_1(n), \dots, f_{L_e-1}(n)]^T, \quad (3.76)$$

$$\mathbf{b}(n) = [b_1(n), b_2(n), \dots, b_{L_b}(n)]^T, \quad (3.77)$$

$$\mathbf{C}(n) = \begin{bmatrix} c_{0,1} & c_{0,2} & \cdots & c_{0,L_b} \\ c_{1,1} & c_{1,2} & \cdots & c_{1,L_b} \\ \vdots & \vdots & \vdots & \vdots \\ c_{L_e-1,1} & c_{L_e-1,2} & \cdots & c_{L_e-1,L_b} \end{bmatrix}. \quad (3.78)$$

Similar to the VPE, we can put a nonlinear function $\xi(\cdot)$ at the equalizer output. We call this a bilinear-perceptron decision feedback equalizer (BPDFE) [18], [21]. The structure of a BPDFE is shown in Fig. 3.8.

3.3.6 Simulations

In this subsection, we use a simulation example to compare the performance of nonlinear equalizers discussed in this section. We consider a nonlinear channel which is a linear

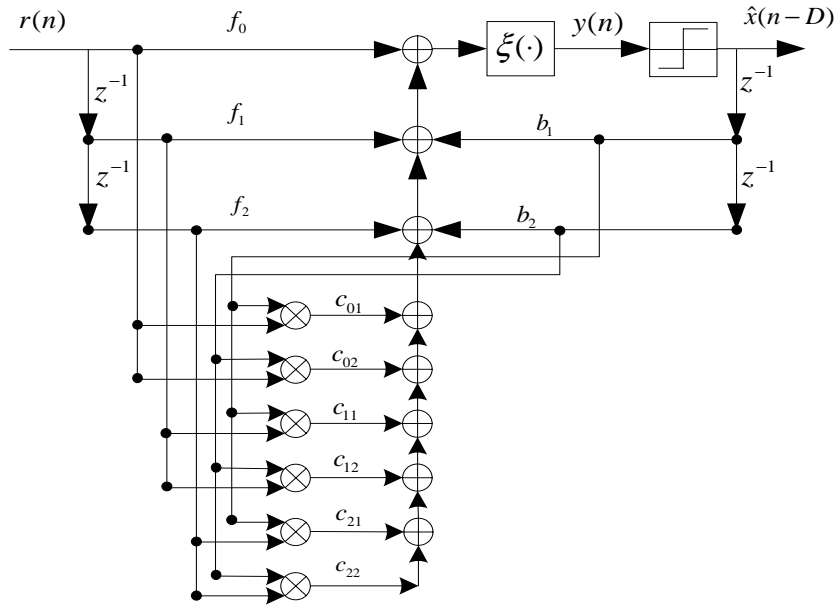


Figure 3.8: The structure of a BPDFE.

channel followed by a memoryless nonlinear function. The input-output relationship of the linear channel is given by

$$q(n) = -0.227x(n) + 0.460x(n-1) + 0.848x(n-2) + 0.460x(n-3) - 0.227x(n-4). \quad (3.79)$$

The output of the memoryless nonlinearity is given by

$$r(n) = q(n) + 0.156q^2(n) - 0.031q^3(n) + v(n). \quad (3.80)$$

The BER performance comparison for the linear, Volterra, Bayesian, and MLSE equalizers is shown in Fig. 3.9. For this set of simulations, we do not use decision feedback. For the MLSE and Bayesian equalizers, we assume that the channel information is known. Also, the decision delay for the MLSE equalizer was set as $D = 19$. For all symbol-by-symbol equalizers, we let $L_e = 5$ and $D = 5$. Thus, the number of signal state for the Bayesian equalizer is $N_a = 512$. The step size for the adaptive linear equalizer was set 0.005 and the training data size was 10^5 . The simulation condition for the LPE was the same as that for the linear equalizer ($\varepsilon = 10$). The order of the Volterra equalizer was set to 3. The total number of parameters in the Volterra equalizer was then 55. We used 4×10^6 number of training data for the adaptive Volterra equalizer. Three step sizes,

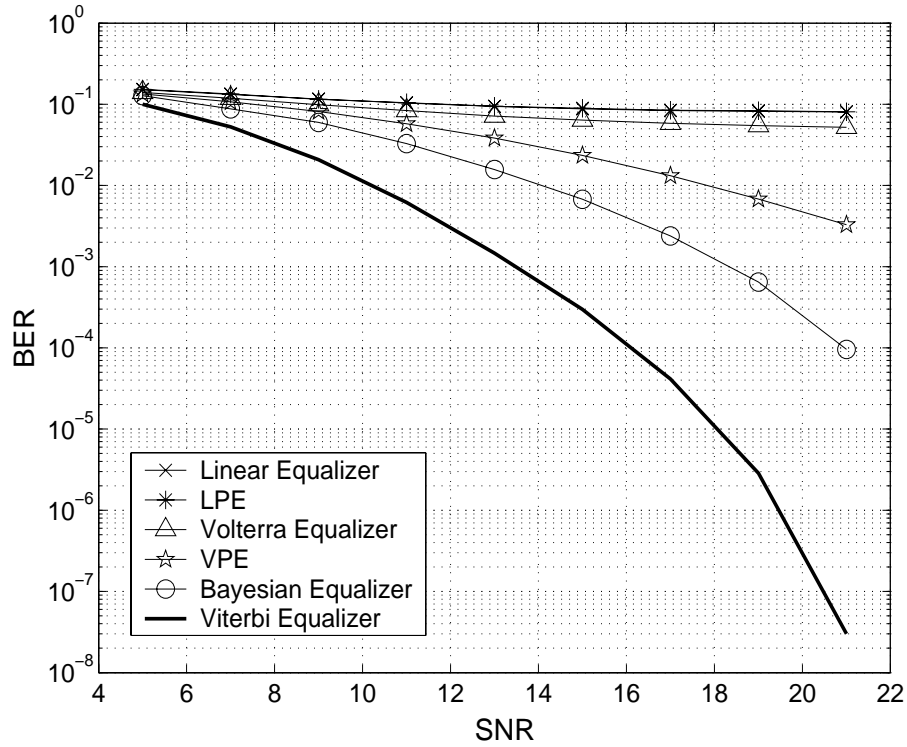


Figure 3.9: BER comparison for the linear, Volterra, Bayesian and MLSE equalizers.

namely, $\mu_1 = 0.005$, $\mu_2 = 0.003$ and $\mu_3 = 0.001$, were used for $\vec{\mathbf{h}}_1$, $\vec{\mathbf{h}}_2$ and $\vec{\mathbf{h}}_3$, respectively. The simulation conditions for the VPE was the same as that for the Volterra equalizer.

As shown in Fig. 3.9, we can find that the MLSE equalizer has the best performance. This is not surprising since it uses the whole observation sequence to perform equalization. The Bayesian equalizer, as expected, is optimal among symbol-by-symbol equalizers. The performance of the linear equalizer was very poor. This is because the signal state was not linearly separable in this nonlinear channel. Note that the VPE has better performance than the Volterra equalizer. This simulation result verified the argument made previously. The MMSE criterion cannot not lead to the MBER performance.

Properly exploitation of the decision feedback signal may offer a better performance. In the following simulations, we demonstrate the performance comparison for various DFEs. Fig. 3.10 shows the results. In all the symbol-by-symbol equalizers, the length of the decision feedback $L_b = 3$, and $\mathbf{x}_b = [\hat{x}(n-6), \hat{x}(n-7), \hat{x}(n-8)]^T$. In the DFE, we set the step size as 0.00005 and the number of training data is 10^5 . In the bilinear DFE, we

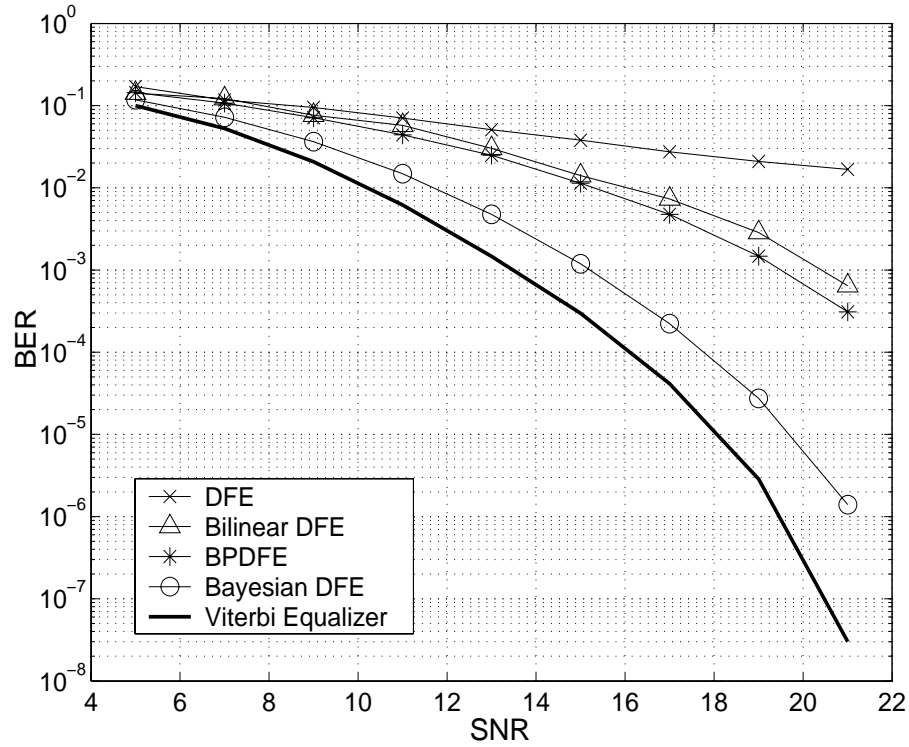


Figure 3.10: BER comparison for the DFE, bilinear DFE, BPDFE, Bayesian DFE and Viterbi equalizers.

set different step sizes for different type of coefficients, $\mu = 0.00005$ for the forward part $\mathbf{f}(n)$, $\mu = 0.00003$ for the feedback part $\mathbf{b}(n)$, and $\mu = 0.00003$ for the bilinear part $\mathbf{C}(n)$ in (3.72)–(3.74). The total number of parameters in bilinear DFE is 23 and the number of training data is 5×10^5 . For the BPDFE, we let $\varepsilon = 10$ and other simulation setup is the same as that for the bilinear DFE.

From the simulation results, we can readily see the advantage of decision feedback. The gap between the Bayesian DFE and MLSE equalizes is smaller than the gap between the feedforward Bayesian equalizer and the MLSE equalizer. The commonly used DFE did not perform well in this nonlinear channel. The performance of the bilinear DFE was better than the Volterra equalizer even though the number of coefficients is smaller. Comparing the bilinear DFE with BPDFE, we can verify again that the zero-one like cost function is better than MMSE cost function.

3.4 Neural Equalizers

The neural network is also a powerful mathematical tool modeling nonlinear systems. Many researchers have applied neural networks to the equalization problem [22]–[27]. There are many structures in neural networks. Here, we describe two structures that have been applied in equalization. They are MLP and RBF networks.

3.4.1 Multi-Layer Perceptron Equalizer

The MLP network organizes simple identical processing elements into layers and the input-output relationship of a processing element is governed by a nonlinear function. Each input of the processing element is a linear combination of the outputs from the previous layer. The MLP network has the following good properties: massive parallelism, high computation rates, great capability for nonlinear problems, continuous adaptation, inherent fault tolerance, and ease for VLSI implementation. Taking these advantages, some have applied the MLP in the equalization problem [22]–[25].

An example of the three layer MLP (2,3,3,2) is shown in Fig. 3.11.

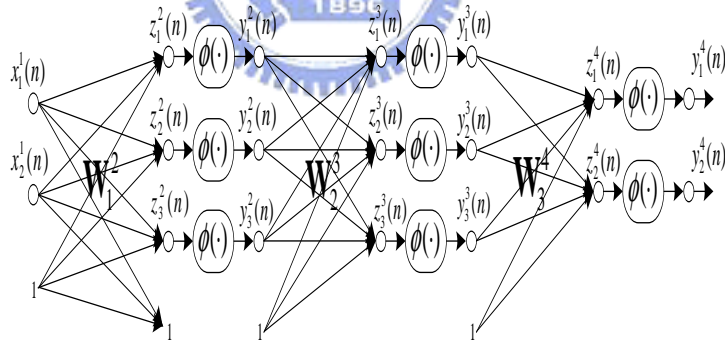


Figure 3.11: A three layers MLP (2,3,3,2).

The following notations are used in Fig. 3.11 :

$N_i + 1$: the number of inputs at each layer, $1 \leq i \leq N + 1$, where N denotes the number of the MLP layer.

$\mathbf{z}^j(n)$: the output vector of the $(j - 1)$ -th layer before the activation function, $\mathbf{z}^j(n) = [z_1^j(n), z_2^j(n), \dots, z_{N_j}^j(n)]^T$, $2 \leq j \leq N + 1$.

$\mathbf{y}^j(n)$: the output vector of the $(j - 1)$ -th layer after activation function at j th layer, where $\mathbf{y}^j(n) = [y_1^j(n), y_2^j(n), \dots, y_{N_j}^j(n)]^T$.

$\mathbf{x}^k(n)$: the input vector of the k -th layer, $\mathbf{x}^1(n) = [x_1^1(n), x_2^1(n), \dots, x_{N_1}^1(n), 1]^T$, for $k = 1$ and $\mathbf{x}^k(n) = [y_1^k(n), y_2^k(n), \dots, y_{N_k}^k(n), 1]^T$, for $1 < k \leq N$.

$\phi(\cdot)$: the activation function at j th layer.

$\phi(\mathbf{z}^j(n))$: the activation functions at j th layer, where $\phi(\mathbf{z}^j(n)) = [\phi(z_1^j(n)), \phi(z_2^j(n)), \dots, \phi(z_{N_j}^j(n))]^T$.

Using the notations above, we have following relationships.

$$\mathbf{z}^{i+1}(n) = \mathbf{W}_i^{i+1} \mathbf{x}^i(n), \quad (3.81)$$

$$\mathbf{y}^{i+1}(n) = \phi(\mathbf{z}^{i+1}(n)), \quad (3.82)$$

where \mathbf{W}_i^{i+1} is the weight matrix between i th and $(i + 1)$ th layer and $1 \leq i \leq N$.

Fig. 3.12 depicts the architecture of an MLP equalizer. To estimate the parameters of

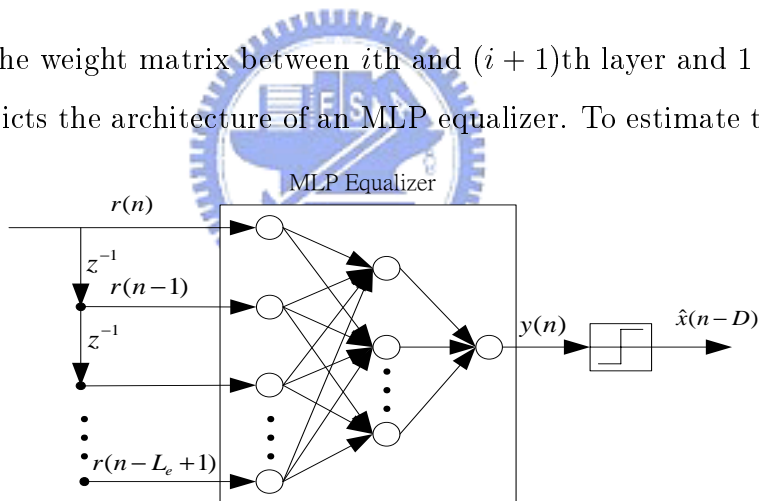


Figure 3.12: The architecture of an MLP equalizer.

an MLP equalizer, an MMSE cost function is usually used.

$$E[e^2(n)] = E[(x(n - D) - y(n))^2]. \quad (3.83)$$

Since the MLP is a highly nonlinear system, a closed-form solution minimizing (3.83) is difficult to attain. An iterative method, which is similar to the SGD, was developed to solve the problem [58]. This is referred to as the back propagation (BP) algorithm. Theoretically, the MLP network can approximate the Bayesian function in (2.21). However,

it is not an easy work in practice. This is because it lacks of a systematic approach for the architecture selection. One other thing is that the MMSE criterion cannot give the MBER performance.

We now report some simulation results to evaluate the performance of the MLP equalizer. We consider a nonlinear channel, a cascade of a linear channel and a memoryless nonlinear function [28]. The input-output relationship of the linear channel is given by

$$q(n) = 0.3482x(n) + 0.8407x(n-1) + 0.3484x(n-2). \quad (3.84)$$

The memoryless nonlinearity is given by

$$r(n) = q(n) + 0.2q^2(n) - 0.1q^3(n) + v(n). \quad (3.85)$$

For this case, $L_c = 3$. We set $L_e = 4$ and $D = 1$ for the equalizers compared. Fig. 3.13

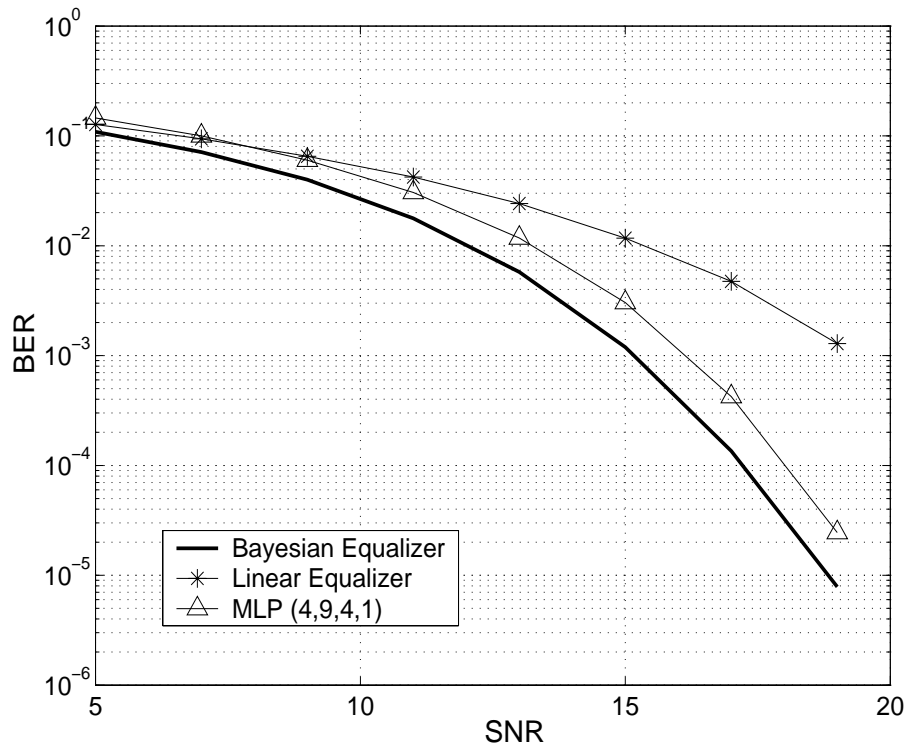


Figure 3.13: BER comparison for the linear equalizer, the MLP equalizer (4,9,4,1), and the Bayesian equalizer.

shows the performance comparison for the MLP (4,9,4,1), the linear, and the Bayesian equalizers. From this figure, we can see that the MLP equalizer is better than the linear

equalizer. This is because that the optimum solution is nonlinear function in the received signal. Also note that the MLP equalizers perform similarly to the Bayesian equalizer.

3.4.2 Radial Basis Function Equalizers

The RBF network is known to have several advantages over the MLP network; this includes its easy trainability and simpler structure. To serve as an equalizer, the RBF is known to be superior to the MLP network. The reason lies in that the structure of the RBF network is close to that of the Bayesian equalizer. Many works have been reported regarding the RBF equalizer [28]–[32], [33]–[36]. The structure of the RBF equalizer is depicted in Fig. 3.14. As the figure shows, the output of the RBF network is

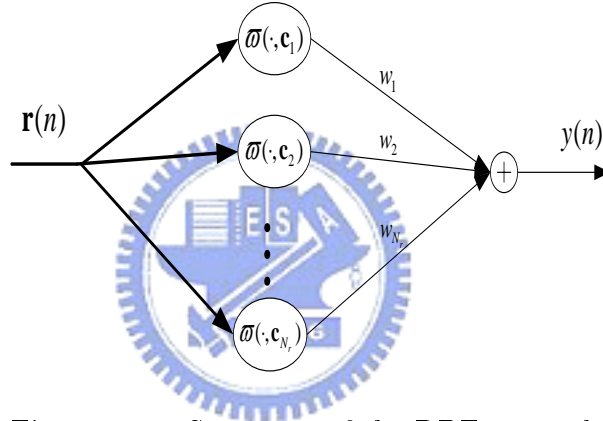


Figure 3.14: Structure of the RBF network.

$$y(n) = \sum_{i=1}^{N_r} w_i \varpi(\mathbf{r}(n), \mathbf{c}_i), \quad (3.86)$$

where $\varpi(\mathbf{x}, \mathbf{c}_i)$ is a Gaussian kernel defined as

$$\varpi(\mathbf{x}, \mathbf{c}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{\sigma_r^2}\right). \quad (3.87)$$

It is apparent that \mathbf{c}_i plays the role of the i -th kernel, $\sigma_r^2/2$ is its variance, and N_r is the number of Gaussian kernels. From (3.86), we can see that the RBF network is very similar to the nonlinear function employed in the Bayesian equalizer. The SGD [30] and clustering methods [28], [29] were proposed to identify the parameters \mathbf{c}_i and σ_r . Other methods reducing the number of Gaussian kernels N_r were also studied in [33]–[36]

3.5 Signal Space Partitioning Technique

In this section, we discuss the signal space partitioning technique proposed by Chen et al. [39]. This method can asymptotically approximate the Bayesian equalizer. Using the approach, the decision boundary can be defined by a set of hyperplanes and the received signal space can then be partitioned accordingly. If the SNR is infinite, the decision boundary will be exactly the same as that of the Bayesian equalizer.

Recall the decision rule for the Bayesian equalizer.

$$\hat{x}(n - D) = \begin{cases} +1, & f_B(\mathbf{r}(n)) > 0 \\ -1, & f_B(\mathbf{r}(n)) \leq 0 \end{cases}, \quad (3.88)$$

where

$$f_B(\mathbf{r}(n)) = \chi^{(+)}(\mathbf{r}(n)) - \chi^{(-)}(\mathbf{r}(n)), \quad (3.89)$$

and

$$\chi^{(\pm)}(\mathbf{r}(n)) = \sum_{\mathbf{s}_i \in \mathcal{S}^{\pm}} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma^2}\right). \quad (3.90)$$

The decision boundary for the Bayesian equalizer, $\mathcal{D}_B \equiv \{\mathbf{r} : f_B(\mathbf{r}) = 0\}$, is generally a hypersurface. However, if the SNR is infinite, the boundary \mathcal{D}_B is known to be piecewisely linear, and made up of a set of hyperplanes. Each of these hyperplane can be defined by a pair of *dominant states*. It can be shown that the hyperplane is orthogonal to the straight line connecting the dominant states, and also passes through the midpoint of the line. A pair of *dominant states* $(\mathbf{s}_l^{(+)}, \mathbf{s}_l^{(-)})$ is defined as

if $\forall \mathbf{s}_i \in \mathcal{S}, \mathbf{s}_i \neq \mathbf{s}_l^{(+)}, \mathbf{s}_i \neq \mathbf{s}_l^{(-)}$

$$\|\mathbf{s}_i - \mathbf{s}_{\perp}\|^2 > \|\mathbf{s}_l^{(+)} - \mathbf{s}_{\perp}\|^2, \quad (3.91)$$

where

$$\mathbf{s}_{\perp} = \frac{\mathbf{s}_l^{(+)} + \mathbf{s}_l^{(-)}}{2}. \quad (3.92)$$

The key issue in this approach is how to find all the pairs of dominant states. We summarize the method in [39] \mathcal{S} as follows:

```

Initially set  $l = 0$ 
For  $\forall \mathbf{s}_i^{(+)} \in \mathcal{S}^{(+)}$ 
  For  $\forall \mathbf{s}_j^{(-)} \in \mathcal{S}^{(-)}$ 
     $\mathbf{s}_\perp = \frac{\mathbf{s}_i^{(+)} + \mathbf{s}_j^{(-)}}{2}$ ;  $\eta = \|\mathbf{s}_i^{(+)} - \mathbf{s}_\perp\|$ 
    For  $\forall \mathbf{s}_m^{(+)} \in \mathcal{S}^{(+)}, m \neq i$ 
      For  $\forall \mathbf{s}_n^{(-)} \in \mathcal{S}^{(-)}, n \neq j$ 
        If  $(\|\mathbf{s}_m^{(+)} - \mathbf{s}_\perp\|^2 > \eta \text{ And } \|\mathbf{s}_n^{(-)} - \mathbf{s}_\perp\|^2 > \eta)$ 
           $l = l + 1$ ;
           $\mathcal{S}_{dom}^{(+)} \leftarrow \mathbf{s}_i^{(+)} = \mathbf{s}_i^{(+)}$ ;
           $\mathcal{S}_{dom}^{(-)} \leftarrow \mathbf{s}_j^{(-)} = \mathbf{s}_j^{(-)}$ ;
        End
      End
    End
  End
End
End
End
End

```

where $\mathcal{S}_{dom}^{(+)}$ and $\mathcal{S}_{dom}^{(-)}$ denote the sets of each dominant pair elements $(\mathbf{s}_i^{(+)}, \mathbf{s}_i^{(-)})$. We denote the number of pairs as N_{dom} .

Once the dominant states are found, decisions can be made based on these states. Each pair $(\mathbf{s}_i^{(+)}, \mathbf{s}_i^{(-)}) \in \mathcal{S}_{dom}$, where $\mathcal{S}_{dom} = \mathcal{S}_{dom}^{(+)} \cup \mathcal{S}_{dom}^{(-)}$, determines a hyperplane. Applying the theory of support vector machines, Chen et al. [37]–[38], [59]–[60] determined the hyperplane H_l associate with $(\mathbf{s}_i^{(+)}, \mathbf{s}_i^{(-)})$ as

$$H_l(\mathbf{r}(n)) = \mathbf{w}_l^T \mathbf{r}(n) + b_l. \quad (3.93)$$

This hyperplane is one part of the asymptotic Bayesian decision boundary. The parameters \mathbf{w}_l and b_l can be computed as

$$\mathbf{w}_l = \frac{2(\mathbf{s}_i^{(+)} - \mathbf{s}_i^{(-)})}{\|\mathbf{s}_i^{(+)} - \mathbf{s}_i^{(-)}\|^2}, \quad (3.94)$$

and

$$b_l = \frac{(\mathbf{s}_i^{(+)} - \mathbf{s}_i^{(-)})^T (\mathbf{s}_i^{(+)} + \mathbf{s}_i^{(-)})}{\|\mathbf{s}_i^{(+)} - \mathbf{s}_i^{(-)}\|^2}. \quad (3.95)$$

The hyperplane defined by (3.94) and (3.95) is a *canonical* hyperplane [59] having the property $H_l(\mathbf{s}_i^{(+)}) = 1$ and $H_l(\mathbf{s}_i^{(-)}) = -1$. Based on these hyperplanes, they then used a Boolean logic function to make decisions. The resultant detector is shown in Fig. 3.15. This technique can also be applied to the equalizer with decision feedback. Due to the feedback signal, the number of signal state will be reduced. For each given feedback signal vector pattern, we have different signal states. Thus, the dominant states could be

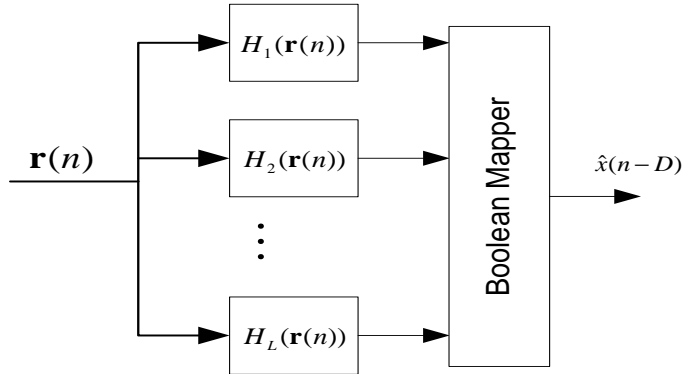


Figure 3.15: Asymptotic Bayesian equalizer using a set of hyperplanes.

searched in the new signal space \mathcal{S}_j using the same algorithm. The same decision rule can also be applied accordingly.

We report some simulation results for Chen's method. We consider a nonlinear channel which is a linear channel followed by a memoryless nonlinear function (the same as the previous section). The input-output relationship of the linear channel is given by

$$q(n) = -0.227x(n) + 0.460x(n-1) + 0.848x(n-2) + 0.460x(n-3) - 0.227x(n-4), \quad (3.96)$$

and the output of the memoryless nonlinear function is given by

$$r(n) = q(n) + 0.156q^2(n) - 0.031q^3(n) + v(n). \quad (3.97)$$

We also set $L_e = 5$ and $D = 5$. The total number of signal states in Bayesian equalizer was found to be $N_a = 512$. The performance comparison for the linear, asymptotic Bayesian, and Bayesian equalizers is shown in Fig. 3.16. From this figure, we can see that Chen's equalizer approximates the Bayesian equalizer quite well. However, we have to test 602 dominant pairs using Chen's algorithm [39]. The computational complexity is even higher than the Bayesian equalizer.

We then include the decision feedback signals into the equalizers. For this set of simulations, we set $L_e = 5$, $P = 2$, $D = 5$, $Q = 0$, and $S = 0$. The feedback decisions are $\hat{\mathbf{x}}_4(n) = [\hat{x}(n-6), \hat{x}(n-7), \hat{x}(n-8)]^T$. The performance comparison is shown in Fig. 3.17. In this scenario, Chen's algorithm has to save 592 hyperplanes. As we mentioned, the number of dominant states tested by Chen's algorithm depends on the channel. We can then conclude that this method is not efficient in non-separable channels.

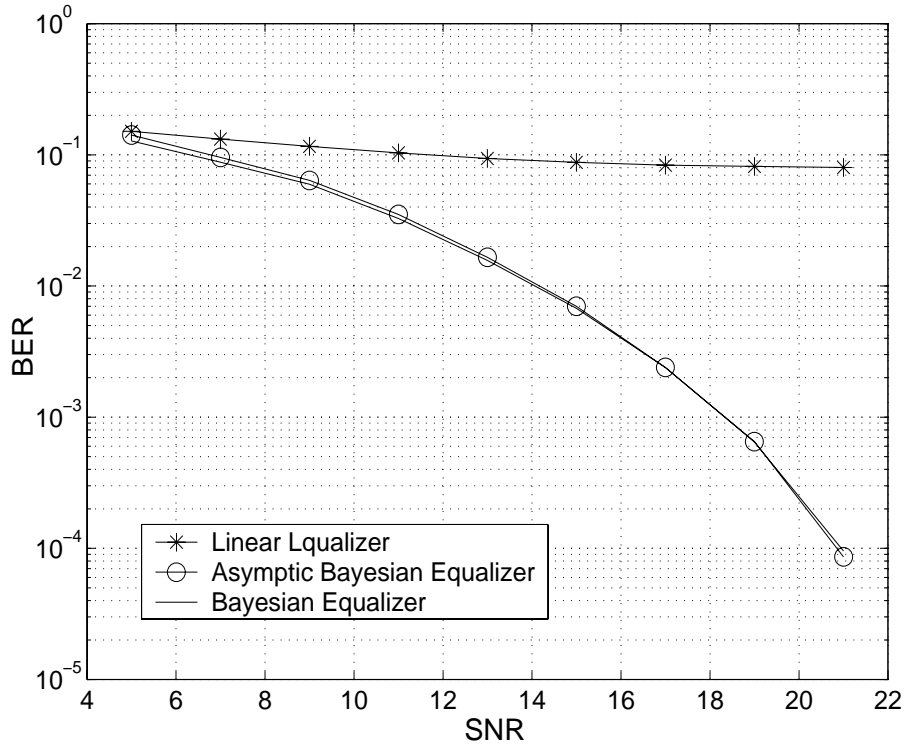


Figure 3.16: BER comparison for the linear, asymptotic Bayesian, and Bayesian equalizers.

3.6 The Discriminant Function Approach

We first briefly describe the classical pattern-recognition approach. Let $\{C_m\}$ denote a set of M -class objects and $\{\mathbf{r}_j\}$ a set of observations. Assume that $\mathbf{r}_j = \Phi(O_j)$ where O_j is an object in C_m and $\Phi(\cdot)$ is a vector mapping function. The goal of pattern classification is to derive a *classifier* that can automatically assign each \mathbf{r}_j to a object class. It has been shown that the optimal classifier chooses the class that maximizes the *a posteriori* probability function $P(C_i|\mathbf{r}_j)$, i.e.,

$$\mathbf{r}_j \in C_m \text{ if } m = \arg \max_i P(C_i|\mathbf{r}_j), \quad 1 \leq i \leq M. \quad (3.98)$$

However, the *a posteriori* probability functions may have complex forms and it may be difficult to derive and identify the functions. An alternative approach is to use a set of pre-defined functions, $f_i(\mathbf{r}_j, \Lambda_i)$, $1 \leq i \leq M$ and apply the following decision rule

$$\mathbf{r}_j \in C_m \text{ if } m = \arg \max_i f_i(\mathbf{r}_j, \Lambda_i), \quad 1 \leq i \leq M. \quad (3.99)$$

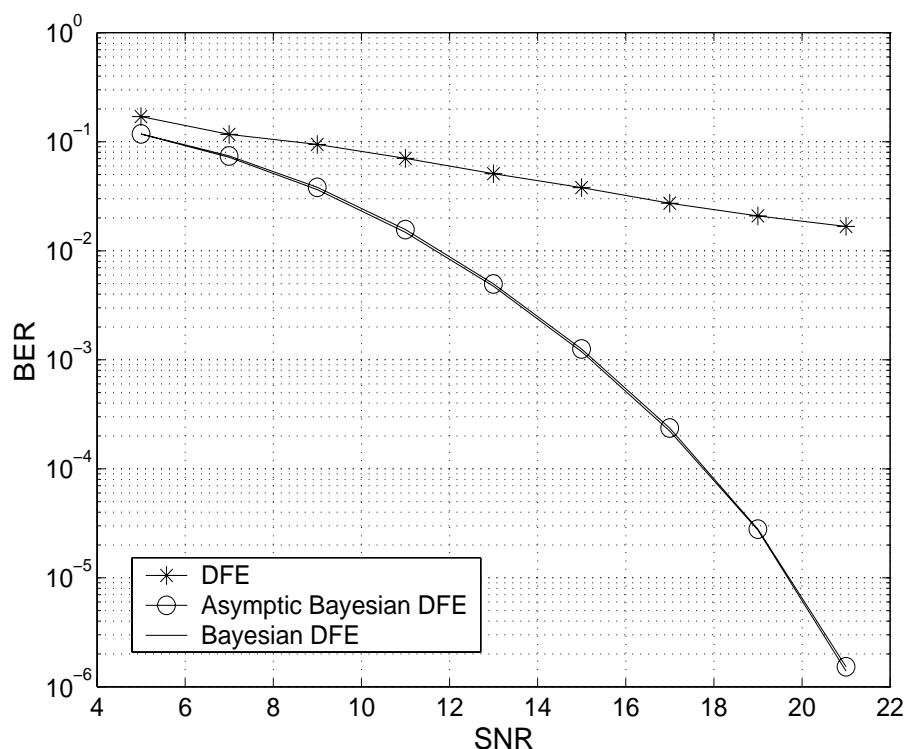


Figure 3.17: BER comparison for the conventional, asymptotic Bayesian, and Bayesian DFEs.

The function $f_i(\mathbf{r}_j, \Lambda_i)$ is called a discriminant function where Λ_i is a vector composing of its parameters. Note that $f_i(\mathbf{r}_j, \Lambda_i)$ is not required to approximate the *a posteriori* probability function directly. The main concern is whether the decision rule in (3.99) can yield the similar decision result as that in (3.98). An well-known discriminant function is the linear discriminant function defined as

$$f_i(\mathbf{r}_j, \Lambda_i) = \mathbf{w}_i^T \mathbf{r}_j + b_i, \quad (3.100)$$

where $\Lambda_i^T = [\mathbf{w}_i^T, b_i]$. Since we do not have to know the *a posteriori* probability function, a classifier with discriminant functions is considered as a nonparametric classifier. It can be easily shown that if the *a posteriori* probability function is Gaussian, the decision rule of (3.99) with the linear discriminant function (3.100) can be made exactly the same as that of (3.98). We can apply other nonlinear discriminant functions with the expense of higher computational complexity.

The central problem in pattern-recognition is how to identify Λ_i . In general, this can

be achieved by minimizing a well-defined cost function. Katagiri [43], [45] has proposed a three-step procedure to derive the cost function. The advantage of this cost function is that it can yield the minimum classification rate. Since we will use this method in the later chapter, we describe the procedure in detail here. The first step is to select the discriminant function $f_i(\mathbf{r}_j, \Lambda_i)$, which can be linear or nonlinear. The second step is then to define a misclassification measure. Katagiri suggested a measure described below [45]. If \mathbf{r}_j belongs to i th class, the misclassification measure is

$$d_i(\mathbf{x}) = -f_i(\mathbf{r}_j, \Lambda_i) + \left[\frac{1}{M-1} \sum_{k \neq i} f_k(\mathbf{r}_j, \Lambda_i)^\eta \right]^{1/\eta}, \quad (3.101)$$

where η is a positive number. The final step is to define the cost function. Katagiri also provided a cost function for this step.

$$J = E[L(d_i(\mathbf{r}_j))], \quad (3.102)$$

where

$$L(\mathbf{r}_j) = \frac{1}{1 + e^{-\varepsilon(d_i(\mathbf{r}_j) + \alpha)}}, \quad (3.103)$$

$\varepsilon > 0$, and $\alpha > 0$. Clearly, if \mathbf{r}_j belongs to i th class and the decision is correct, $f_i(\mathbf{r}_j, \Lambda_i)$ will be larger than $f_k(\mathbf{r}_j, \Lambda_i)$, $k \neq i$. Then, $d_i(\mathbf{r}_j) < 0$, a small cost is incurred in (3.102). On the other hand, if the decision is not correct, $d_i(\mathbf{r}_j) > 0$ and the cost function (3.102) will approach one. It is this “zero-one” characteristics that makes the classifier have the minimum classification rate. This can be proved as follows:

$$J = E[L(d_i(\mathbf{r}_j))] \quad (3.104)$$

$$= \sum_{k=1}^M P(C_k) \int_{\mathcal{R}} P(\mathbf{r}_j | C_k) L(d_i(\mathbf{r}_j)) d\mathbf{r}_j \quad (3.105)$$

$$\approx \sum_{k=1}^M P(C_k) \int_{\mathcal{R}} P(\mathbf{r}_j | C_k) \mathbf{1}(\mathbf{r}_j \in C_k) \mathbf{1}(k \neq \arg \min_i P(C_i | \mathbf{r}_j)) d\mathbf{r}_j \quad (3.106)$$

$$= \sum_{k=1}^M P(C_k) \int_{\mathcal{R}_k} P(\mathbf{r}_j | C_k) \mathbf{1}(\mathbf{r}_j \in C_k) d\mathbf{r}_j, \quad (3.107)$$

where \mathcal{R} is the entire observation space of \mathbf{r}_j , $\mathbf{1}(x)$ is the indicator function

$$\mathbf{1}(x) = \begin{cases} 1, & \text{if } x \text{ is true} \\ 0, & \text{otherwise} \end{cases}, \quad (3.108)$$

and $\mathcal{R}_k \in \mathcal{R}$ is an observation space where the mis-classification takes place. The approximation in (3.106) can be made arbitrarily close by varying the values of ε and α . Note that (3.107) indicate the MAP rule. Thus, we can conclude that a cost function with the zero-one characteristic can result in the minimum classification error.

If we treat the equalization problem as a classical classification problem, we can then have two discriminant functions (for BPSK signal) as shown in Fig. 3.18. We can make

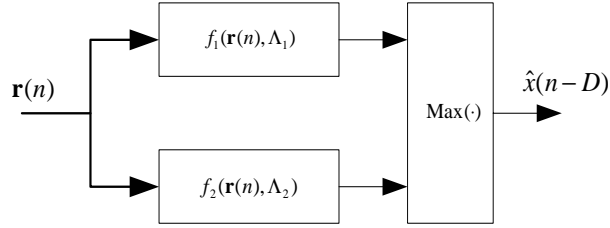
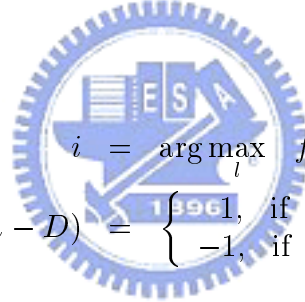


Figure 3.18: Discriminant functions for equalization.

the decision as



$$i = \arg \max_l f_l(\mathbf{r}(n), \Lambda_l), \quad (3.109)$$

$$\hat{x}(n - D) = \begin{cases} 1, & \text{if } i = 1 \\ -1, & \text{if } i = 2 \end{cases} . \quad (3.110)$$

In [43], the MLP network was chosen as the discriminant functions. Unfortunately, the MLP is highly nonlinear. It is difficult to save computations and at the same time achieve good performance.

Chapter 4

Adaptive Asymptotic Bayesian Equalization Using a Signal Space Partitioning Technique

From previous discussion, we know that the Bayesian equalizer is optimal for symbol-by-symbol equalizers; however, its computational complexity is usually very high. The signal space partitioning technique was proposed to reduce the computational complexity [39]–[41]. It was shown that the decision boundary of the equalizer consists of a set of hyperplanes. Although the resultant equalizer has low complexity, the process for searching proper hyperplanes require many efforts. Also, the number of hyperplanes consisting of the decision boundary is not known in advance and cannot be controlled either. As we showed, the number of hyperplanes required for testing may be even more than the signal states in the Bayesian equalizer. If the channel is time-varying, the problem becomes particularly troublesome. Whenever the channel response is changed, we have to re-search these hyperplanes .

In this chapter, we propose an efficient algorithm to remedy the problems mentioned. As the signal space partitioning method [39]–[41], we use a set of hyperplanes to form the decision boundary; however, the implication of these planes and the method for finding them are quite different. In the original signal space partitioning method, the number of hyperplanes is determined by the channel response. In our approach, the number of hyperplanes can be arbitrarily set. The hyperplanes found by the proposed algorithm are generally different from those found by the method in [39]–[41]. Our method allows an easy tradeoff between complexity and performance. In many cases, we can make the

performance loss small while the computational complexity reduction is large. Another feature of our approach is that the parameters of these hyperplanes can be adaptively identified using a SGD method. As a result, the proposed equalizer can be effectively applied to time-varying channels. In addition to the feedforward equalization algorithm, we have also consider efficient and fast convergent decision feedback algorithms. The computational complexity of the proposed equalizer is low and suitable for real-world implementation.

4.1 An approximate Bayesian Criterion

Although the Bayesian solution is optimal (in the MBER sense), the computational complexity is usually very high because in (2.12)–(2.14) there are N_a exponential terms to be evaluated and N_a grows exponentially with $L_c + L_e$. In this section, we propose a new method for solving this problem. Our idea is to reduce the number of terms involved in (2.14). First, we approximate (2.12)–(2.14) using the following decision rule:

$$\hat{x}(n-D) = \begin{cases} +1, & \mathbf{s}_{\max} \in \mathcal{S}^+ \\ -1, & \mathbf{s}_{\max} \in \mathcal{S}^-, \end{cases} \quad (4.1)$$

$$\begin{aligned} \mathbf{s}_{\max} &= \arg \max_{\mathbf{s}_k \in \mathcal{S}^+ \cup \mathcal{S}^-} \left\{ \exp \left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_k\|^2}{2\sigma_v^2} \right) \right\} \\ &= \arg \max_{\mathbf{s}_k \in \mathcal{S}^+ \cup \mathcal{S}^-} \left(\mathbf{s}_k^T \mathbf{r}(n) - \frac{\|\mathbf{s}_k\|^2}{2} \right). \end{aligned} \quad (4.2)$$

Because the exponential operation is a monotonic function, the second equality in (4.2) holds. Due to the rapid-decay property of the Gaussian function, the decision using (4.1)–(4.2) usually provides a good approximation to that using (2.12)–(2.14). It is straightforward to see that the smaller the noise level, the smaller the approximation error. The decision using (4.1)–(4.2) is asymptotically identical to that using (2.12)–(2.14). It is simple to show that the decision boundary of (2.12) always consists of hyperplanes regardless of noise variance. The advantage of using (4.1)–(4.2) is that we do not have to evaluate exponential functions. However, we still have N_a inner product terms to evaluate. To reduce computation, we divide \mathcal{S}^+ (as well as \mathcal{S}^-) into some subsets and merge states in each subset into a new state. If we let the number of all subsets be N_p , we then have N_p new states.

How to determine these N_p new states optimally becomes the key problem. As we show below, the optimal new states are usually not in the \mathcal{S} signal space. For this reason, we call them pseudo states. Denote the set consisting of the pseudo states as \mathcal{S}_p^\pm . We can then further approximate (4.2) as

$$\mathbf{s}_{\max} = \arg \max_{\mathbf{s}_k \in \mathcal{S}_p^+ \cup \mathcal{S}_p^-} \left(\mathbf{s}_k^T \mathbf{r}(n) - \frac{\|\mathbf{s}_k\|^2}{2} \right). \quad (4.3)$$

We now have N_p inner product terms to evaluate instead of N_a . If N_p is significantly smaller than N_a and equalizer performance is not affected, then the goal of complexity reduction has been achieved. As [39] showed, the Bayesian decision boundary consists of a set of hyperplanes if noise is absent. Each hyperplane is determined by a dominant state pair (a state in \mathcal{S}^+ and another state in \mathcal{S}^-). Only those terms associated with the dominant states need be evaluated. The number of dominant pairs, however, is determined by the channel response. Although our approach also leads to a decision boundary consisting of hyperplanes, there are some fundamental differences to the method in [39]. First, the number of pseudo states can be set arbitrarily in our approach and is not dependent on channels. We can then easily trade performance for complexity. Second, the pseudo states are found through minimization of some criterion, not searching in \mathcal{S} space. Thus, the stochastic gradient method can be applied and this results in an adaptive equalization algorithm. Note that hyperplanes found by the method in (4.3) are generally not identical to those by the method in [39]. The key problem is, as mentioned above, how to find those pseudo states. As we will show, our method is simple and effective.

4.2 Equalization Using a Single Hyperplane

We start with a simple case in which the number of pseudo states is two (i.e., one for \mathcal{S}^+ and the other for \mathcal{S}^-). From (4.3), we can see that the decision boundary is just a hyperplane. We propose using the following MMSE criterion and nonlinear function to estimate the pseudo states:

$$\min_{\mathbf{m}_1, \mathbf{m}_2} J(n) = E\{[x(n-D) - y(n)]^2\}, \quad (4.4)$$

where

$$y(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_1\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_2\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_1\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_2\|^2}{2\sigma_v^2}\right)}, \quad (4.5)$$

where \mathbf{m}_1 and \mathbf{m}_2 denote the pseudo states. The signal $y(n)$ in (4.5) corresponds to the MMSE estimate of $x(n-D)$ when the number of signal states are two and their values are known. Derivation of the nonlinear function in (4.5) is shown in the followings.

Consider an equalizer whose output is determined using the MMSE criterion described below.

$$\min J(n) = E\{(x(n-D) - y(n))^2\}. \quad (4.6)$$

The optimal solution for this cost function is known to be the conditional mean [70], which is

$$y(n) = E\{x(n-D)|\mathbf{r}(n)\}. \quad (4.7)$$

For the equalization problem mentioned above, we have

$$E\{x(n-D)|\mathbf{r}(n)\} = +1 \cdot P(x(n-D) = 1|\mathbf{r}(n)) + (-1) \cdot P(x(n-D) = -1|\mathbf{r}(n)). \quad (4.8)$$

Let the number of elements in \mathcal{S}^\pm be one. Using the Bayes rule and the fact that $P(x(n-D) = +1) = P(x(n-D) = -1) = 1/2$, we have

$$P(x(n-D) = \pm 1|\mathbf{r}(n)) = \frac{P(x(n-D) = \pm 1)p(\mathbf{r}(n)|x(n-D) = \pm 1)}{p(\mathbf{r}(n))}, \quad (4.9)$$

where $p(\mathbf{r}(n)|x(n-D) = \pm 1)$ is the *a priori* probability density function (PDF) conditioned on $x(n-D) = \pm 1$, and $p(\mathbf{r}(n))$ is the PDF of $\mathbf{r}(n)$. They are given as follows:

$$p(\mathbf{r}(n)|x(n-D) = +1) = \frac{1}{\sqrt{2\pi\sigma_v^2}} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_1\|^2}{2\sigma_v^2}\right), \quad (4.10)$$

$$p(\mathbf{r}(n)|x(n-D) = -1) = \frac{1}{\sqrt{2\pi\sigma_v^2}} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_2\|^2}{2\sigma_v^2}\right), \quad (4.11)$$

where \mathbf{s}_1 is the only state in \mathcal{S}^+ and \mathbf{s}_2 is the only state in \mathcal{S}^- . Note that

$$p(\mathbf{r}(n)) = P(x(n-D) = +1)p(\mathbf{r}(n)|x(n-D) = +1) + P(x(n-D) = -1)p(\mathbf{r}(n)|x(n-D) = -1). \quad (4.12)$$

Substituting (4.9)–(4.12), into (4.7) and (4.8) and simplifying the result, we obtain

$$y(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{s}_1\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{s}_2\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{s}_1\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{s}_2\|^2}{2\sigma_v^2}\right)}. \quad (4.13)$$

Back to the development of the proposed equalizer. Equation (4.5) corresponds to the MMSE estimate of $x(n-D)$ when the number of signal states are two and their values are known. Thus, we can say that if the number of signal states is actually two, the solution of (4.4) will give the true signal states. In general cases, the number of states in \mathcal{S}^\pm is greater than two. How do we interpret the equalization results using the pseudo state found in (4.4)–(4.5) and the decision rule in (4.3)? To answer this question, we first rewrite (4.4) as follows. If $x(n-D)$ is $+1$, the resultant signal state is in \mathcal{S}^+ and the corresponding pseudo state is \mathbf{m}_1 . On the contrary, if $x(n-D)$ is -1 , the resultant signal state is in \mathcal{S}^- and the corresponding pseudo state is \mathbf{m}_2 . Then, (4.4) can be rewritten as

$$\min_{\mathbf{m}_1, \mathbf{m}_2} J(n) = \frac{1}{4} E\{[1 - y_{i,k}(n)]^2\}, \quad (4.14)$$

where

$$y_{i,k}(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_i\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_k\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_i\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n)-\mathbf{m}_k\|^2}{2\sigma_v^2}\right)}, \quad (4.15)$$

$i = 1, k = 2$ when $x(n-D)$ is $+1$, and $i = 2, k = 1$ when $x(n-D)$ is -1 . Note that $y_{1,2}(n)$ is the same as $y(n)$ in (4.5). We introduce this notation for later development convenience. From (4.14)–(4.15), we can observe that the form of the cost function is similar for both transmitted symbols. Only the output definition is different. If the signal state for a transmitted symbol is in \mathcal{S}^+ and the received signal $\mathbf{r}(n)$ is closer to \mathbf{m}_1 than \mathbf{m}_2 , using the decision rule (4.3), we find that the decision is $+1$ and it is correct. However, if $\mathbf{r}(n)$ is closer to \mathbf{m}_2 than \mathbf{m}_1 , the decision is -1 and it is wrong. Note that if σ_v^2 is small, the nonlinear function in (4.15) approaches a step function. When the decision is correct, the cost function $J(n)$ tends to be 0. When the decision is wrong, the cost function $J(n)$ tends to be a constant 1. The result is similar for the case in which the transmitted symbol signal state is in \mathcal{S}^- . This property has a significant implication as we described before. As [45] reveals, a classifier using a discriminant function will yield a minimum classification error probability if the parameters of the function is obtained by

minimizing a cost function which gives 0 value when the decision is right and a constant when the decision is wrong. Two functions in (4.3) may be seen as two discriminant functions. Thus, we conclude that equalization results using the decision rule in (4.3) and the associated pseudo state estimates found in (4.14)–(4.15) will achieve a minimum error probability.

Since (4.4) is highly nonlinear, it is difficult to obtain the solution directly. Here, we employ the adaptive method to solve the problem. There are at least two advantages to this approach. First, the optimal solution can be found using a process called training and the computational complexity of the training algorithm is usually very low. Second, the equalizer can be continuously trained using former decisions such that it can operate in a time-varying environment. The specific method we use is called the steepest descent method [61]. Let $i, k \in \{1, 2\}$ and $i \neq k$. For equations shown below, if the signal state for a training symbol is in \mathcal{S}^+ , then $i = 1$. Otherwise, $i = 2$. Using the chain rule, we can have the gradient vectors from (4.14). Then, the update equations are given as

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_i(n)), \quad (4.16)$$

$$\mathbf{m}_k(n+1) = \mathbf{m}_k(n) - \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_k(n)), \quad (4.17)$$

where μ is the step size. As we can see, the computational complexity requirement for the adaptive algorithm is quite low. Once the pseudo states are obtained, we can have \mathbf{s}_{\max} as

$$\mathbf{s}_{\max} = \arg \max_{i \in \{1,2\}} \left(\mathbf{m}_i^T \mathbf{r}(n) - \frac{\|\mathbf{m}_i\|^2}{2} \right). \quad (4.18)$$

The decision is then

$$\hat{x}(n-D) = \begin{cases} +1, & \mathbf{s}_{\max} = \mathbf{m}_1 \\ -1, & \mathbf{s}_{\max} = \mathbf{m}_2. \end{cases} \quad (4.19)$$

From the stand point of classification, (4.18)–(4.19) divides received signal space \mathcal{R}^{L_e} into two regions \mathcal{R}_d^+ , \mathcal{R}_d^- and assigns a decision value to each region. We call \mathcal{R}_d^\pm the decision region for $x(n-D) = \pm 1$. From (4.18), we can have the decision region as

$$\mathcal{R}_d^+ \triangleq \{\mathbf{r}(n) \in \mathcal{R}^{L_e} : f_1(\mathbf{r}(n)) > f_2(\mathbf{r}(n))\}, \quad (4.20)$$

$$\mathcal{R}_d^- \triangleq \{\mathbf{r}(n) \in \mathcal{R}^{L_e} : f_1(\mathbf{r}(n)) \leq f_2(\mathbf{r}(n))\}, \quad (4.21)$$

where

$$f_1(\mathbf{r}(n)) = \mathbf{m}_1^T \mathbf{r}(n) - \frac{\|\mathbf{m}_1\|^2}{2}, \quad (4.22)$$

$$f_2(\mathbf{r}(n)) = \mathbf{m}_2^T \mathbf{r}(n) - \frac{\|\mathbf{m}_2\|^2}{2}. \quad (4.23)$$

We now use an example to describe the algorithm proposed above. As the same Example 1. in 2.3, the result was shown in Fig. 4.1. The received signal space is a plane and

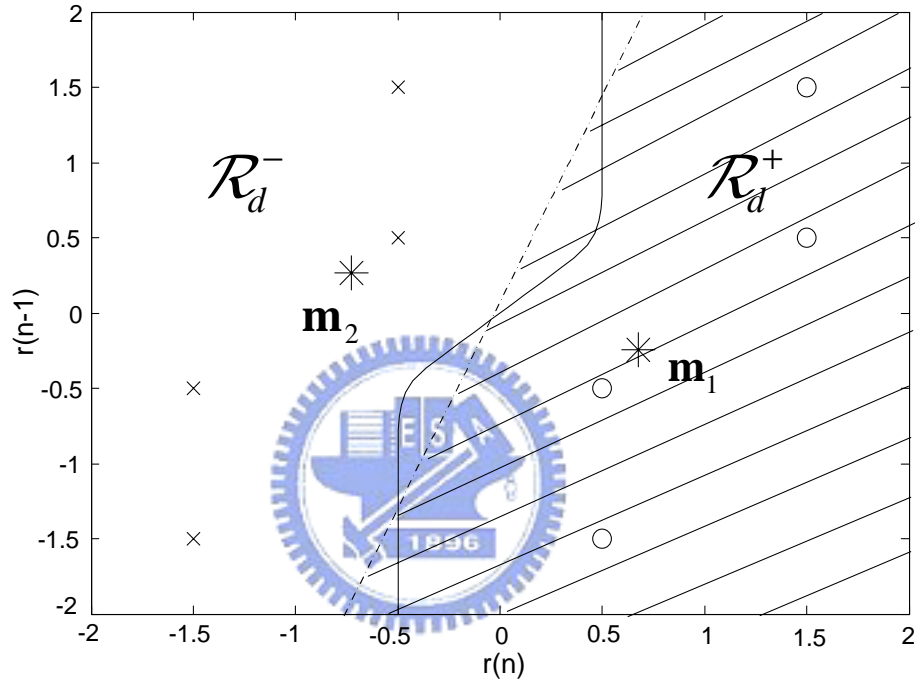


Figure 4.1: Decision boundaries for the proposed (dashed line) and the Bayesian (solid line) equalizer with two subsets.

the decision boundary is a one-dimensional curve. As we can see from the figure, the decision boundary of the proposed algorithm is linear, which was expected. It is also interesting to note that the determined pseudo state in \mathcal{S}^+ (or \mathcal{S}^-) is quite close to one of the states in \mathcal{S}^+ (or \mathcal{S}^-). From the figure, it is apparent that there is much room for performance enhancement. If we can break the decision boundary into smaller pieces and approximate each piece using a linear boundary, we can better approximate the optimal decision boundary. To implement this idea, we must then subdivide \mathcal{S}^+ (or \mathcal{S}^-) into smaller subsets. This is elaborated in the next subsection.

4.3 Equalization Using Multiple Hyperplanes

In the previous subsection, we developed an equalization algorithm using a single hyperplane. The signal state set \mathcal{S} was divided into two subsets \mathcal{S}^\pm corresponding to $x(n-D) = \pm 1$ and two pseudo states were used. In this subsection, we extend this method to accommodate the general equalization problem. The idea is to subdivide each signal state set \mathcal{S}^\pm into more subsets. Each subset is represented by a pseudo state. A hyperplane decision boundary is then determined using a pair of pseudo states. Our subdividing approach is simple and straightforward. For example, the signal state set \mathcal{S} can be divided into four subsets corresponding to $x(n-D) = \pm 1$ and $x(n-D-1) = \pm 1$, or one corresponding to $x(n-D+1) = \pm 1$ and $x(n-D) = \pm 1$. To have a general formulation, we first rewrite the input vector $\mathbf{x}_c(n)$ as a combination of three vectors.

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T, \quad (4.24)$$

where

$$\begin{aligned} \mathbf{x}_1(n) &= [x(n), x(n-1), \dots, x(n-P+1)]^T, \\ \mathbf{x}_2(n) &= [x(n-P), x(n-P-1), \dots, x(n-Q+1)]^T, \\ \mathbf{x}_3(n) &= [x(n-Q), x(n-Q-1), \dots, x(n-L_c-L_e+2)]^T, \end{aligned} \quad (4.25)$$

where $Q > D \geq P \geq 0$ and $L_c + L_e - 2 \geq Q \geq 1$. Note that $\mathbf{x}_2(n)$ includes $x(n-D)$ and its length is $Q - P$. Depending on the value of D , $\mathbf{x}_1(n)$ or $\mathbf{x}_3(n)$ may or may not exist. For example, if $D = 0$, we do not have $\mathbf{x}_1(n)$. Let $M = 2^{Q-P}$. We then have M possible vector values for $\mathbf{x}_2(n)$. Denote these vectors as \mathbf{x}_i , $i = 1, 2, \dots, M$. Now, we can divide the signal state set according to the value of $\mathbf{x}_2(n)$:

$$\mathcal{S}^i \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = \mathbf{x}_i\}, \quad 1 \leq i \leq M \quad (4.26)$$

and

$$\mathcal{S} = \bigcup_{1 \leq i \leq M} \mathcal{S}^i. \quad (4.27)$$

For example, if we let $D = 0$, $P = 0$, and $Q = 2$, then $\mathbf{x}_2(n) = [x(n), x(n-1)]^T$ and the corresponding subsets are

$$\mathcal{S}^1 \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = [+1, +1]^T\}, \quad (4.28)$$

$$\mathcal{S}^2 \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = [+1, -1]^T\}, \quad (4.29)$$

$$\mathcal{S}^3 \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = [-1, +1]^T\}, \quad (4.30)$$

$$\mathcal{S}^4 \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = [-1, -1]^T\}. \quad (4.31)$$

If $P = Q = 0$, we then have two signal subsets and this degenerates to the case discussed above. We then define M pseudo states, $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_M$ to represent the corresponding M subsets. Thus, we have M decision regions associated with these subsets. For the time being, we assume that these pseudo states are known. The decision rule using (4.3) suggests that we can assign a received signal vector $\mathbf{r}(n)$ to Subset i if the distance to \mathbf{m}_i is minimal. Once the signal subset has been determined, the $x(n-D)$ value corresponding to the subset gives the decision. Define \mathcal{I}^\pm as a set with indexes such that

$$\mathcal{I}^\pm \triangleq \{i \in \mathcal{N} : \mathbf{J}_D^T \mathbf{x}_i = \pm 1\}, \quad (4.32)$$

where \mathbf{J}_D is a $(Q-P) \times 1$ vector and

$$\mathbf{J}_D = \underbrace{[0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]^T}_{D-P+1}. \quad (4.33)$$

Also, define

$$f_i(\mathbf{r}(n)) = \mathbf{m}_i^T \mathbf{r}(n) - \frac{\|\mathbf{m}_i\|^2}{2}. \quad (4.34)$$

Thus, the output decision can be obtained as

$$i = \arg \max_l f_l(\mathbf{r}(n)), \quad (4.35)$$

$$\hat{x}(n-D) = \begin{cases} +1, & i \in \mathcal{I}^+ \\ -1, & i \in \mathcal{I}^- \end{cases}. \quad (4.36)$$

The overall structure of the proposed nonlinear equalizer is shown in Fig. 4.2. The response of each linear discriminant function in Fig. 4.2 corresponds to $\mathbf{f}_i = [\mathbf{m}_i^T, \frac{\|\mathbf{m}_i\|^2}{2}]^T$, and the input of each function is $\tilde{\mathbf{r}}(n) = [\mathbf{r}(n)^T, 1]^T$. If $\mathbf{r}(n)$ is in the decision region of Subset i , then

$$f_i(\mathbf{r}(n)) > f_j(\mathbf{r}(n)), \quad j = 1, 2, \dots, M \quad \text{and} \quad i \neq j. \quad (4.37)$$

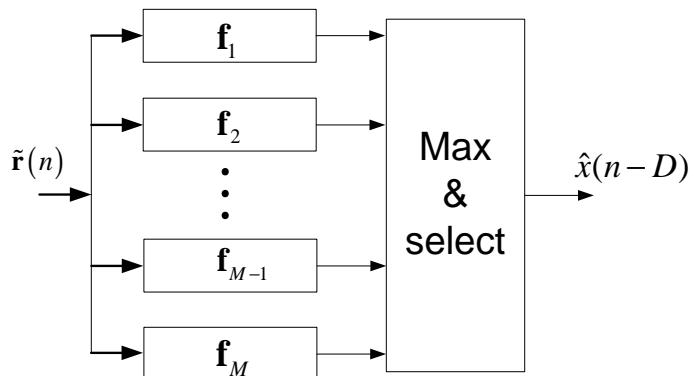


Figure 4.2: The structure of the proposed nonlinear equalizer.

Note that each inequality in (4.37) forms a hyperplane between Subset i and j . If we pretend that only these two subsets exist, this gives a decision region for Subset i . For the same Subset i , there are $M - 1$ such regions ($j = 1, 2, \dots, M, j \neq i$) and the true decision region for Subset i is the intersection of these regions. Let $\mathcal{R}_d^{i,j}$ be the decision region formed by Subset i and j (with respect to Subset i) and \mathcal{R}_d^i is the decision region for Subset i . Then,

$$\mathcal{R}_d^{i,j} \triangleq \{\mathbf{r}(n) \in \mathcal{R}^{L_e} : f_i(\mathbf{r}(n)) > f_j(\mathbf{r}(n))\} \quad (4.38)$$

and

$$\mathcal{R}_d^i \triangleq \bigcap_{j \neq i} \mathcal{R}_d^{i,j}. \quad (4.39)$$

Then the final decision region \mathcal{R}_d^\pm corresponding to $x(n - D) = \pm 1$ is

$$\mathcal{R}_d^\pm \triangleq \bigcup_{i \in \mathcal{I}^\pm} \mathcal{R}_d^i. \quad (4.40)$$

To have a better understanding of this idea, we give an example here.

Consider the same scenario given in Example 1 in 2.3. We divide signal state set \mathcal{S} into four subsets ($M = 4$) using the method described in (4.28)–(4.31). This means that $\mathbf{x}(n) = [\mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T$, where $\mathbf{x}_2(n) = [x(n), x(n - 1)]^T$ and $\mathbf{x}_3(n) = x(n - 2)$. Note that

$\mathbf{x}_1(n)$ is absent in this case. We then have the corresponding subsets

$$\mathcal{S}^1 \triangleq \{\mathbf{s}_1 = [1.5, 1.5]^T, \mathbf{s}_2 = [1.5, 0.5]^T : \mathbf{x}_2(n) = [+1, +1]^T\}, \quad (4.41)$$

$$\mathcal{S}^2 \triangleq \{\mathbf{s}_3 = [0.5, -0.5]^T, \mathbf{s}_4 = [0.5, -1.5]^T : \mathbf{x}_2 = [+1, -1]^T\}, \quad (4.42)$$

$$\mathcal{S}^3 \triangleq \{\mathbf{s}_5 = [-0.5, 1.5]^T, \mathbf{s}_6 = [-0.5, 0.5]^T : \mathbf{x}_2 = [-1, +1]^T\}, \quad (4.43)$$

$$\mathcal{S}^4 \triangleq \{\mathbf{s}_7 = [-1.5, -0.5]^T, \mathbf{s}_8 = [-1.5, -1.5]^T : \mathbf{x}_2 = [-1, -1]^T\}. \quad (4.44)$$

The proposed algorithm is then applied to perform equalization. Fig. 4.3 shows how a decision region is formed in detail. Here, we use \mathcal{R}_d^1 for detailed description. Since $M = 4$,

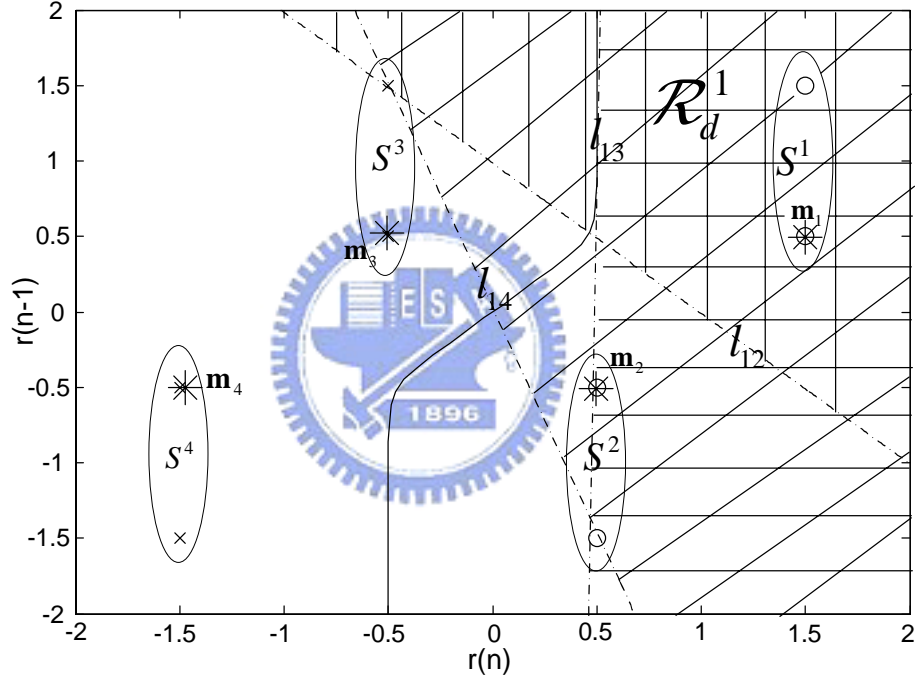


Figure 4.3: The decision region of Subset 1, \mathcal{R}_d^1 .

the region is formed by three dashed lines l_{12} , l_{13} , l_{14} , where l_{1j} is the decision boundary between \mathcal{R}_d^1 and \mathcal{R}_d^j . It is obtained by solving $f_1(\mathbf{r}(n)) = f_j(\mathbf{r}(n))$, $j = 2, 3, 4$. Symbol * in the figure represents pseudo states. The three shadow regions on the right side of l_{1j} show the region of $f_1(\mathbf{r}(n)) > f_j(\mathbf{r}(n))$. We can see that shadow region \mathcal{R}_d^1 is the intersection of all regions for $f_1(\mathbf{r}(n)) > f_j(\mathbf{r}(n))$, $j = 2, 3, 4$. Fig. 4.4 shows all of the decision regions \mathcal{R}_d^i , $j = 1, 2, 3, 4$. By definition in (4.32), $\mathcal{I}^+ = \{1, 2\}$, and $\mathcal{I}^- = \{3, 4\}$.

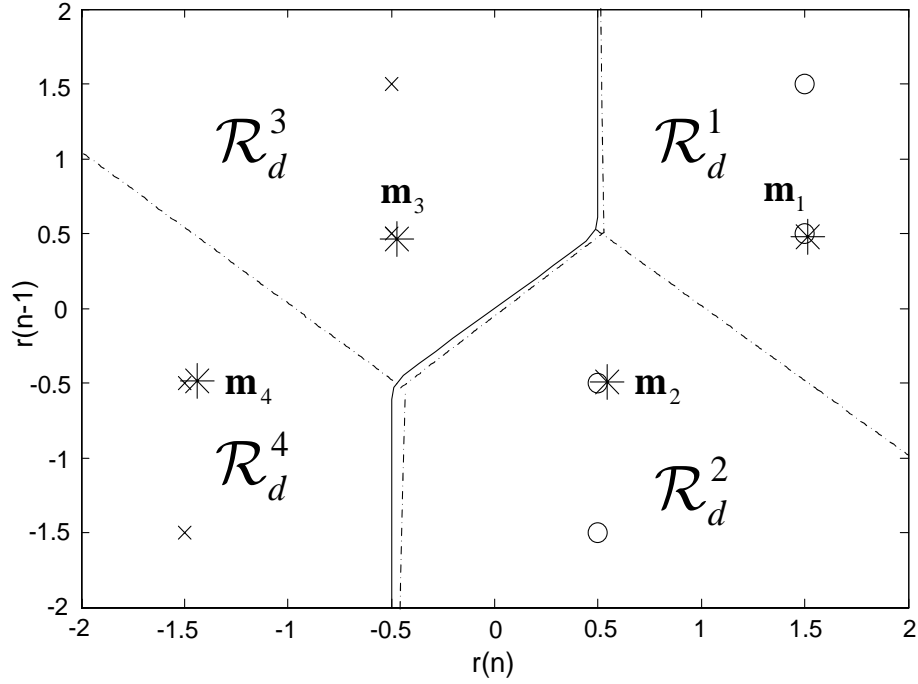


Figure 4.4: Decision regions for all subsets, \mathcal{R}_d^i , $i = 1, 2, 3, 4$, and decision boundaries of the proposed (dashed line) and the Bayesian (solid line) equalizer.

The decision \mathcal{R}_d^\pm corresponding to $x(n) = \pm 1$ are then

$$\mathcal{R}_d^+ = \mathcal{R}_d^1 \cup \mathcal{R}_d^2, \quad (4.45)$$

$$\mathcal{R}_d^- = \mathcal{R}_d^3 \cup \mathcal{R}_d^4. \quad (4.46)$$

The final decision boundary can also be seen in Fig. 4.4. It shows that the decision boundary of the proposed equalizer is very close to that of the Bayesian equalizer. However, as we show below, the computational complexity of the proposed algorithm is significantly lower.

We next describe how to find the pseudo states $\mathbf{m}_1 \ \mathbf{m}_2 \ \cdots \ \mathbf{m}_M$. We extend the method in (4.14). If the signal state for a transmitted symbol is in \mathcal{S}^i , then the cost function to be minimized is defined as

$$\min J(n) = \frac{1}{4(M-1)} \sum_{k \neq i} E\{[1 - y_{i,k}(n)]^2\}, \quad (4.47)$$

where

$$y_{i,k}(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_i\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_k\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_i\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_k\|^2}{2\sigma_v^2}\right)}. \quad (4.48)$$

Note that $J(n)$ tends to be 0 if the decision is right and a value less than 1 if the decision is wrong. Thus, as mentioned above, the decision rule associated with the estimates in (4.48) will yield a minimum error probability among all detectors using the same number of hyperplanes. To find the pseudo states, we still apply the adaptive method. For each received signal vector $\mathbf{r}(n)$, we can obtain its corresponding signal state $\mathbf{x}_c(n)$. According to $\mathbf{x}_2(n)$, we know to which subset the signal state belongs (in training mode). The stochastic gradient descent method can still be used here. We then perform pseudo state adaptation similar to (4.16)–(4.17). If a received signal vector belongs to Subset i , we then have the following adaptive algorithm ($i \neq k$).

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \mu \sum_{k \neq i} (1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_i(n)), \quad (4.49)$$

$$\mathbf{m}_k(n+1) = \mathbf{m}_k(n) - \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_k(n)), \quad (4.50)$$

$$y_{i,k}(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_i(n)\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_k(n)\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_i(n)\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_k(n)\|^2}{2\sigma_v^2}\right)}. \quad (4.51)$$

Note that the physical interpretation of the value 1 in (4.49) and (4.50) is different from that in (4.16)–(4.17). In (4.16)–(4.17), the value of 1 corresponds to the desired signal, which is $x(n - D)$, while in (4.49) and (4.50) it is just a desired constant for a right decision.

If SNR is high (σ_v^2 small), we can simplify the adaptation in (4.49). We can only update the pseudo state k ($k \neq i$) that is closest to $\mathbf{r}(n)$. Thus, the adaptive algorithm (4.49)–(4.51) becomes

$$k = \arg \min_{l \in \mathcal{I}_i} \|\mathbf{r}(n) - \mathbf{m}_l(n)\|^2 \quad (4.52)$$

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_i(n)), \quad (4.53)$$

$$\mathbf{m}_k(n+1) = \mathbf{m}_k(n) - \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_k(n)), \quad (4.54)$$

where $\mathcal{I}_i = \{1, 2, \dots, i-1, i+1, \dots, M\}$.

Since the cost function is a nonlinear function, the adaptive algorithm may converge to a local minimum. Thus, proper choice of initial value is important. We suggest using the clustering method [28] to obtain a reliable initial value. The computational complexity

requirement for this algorithm is very low. The clustering method is given as:

$$\begin{aligned}
& \text{if } \mathbf{x}_2(n) == \mathbf{x}_i \\
& \quad \mathbf{m}_i(n+1) = \text{counter}_i \times \mathbf{m}_i(n) + \mathbf{r}(n); \\
& \quad \text{counter}_i = \text{counter}_i + 1; \\
& \quad \mathbf{m}_i(n+1) = \mathbf{m}_i(n+1) / \text{counter}_i; \\
& \text{end} \tag{4.55}
\end{aligned}$$

We have thus assumed that the noise variance is known *a priori*. As a matter of fact, this assumption is not required. The smaller the noise variance, the more closely the function in (4.51) approaches a step function and the closer the equalizer is to achieving the optimal performance (minimum classification rate). However, the adaptive algorithm also converges more slowly which affects the final pseudo states positions. Thus, we can treat σ_v^2 as a design parameter. As we show in the next section, the equalization results are not sensitive to the choice of this parameter. Denote the parameter as α . According to the adaptive algorithm given above, we may summarize the overall adaptive algorithm to adjust $\mathbf{m}_1 \ \mathbf{m}_2 \ \cdots \ \mathbf{m}_M$ as follows:

1. Initially set $\mathbf{m}_1 \ \mathbf{m}_2 \ \cdots \ \mathbf{m}_M$ by the clustering method
2. For each instant of time, $n = 1, 2, \dots$,
 - if $\mathbf{x}_2(n) == \mathbf{x}_i$
 - for $l = 1 : M$
 - $\zeta_l(n) = \|\mathbf{r}(n) - \mathbf{m}_l(n)\|^2$
 - end
 - $k = \arg \min_{l \in \mathcal{I}_i} \zeta_l(n)$
 - $y_{i,k}(n) = \frac{\exp(-\zeta_i(n)/2\alpha) - \exp(-\zeta_k(n)/2\alpha)}{\exp(-\zeta_i(n)/2\alpha) + \exp(-\zeta_k(n)/2\alpha)}$
 - $\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_i(n));$
 - $\mathbf{m}_k(n+1) = \mathbf{m}_k(n) - \mu(1 - y_{i,k}(n))(1 - y_{i,k}^2(n))(\mathbf{r}(n) - \mathbf{m}_k(n));$
 - end

The proposed method described above partitions the signal space into 2^I subsets, where I is an integer. It is straightforward to see that if we proper combine the decision regions, we can have arbitrary number of discriminant functions. In Example 1, we can only consider three $\mathbf{x}_2(n)$ patterns, i.e., $\mathbf{x}_2(n) = [+1, \times]$, $\mathbf{x}_2(n) = [-1, +1]$, and $\mathbf{x}_2(n) = [-1, +1]$

where \times denotes don't-care. Then, we can have three discriminant functions.

With a minor modification, the clustering algorithm described above can find the mean of each cluster as well as the variance. Assuming that each cluster has a Gaussian distribution and treating its mean as a state, we can then apply the Bayesian decision rule. We call this a reduced-state (RS) Bayesian equalizer. Depending on the number of clusters, the computational complexity of the RS Bayesian equalizer can be much lower than that of the original Bayesian equalizer. However, as we show in the next section, the performance of the RS Bayesian equalizer is poor unless the number of the clusters is large.

4.4 Decision Feedback Equalization

The Bayesian DFE is known to be the optimal symbol-by-symbol DFE. Similar to the Bayesian feedforward equalizer, its computational complexity is high. We can extend the algorithm proposed above to reduce the computational complexity of the Bayesian DFE. Define the input data vector $\mathbf{x}(n)$ be a combination of four vectors

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n), \mathbf{x}_4^T(n)]^T \quad (4.56)$$

where

$$\mathbf{x}_1(n) = [x(n), x(n-1), \dots, x(n-P+1)]^T \quad (4.57)$$

$$\mathbf{x}_2(n) = [x(n-P), x(n-P-1), \dots, x(n-Q+1)]^T \quad (4.58)$$

$$\mathbf{x}_3(n) = [x(n-Q), x(n-Q-1), \dots, x(n-S+1)]^T \quad (4.59)$$

$$\mathbf{x}_4(n) = [x(n-S), x(n-S-1), \dots, x(n-L_c-L_e+2)]^T \quad (4.60)$$

$$(4.61)$$

where $S \geq Q > D \geq P \geq 0$ and $L_c + L_e - Q - 2 \geq S \geq Q \geq 1$. The fourth component will be obtained from decision feedback. Let $\hat{\mathbf{x}}_b(n)$ be the feedback vector and decisions be correct. Then, $\mathbf{x}_4(n) = \hat{\mathbf{x}}_b(n)$. Since the length of $\hat{\mathbf{x}}_b(n)$ is $L_b = L_c + L_e - S - 1$, there are $N_b = 2^{L_b}$ possible decision pattern. As defined, we call each pattern a decision state and its possible value is $\hat{\mathbf{x}}_j$, $1 \leq j \leq N_b$. We then define the input data vector given the

decision state j as

$$\mathbf{x}_j(n) = [\mathbf{x}_{1,j}^T(n), \mathbf{x}_{2,j}^T(n), \mathbf{x}_{3,j}^T(n), \hat{\mathbf{x}}_b^T(n) = \hat{\mathbf{x}}_j^T]^T \quad (4.62)$$

Note that $\mathbf{x}_{2,j}(n)$ includes $x(n-D)$ and its length is $Q-P$. Similar to the feedforward equalization scenario, $\mathbf{x}_{1,j}(n)$ and $\mathbf{x}_{2,j}(n)$ may or may not exist. We then have $M = 2^{P+Q+1}$ possible vector values for $\mathbf{x}_{2,j}(n)$. Denote these vector as $\mathbf{x}_{i,j}$, $1 \leq i \leq M$. Thus, we can divide the received signal space into M classes according to $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$ (for the j th decision feedback state). Thus, we can divide received signal space into M classes according to $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$. Then we use M pseudo states $\{\mathbf{m}_{1,j}, \mathbf{m}_{2,j}, \dots, \mathbf{m}_{M,j}\}$ to represent these classes.

Define \mathcal{I}_j^\pm as a set with indexes such that

$$\mathcal{I}_j^\pm \triangleq \{i \in \mathcal{N} : \mathbf{J}_D^T \mathbf{x}_{i,j} = \pm 1\}. \quad (4.63)$$

where \mathbf{J}_D is $(Q-P) \times 1$ vector and

$$\mathbf{J}_D = \underbrace{[0 \ \dots \ 0 \ 1 \ 0 \ \dots \ 0]^T}_{D-P+1}. \quad (4.64)$$

Thus, \mathcal{I}_j^\pm is the index set corresponding to the subset in \mathcal{S}_j^\pm . Also, define

$$f_{i,j}(\mathbf{r}(n)) = \mathbf{m}_{i,j}^T \mathbf{r}(n) - \frac{\|\mathbf{m}_{i,j}\|^2}{2}. \quad (4.65)$$

Then, the final decision can be obtained as

$$i = \arg \min_l f_{l,j}(\mathbf{r}(n)), \quad (4.66)$$

$$\hat{x}(n-D) = \begin{cases} 1, & i \in \mathcal{I}_j^+ \\ -1, & i \in \mathcal{I}_j^- \end{cases}. \quad (4.67)$$

The overall structure of the proposed equalizer shown in Fig. 4.5. Note that, if we do not use the feedback signal, the selection part can be ignored.

We need also to decide the unknown parameters $\mathbf{m}_{i,j}$'s. Using the cost function in (4.47), we have

$$\min J_j(n) = \frac{1}{4(M-1)} \sum_{k \neq i} E\{[1 - y_{i,k,j}(n)]^2\}, \quad (4.68)$$

where

$$y_{i,k,j}(n) = \frac{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_{i,j}\|^2}{2\sigma_v^2}\right) - \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_{k,j}\|^2}{2\sigma_v^2}\right)}{\exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_{i,j}\|^2}{2\sigma_v^2}\right) + \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{m}_{k,j}\|^2}{2\sigma_v^2}\right)}. \quad (4.69)$$

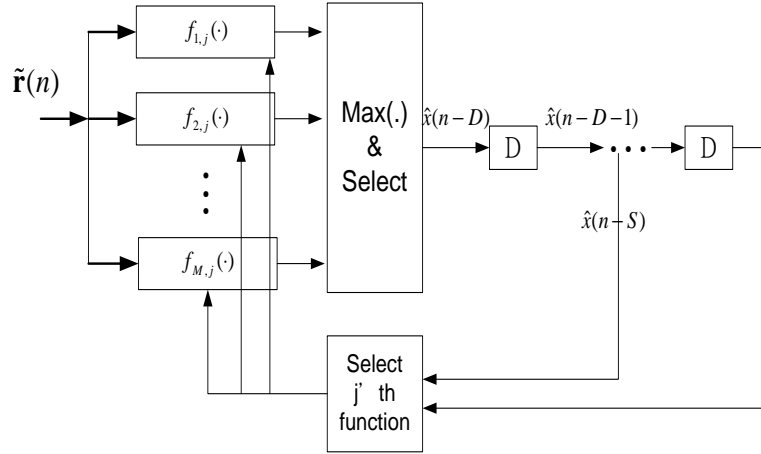


Figure 4.5: The structure of the proposed nonlinear DFE.

Using the same technique described in the previous section, we can obtain the corresponding adaptive algorithm. We summarize the overall adaptive algorithm for $\mathbf{m}_{i,j}$ identification in Table 4.1.

Since there are N_b feedback states, we then will have N_b sets of discriminant functions. Thus, we have to store all these functions and the memory size may be large. Here, we propose a method to alleviate this problem. As mentioned above, $\mathbf{s}_{i,j}$ denotes one possible signal state given $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$. More accurately, $\mathbf{s}_{i,j}$ in (2.17) is a mapping of the i th input vector $[\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T$ given $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$. Consider the linear channel scenario, i.e.,

$$\psi(x(n), \dots, x(n - L_c + 1)) = \sum_{i=0}^{L_c-1} c_i \mathbf{x}(n - i) \quad (4.70)$$

where c_i 's are the channel input responses. The signal vector $\mathbf{s}(n)$ can be written as

$$\mathbf{s}(n) = \mathbf{C}\mathbf{x}(n) \quad (4.71)$$

where \mathbf{C} is a Toeplitz matrix with the channel responses c_i 's and

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots & 0 \\ 0 & c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{L_c-1} \end{bmatrix}. \quad (4.72)$$

Decompose $\mathbf{x}(n)$ as $\mathbf{x}(n) = [\mathbf{x}_f^T(n), \hat{\mathbf{x}}_b^T(n)]^T$ where $\mathbf{x}_f(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T$. The matrix \mathbf{C} can then be decomposed accordingly, i.e., $\mathbf{C} = [\mathbf{C}_f, \mathbf{C}_b]$. Thus, (4.71) can be

Table 4.1: The adaptive algorithm of proposed DFE

-
1. Initialize variables: $n = 0$,
 $\mathbf{m}_{i,j}(0) = \mathbf{0}$ for $i = 1, \dots, M$ and $j = 1, \dots, N_b$
 2. Clustering method (find the initial value), $n = 1, 2, \dots$
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 $\mathbf{m}_{i,j}(n+1) = \text{counter}_{i,j} \times \mathbf{m}_{i,j}(n) + \mathbf{r}(n)$,
 $\text{counter}_{i,j} = \text{counter}_{i,j} + 1$,
 $\mathbf{m}_{i,j}(n+1) = \mathbf{m}_{i,j}(n+1) / \text{counter}_{i,j}$.
 end
 3. Adaptive algorithm
 if $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 for $l = 1 : M$
 $\zeta_{l,j}(n) = \|\mathbf{r}(n) - \mathbf{m}_{l,j}(n)\|^2$,
 end
 $k = \arg \min_{l \neq i} \zeta_{l,j}(n)$
 $y_{i,k,j} = \frac{\exp(-\zeta_{i,j}(n)/2\alpha) - \exp(-\zeta_{k,j}(n)/2\alpha)}{\exp(-\zeta_{i,j}(n)/2\alpha) + \exp(-\zeta_{k,j}(n)/2\alpha)}$
 $\mathbf{m}_{i,j}(n+1) = \mathbf{m}_{i,j}(n) + \mu(1 - y_{i,k,j}(n))(1 - y_{i,k,j}^2(n))(\mathbf{r}(n) - \mathbf{m}_{i,j}(n))$,
 $\mathbf{m}_{k,j}(n+1) = \mathbf{m}_{k,j}(n) - \mu(1 - y_{i,k,j}(n))(1 - y_{i,k,j}^2(n))(\mathbf{r}(n) - \mathbf{m}_{k,j}(n))$,
 end

rewritten as

$$\mathbf{s}(n) = \mathbf{C}_f \mathbf{x}_f(n) + \mathbf{C}_b \hat{\mathbf{x}}_b(n) \quad (4.73)$$

From (4.73), we can see that $\mathbf{s}_{i,j} - \mathbf{s}_{i,k} = \mathbf{C}_b(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_k)$, $j \neq k$, for $1 \leq i \leq N_d$. This is to say that signal states for two feedback states has a common difference vector. Thus, we can describe all signal states (for different given feedback states) as

$$\mathbf{s}_{i,j} = \mathbf{s}_{i,1} + \mathbf{d}_j, \quad 1 \leq j \leq N_b, \quad (4.74)$$

where \mathbf{d}_j is a difference vector and $\mathbf{d}_1 = \mathbf{0}$.

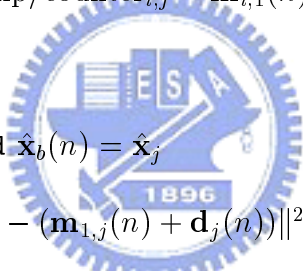
According to the symmetric property, each pair pseudo states $\mathbf{m}_{i,j}$ and $\mathbf{m}_{i,k}$ can also have the same difference. The storage of pseudo states can be reduced by this approach. We can express these pseudo states by the following vectors

$$\mathbf{m}_{i,j} = \mathbf{m}_{i,1} + \mathbf{d}_j, \quad 1 \leq i \leq M, \quad 1 \leq j \leq N_b. \quad (4.75)$$

By this approach, all the parameters needed to identify are $\mathbf{m}_{i,1}$ and \mathbf{d}_j , $1 \leq i \leq M$, $1 \leq j \leq N_b$, where $\mathbf{d}_1 = \mathbf{0}$. The number of vectors required is reduced from $M \times N_b$

vectors to $M + N_b - 1$ ($\mathbf{m}_{i,1}$ for $1 \leq i \leq M$ and \mathbf{d}_j for $2 \leq i \leq N_b$). Using the same cost function (4.68), we can derive the corresponding adaptive algorithm summarized in Table 4.2.

Table 4.2: The adaptive algorithm for the proposed reduced storage DFE

<p>1. Initialize variables: $n = 0$, $\mathbf{m}_{i,1}(0) = \mathbf{0}$ and $\mathbf{d}_j = \mathbf{0}$ for $i = 1, \dots, M$ and $j = 1, \dots, N_b$</p> <p>2. Clustering method (find the initial value), $n = 1, 2, \dots$ if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$ if $j = 1$ $\mathbf{m}_{i,1}(n+1) = \text{counter}_{i,1} \times \mathbf{m}_{i,1}(n) + \mathbf{r}(n)$, $\text{counter}_{i,1} = \text{counter}_{i,1} + 1$, $\mathbf{m}_{i,1}(n+1) = \mathbf{m}_{i,1}(n+1) / \text{counter}_{i,1}$, else $\text{tmp} = \text{counter}_{i,j}(n) \times (\mathbf{m}_{i,1}(n) + \mathbf{d}_j(n)) + \mathbf{r}(n)$, $\text{counter}_{i,j} = \text{counter}_{i,j} + 1$, $\mathbf{d}_j(n+1) = \text{tmp} / \text{counter}_{i,j} - \mathbf{m}_{i,1}(n)$. end end</p> <p>3. Adaptive algorithm if $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$ for $l = 1 : M$ $\zeta_{l,j}(n) = \ \mathbf{r}(n) - (\mathbf{m}_{1,j}(n) + \mathbf{d}_j(n))\ ^2$, end $k = \arg \min_{l \neq i} \zeta_{l,j}(n)$ $y_{i,k,j} = \frac{\exp(-\zeta_{i,j}(n)/2\alpha) - \exp(-\zeta_{k,j}(n)/2\alpha)}{\exp(-\zeta_{i,j}(n)/2\alpha) + \exp(-\zeta_{k,j}(n)/2\alpha)}$ if $j = 1$ $\mathbf{m}_{i,1}(n+1) = \mathbf{m}_{i,1}(n) + \mu(1 - y_{i,k,1}(n))(1 - y_{i,k,1}^2(n))(\mathbf{r}(n) - \mathbf{m}_{i,1}(n))$, $\mathbf{m}_{k,1}(n+1) = \mathbf{m}_{k,1}(n) + \mu(1 - y_{i,k,1}(n))(1 - y_{i,k,1}^2(n))(\mathbf{r}(n) - \mathbf{m}_{k,1}(n))$, else $\mathbf{d}_j(n+1) = \mathbf{d}_j(n) + \mu(1 - y_{i,k,j}(n))(1 - y_{i,k,j}^2(n))(\mathbf{m}_{k,1}(n) - \mathbf{m}_{i,1}(n))$ end</p>	
---	--

Although the algorithm can reduce the storage, the performance may be affected in nonlinear channels. Since in nonlinear channels, each pair signal state $\mathbf{s}_{i,j}$ and $\mathbf{s}_{i,k}$ do not have the same difference. We will show later that the performance loss is small.

4.5 Computational Complexity

There are two phases in the operation of the proposed algorithm, the training and decision phases. For simplicity, we treat the computational requirement for division the same as that for multiplication. We summarize the overall computational requirement without decision feedback in Table 4.3 and 4.4. Roughly speaking, in the decision phase, the complexity of the proposed equalizer is M times higher than the linear equalizer. The actual choice of M is dependent on the desired performance. In our design, the decision region for each subset is automatically and adaptively formed, and these decision regions approximate Bayesian regions. In principle, if the Bayesian decision boundary has a complex shape, we need more subsets. In many cases, however, only a small number of subsets is sufficient. The computational complexity of the proposed equalizer can be much lower than that of the Bayesian equalizer because M can be much smaller than N_a without significantly sacrificing performance. Note that if a time-varying channel is considered, decisions can be used to train the proposed equalizer continuously such that channel variations can be properly tracked. In this case, the computational complexity for the proposed equalizer is the summation of that listed in Table 4.3 (the adaptive part) and that in Table 4.4. Since the value of M used is usually small, the overall complexity will not be increased significantly.

Table 4.3: Computational complexity comparison for the linear and proposed equalizers in the training phase

	Proposed		Linear
	Initial	Adaptive	
Multiplications	L_e	$M + 2L_e + 5$	$2L_e + 1$
Additions	L_e	$(M + 4)L_e + 4$	$2L_e$
Others		$2 \times \exp(\cdot)$	

We also summarize the computational complexity for the proposed DFE algorithms in Table 4.5 and 4.6, respectively. Similarly, the computational complexity of the proposed equalizer can be much lower than that of the Bayesian DFE because M can be much smaller than N_d without significantly sacrificing performance.

Table 4.4: Computational complexity comparison for the linear, proposed, and Bayesian equalizers in the decision phase

	Bayesian	Proposed	Linear
Multiplications	$N_a(L_e + 1)$	ML_e	L_e
Additions	$2L_eN_a - 1$	ML_e	$L_e - 1$
Others	$N_a \times \exp(\cdot)$	Compare logics	

Table 4.5: Computational complexity comparison for the conventional and proposed DFEs in the training phase

	Proposed		DFE
	Initial	Adaptive	
Multiplications	L_e	$M + 2L_e + 5$	$2(L_e + L_b)$
Additions	L_e	$(M + 4)L_e + 4$	$2(L_e + L_b + 1)$
Others		$2 \times \exp(\cdot)$	

4.6 Simulation Results

In this section, we report some simulation results demonstrating the effectiveness of the proposed equalizers. Two linear channels and two nonlinear channels were used. We compared the proposed equalizer with an optimum Bayesian equalizer, a conventional MMSE linear equalizer, and a MBER linear equalizer [52]. The bit error rate (BER) was used as the performance measure. We first considered a linear channel [28]. The channel is described using a difference equation as:

$$r(n) = 0.3482x(n) + 0.8764x(n - 1) + 0.3482x(n - 2) + v(n). \quad (4.76)$$

Table 4.6: Computational complexity comparison for the conventional, proposed, and Bayesian DFEs in the decision phase

	Bayesian DFE	Proposed DFE	DFE
Multiplications	$N_d \times (L_e + 1)$	ML_e	L_e
Additions	$2L_eN_d - 1$	ML_e	$L_e - 1 + L_b$
Others	$N_d \times \exp(\cdot)$	Compare logics	

As we can see, the channel length is 3 ($L_c = 3$). For all equalizers compared, we let $L_e = 4$ and $D = 1$. Thus, $N_a = 2^{L_c+L_e-1} = 64$. For the proposed equalizer, we let $M = 4$ and $M = 8$, for which $\mathbf{x}_2(n) = [x(n-1), x(n-2)]^T$ and $\mathbf{x}_2(n) = [x(n-1), x(n-2), x(n-3)]^T$, respectively. The simulation results for various SNR conditions are shown in Fig. 4.6.

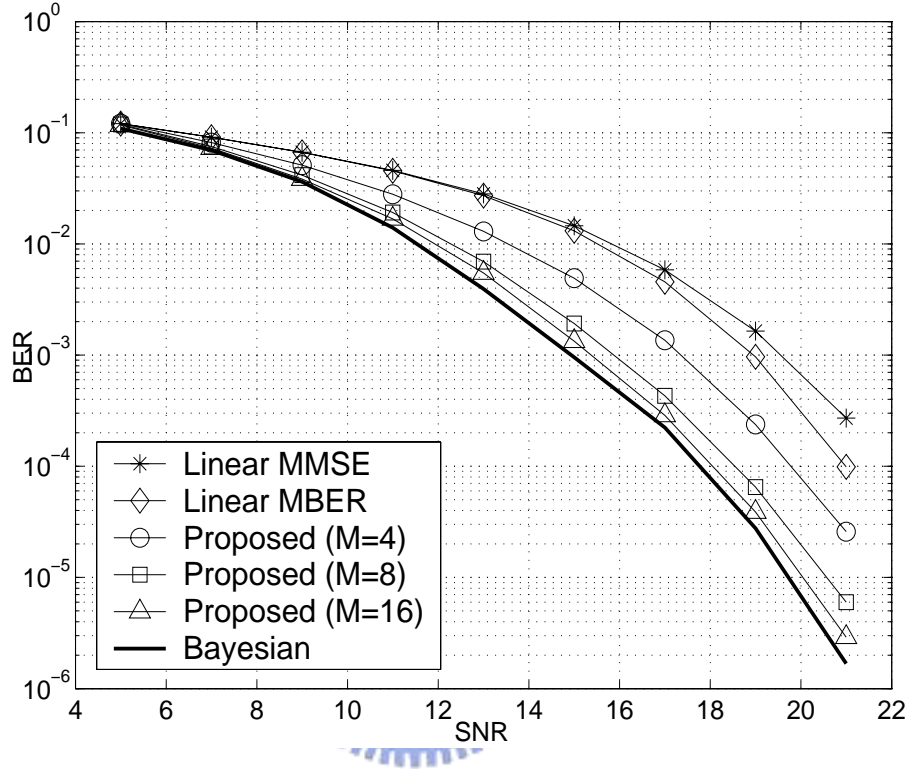


Figure 4.6: BER comparison for the MMSE linear, MBER linear, proposed, and Bayesian equalizers.

In the figure we see that the performance of the MMSE linear equalizer was the worst and the Bayesian one was the best. There is a performance gap in between. The performance of the linear MBER equalizer [53], [54] was only slightly better than that of the linear MMSE equalizer. The performance of the proposed equalizer was close to that of the Bayesian equalizer when $M = 8$. However, the computational complexity was significantly lower. While the performance of the proposed equalizer when $M = 4$ was worse than when $M = 8$, it was much better than that of the MBER linear equalizer. We observed the learning curve for the proposed equalizer ($M = 4$ and SNR=15dB) and found that the adaptive algorithm converges around 5000 iterations. Fig. 4.7 gives a performance comparison for the proposed and the RS Bayesian equalizers.

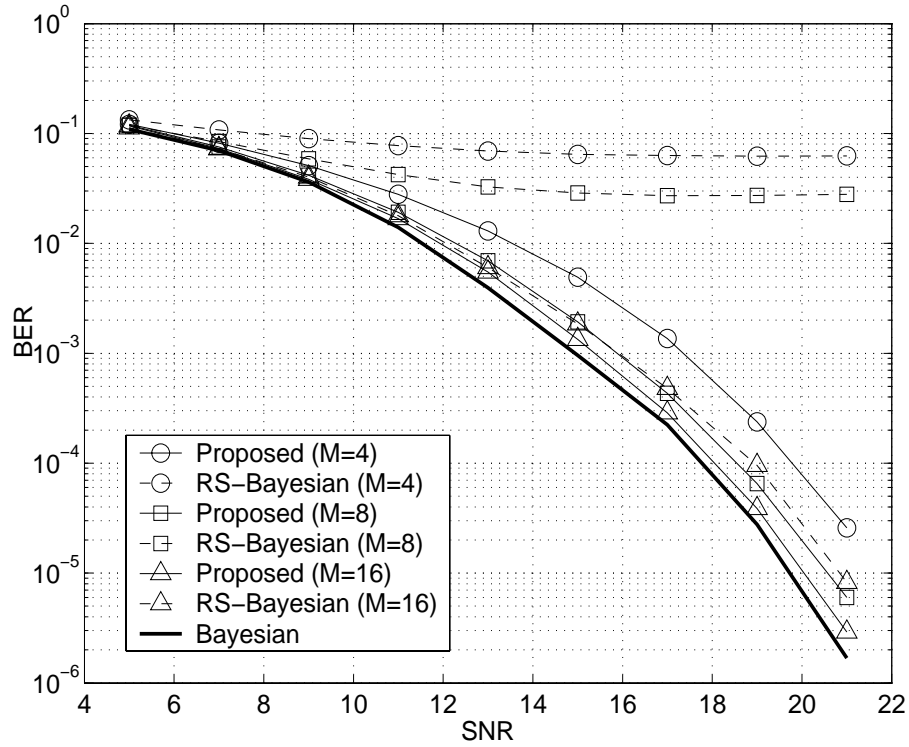


Figure 4.7: BER comparison for the proposed and RS Bayesian equalizers.

The RS Bayesian equalizer performed poorly when $M = 4$ and $M = 8$. Although the performance of the RS Bayesian equalizer was good when $M = 16$, it was worse than that of the proposed equalizer when $M = 8$. We have also tried various α values for the proposed algorithm ($M = 4$). We varied α from 0.05 to 0.8 and found that the performance was almost unaffected. The proposed algorithm performance is not sensitive to the choice of α .

The second channel we considered was also a linear channel [31]:

$$r(n) = 0.5x(n) + x(n-1) + v(n). \quad (4.77)$$

We let $L_e = 2$ and $D = 0$ for all equalizers. Note that for this scenario, the signal state spaces \mathcal{S}^+ and \mathcal{S}^- were not linearly separable. Since $L_c = 2$, $N_a = 8$. We also used $M = 4$ and $M = 8$ for the proposed equalizer. The simulation results are shown in Fig. 4.8. In the figure, we can see that the performance of the linear equalizers was very poor. This is not surprising since the signal space was not linearly separable. Even for high SNRs, the linear equalizers could not give satisfactory results. Nonlinear equalizers are

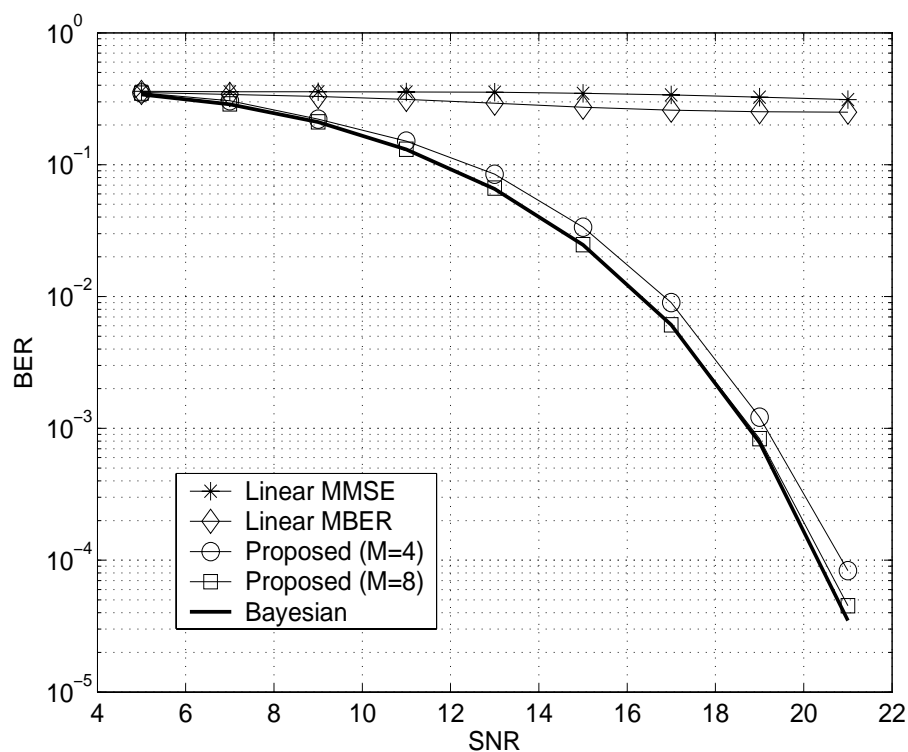


Figure 4.8: BER comparison for the MMSE linear, MBER linear, proposed and Bayesian equalizers.

not affected by this problem: the higher the SNR, the lower the BER. The performance of the proposed equalizer was very close to that of the Bayesian equalizer. When $M = 8$, there was almost no difference. To explain the reason, we show decision boundaries for the Bayesian equalizer and the proposed equalizer when $M = 4$ in Fig. 4.9. In the figure, we can clearly see that the decision boundary of the proposed equalizer closely approximated that of the Bayesian equalizer. Only in the central region in Fig. 4.9, is there some discrepancy. However, this does not contribute significant bit errors.

The decision boundary for the proposed equalizer when $M = 8$ is further shown in Fig. 4.10. Note that the decision boundary of the proposed equalizer almost exactly matched that of the Bayesian equalizer. Note also that the pseudo states identified were very close to the true signal states. This was expected since there was only one state in each subset. It is important to realize that even in this case, the computational complexity of the proposed algorithm was still lower than that of the Bayesian equalizer.

We next considered a nonlinear channel, a linear channel followed by a memoryless

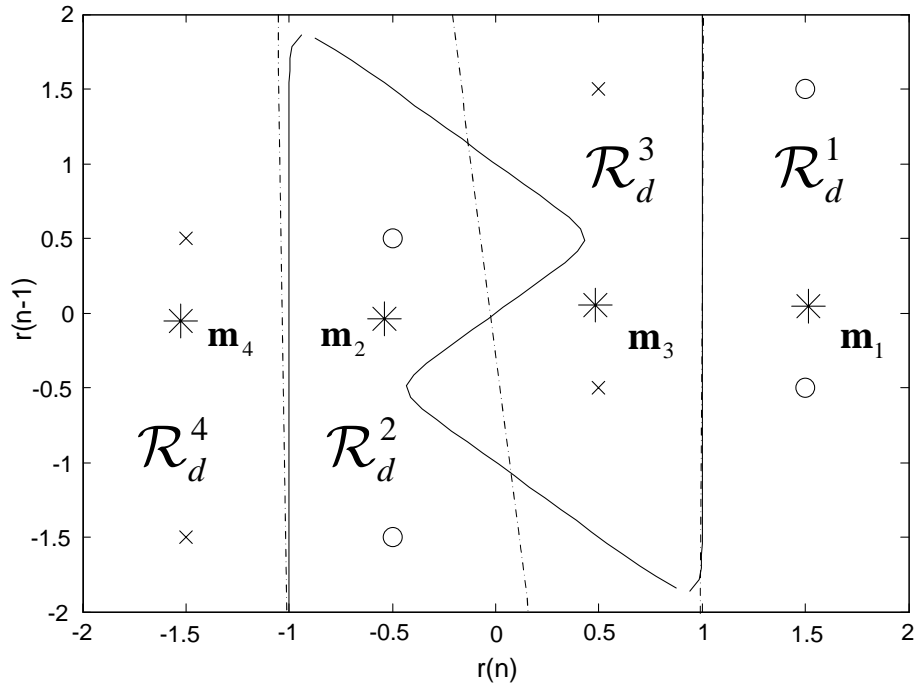


Figure 4.9: Decision regions for all subsets, $\mathcal{R}_d^i, i = 1, 2, 3, 4$, and decision boundaries of the proposed equalizer when $M = 4$ (dashed line) and the Bayesian (solid line) equalizer.

nonlinearity. The input-output relationship of the linear channel is given by

$$q(n) = 0.08x(n) + 0.42x(n-1) + x(n-2) + 0.38x(n-3) - 0.1x(n-4) + 0.09x(n-5). \quad (4.78)$$

The output of the memoryless nonlinearity is given by

$$r(n) = q(n) + 0.036q^2(n) - 0.01q^3(n) + v(n). \quad (4.79)$$

For this case, $L_c = 6$ and set $L_e = 4$ and $D = 2$ for the equalizers. It turned out that $N_a = 512$. The number of states was then large and the computational complexity of the Bayesian equalizer became huge. Fig. 4.11 shows the performance for these equalizers. Each BER result was obtained using 10^8 runs. As the figure shows, the nonlinear equalizers significantly outperform the linear ones. The proposed equalizer efficiently approximated the optimal Bayesian equalizer. While the performance loss was moderate, the computational complexity reduction was significant. Table 4.7 shows a computational complexity comparison for all equalizers (in the decision phase).

Next, we consider another nonlinear channel which has been considered in the previous

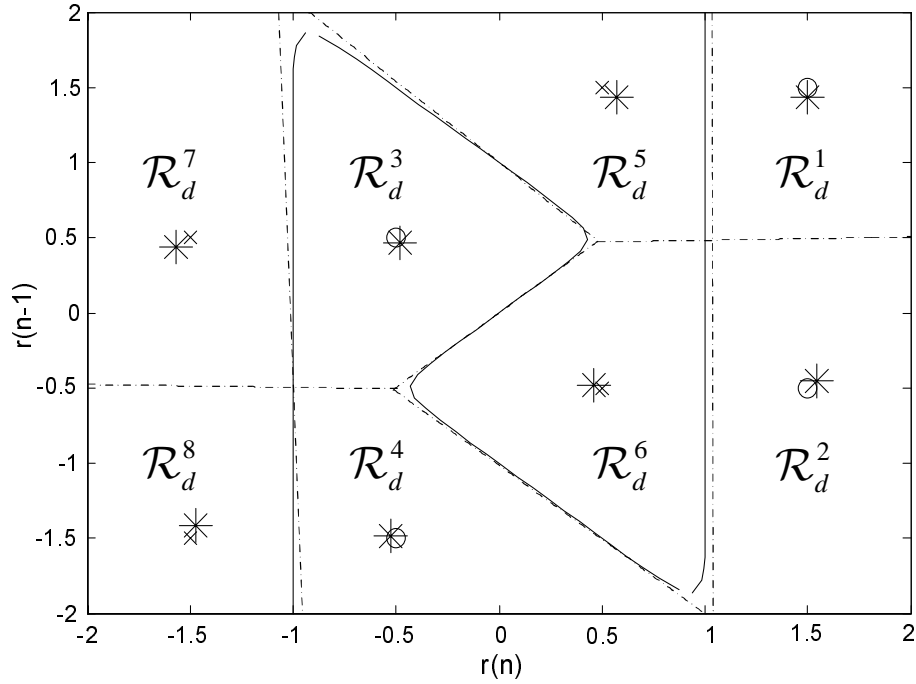


Figure 4.10: Decision regions for all subsets, $\mathcal{R}_d^i, i = 1, 2, \dots, 8$, and the decision boundaries of the proposed equalizer when $M = 8$ (dashed line) and the Bayesian (solid line) equalizer.

chapter. The input-output relationship of the linear channel is given by

$$q(n) = -0.227x(n) + 0.460x(n-1) + 0.848x(n-2) + 0.460x(n-3) - 0.227x(n-4). \quad (4.80)$$

The output of the memoryless nonlinearity is given by

$$r(n) = q(n) + 0.156q^2(n) - 0.031q^3(n) + v(n). \quad (4.81)$$

First, we do not include the decision feedback signals and set $L_e = 5, D = 5$. The total number of signal states in Bayesian equalizer is $N_a = 512$. For the proposed equalizer, we let $M = 4, M = 8, M = 16$ and $M = 32$, for which $\mathbf{x}_2(n) = [x(n-4), x(n-5)]^T$ and $\mathbf{x}_2(n) = [x(n-3), x(n-4), x(n-5)]^T$, $\mathbf{x}_2(n) = [x(n-2), x(n-3), x(n-4), x(n-5)]^T$, $\mathbf{x}_2(n) = [x(n-2), x(n-3), x(n-4), x(n-5), x(n-6)]^T$, respectively. The result is shown in Fig. 4.12. As the figure shows, the nonlinear equalizers significantly outperform the linear ones. The performance was very close to the Bayesian equalizer when $M = 32$. However, the Bayesian equalizer needs 512 signal states and the method proposed by Chen [39] needs 301 hyperplanes. Our approach is much more efficient than existing methods.

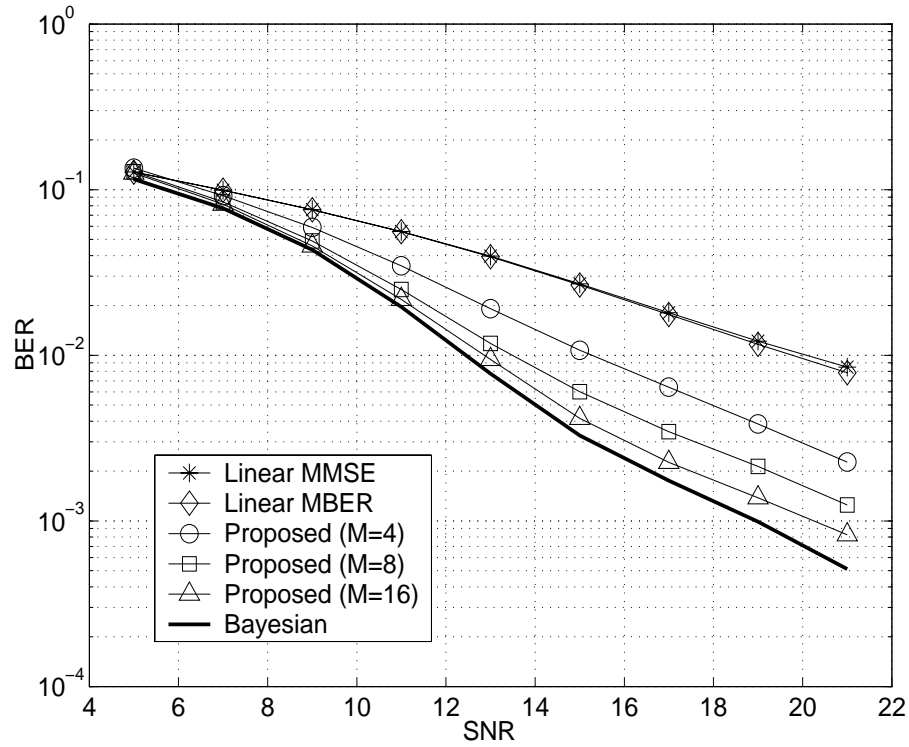


Figure 4.11: BER comparison for the MMSE linear, MBER linear, proposed, and Bayesian equalizers.

We then compare the performance of all equalizers with decision feedback. For this case, we set $L_e = 5$, $P = 2$, $D = 5$, $Q = 0$ and $S = 0$. The decision feedback term are $\hat{\mathbf{x}}_4(n) = [\hat{x}(n-6), \hat{x}(n-7), \hat{x}(n-8)]^T$. For the proposed equalizer, we let $M = 4$, $M = 8$ and $M = 16$, for which $\mathbf{x}_1(n) = [x(n-4), x(n-5)]^T$, $\mathbf{x}_2(n) = [x(n-3), x(n-4), x(n-5)]^T$ and $\mathbf{x}_3(n) = [x(n-2), x(n-3), x(n-4), x(n-5)]^T$, respectively. The performance comparison is shown in Fig. 4.13. This figure shows that, once again, the performance of the conventional DFE is the worst and the Bayesian one is the best. Note that the performance gap between them is bigger due to higher nonlinearity of the channel. While the proposed equalizer when $M = 4$ was worse than when $M = 8$ and $M = 16$, it was much better than that of conventional DFE. The performance of the proposed DFE with $M = 16$ is very close to that of the Bayesian DFE. Also note that the performance of the reduced storage method degrade slightly.

Table 4.7: Computational complexity comparison for the linear, proposed ($M = 8$), and Bayesian equalizers (for the nonlinear channel in simulations)

	Bayesian	Proposed	Linear
Multiplications	2560	32	4
Additions	4095	32	3
Others	$512 \times \exp(\cdot)$	Compare logics	

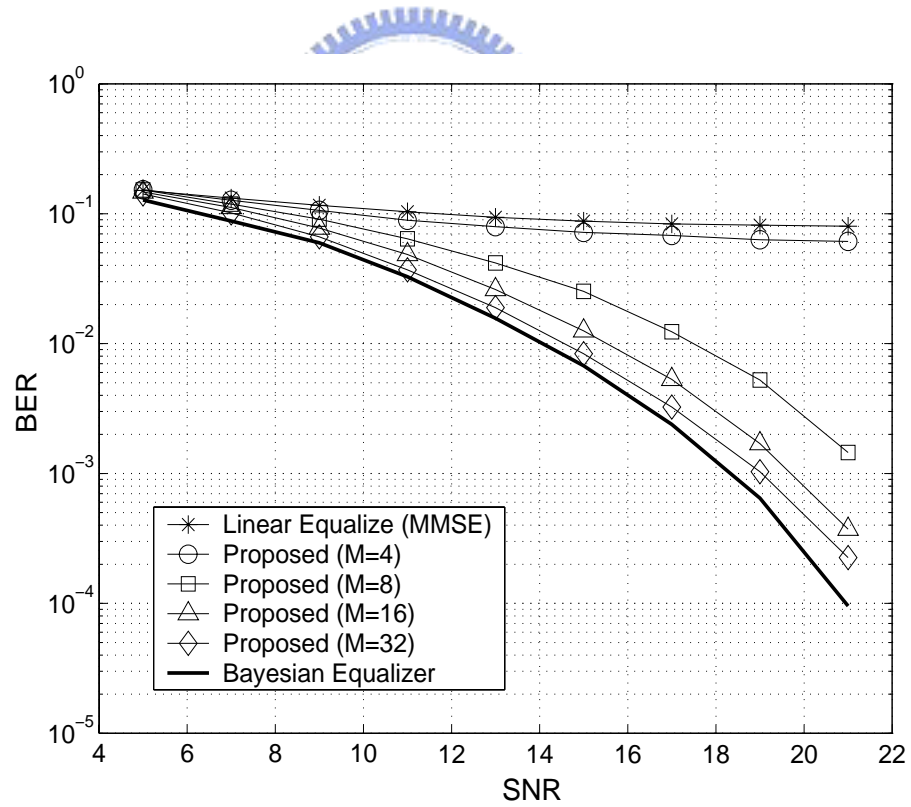


Figure 4.12: BER comparison for the MMSE linear, proposed, and Bayesian equalizers.

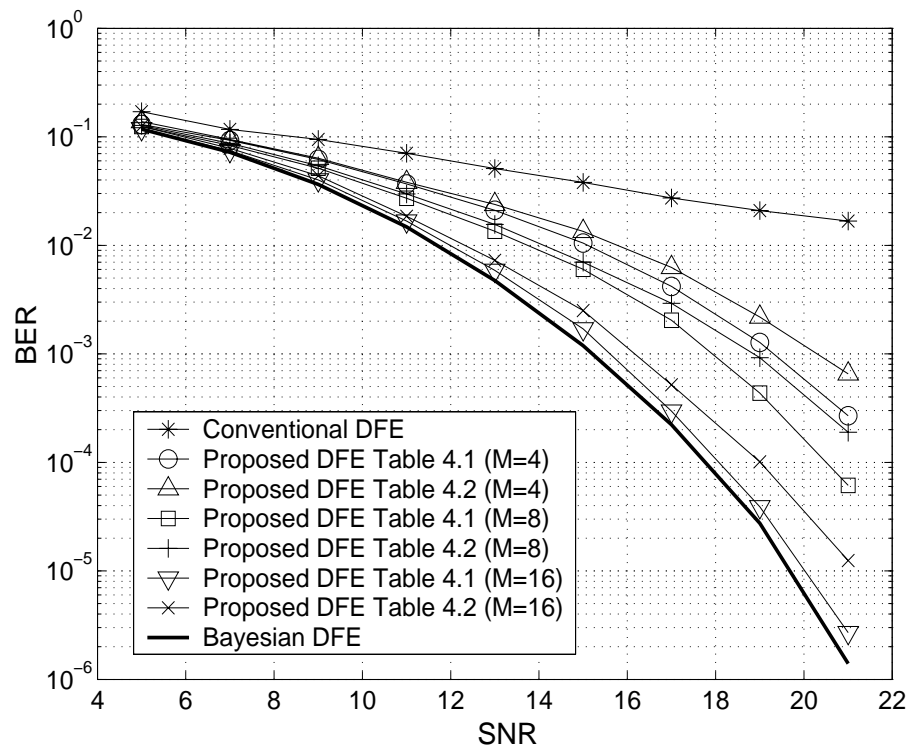


Figure 4.13: BER comparison for the Conventional DFE, proposed DFE, and Bayesian DFE.

Chapter 5

Adaptive Asymptotic Bayesian Equalization Using a set of Discriminant Functions

In this chapter, we consider equalization as a classical pattern recognition problem and propose nonlinear equalizers using the discriminant function approach. The classifier proposed in [43], [44] belongs to the same type. In [43], [44], observations for a possible transmit symbol are mapped to a class and a nonlinear discriminant (neural network) is developed for the class. In the proposed method, we let observations for a possible transmit symbol be mapped to multiple classes and multiple linear discriminant functions are derived for the class. In other words, we employ a large set of linear discriminant functions instead of a small set of nonlinear functions in the equalizer. We develop a mapping method that is independent of the channel response and can arbitrarily set the number of discriminant functions. This allows a easy trade-off between performance and computational complexity. We also develop an adaptive method that can identify the linear discriminant functions and make the proposed algorithm applicable in time-varying environments. Except for the feedforward equalization algorithm, we have also consider efficient and fast convergent decision feedback algorithms. The strategy to use the decision feedback signal is different from that in [39] and [41]. The choice of discriminant functions does not depend on channel responses and the equalizer design is the same for linear and nonlinear channels.

Note here that the results derived in this chapter is very similar to those in the previous chapter. However, there implications are different. The method in the previous chapter

starts from the signal state point of view while the method here from the discriminant function point of view. The number of parameters required to identify is also different. It turns out that the linear discriminant function used here has one more parameter. This will not affect equalization in general scenario. However, for efficient and fast implementations, there will be a difference. The method developed in this chapter will perform better.

5.1 Bayesian Equalization Using the Linear Discriminant Function Approach

Although the feedforward and decision feedback Bayesian equalizers are optimal, their computational complexities are usually very high. This is because we have to evaluate N_a exponential terms in (2.14) and N_d terms in (2.22), and N_a and N_d are increased exponentially with the channel length.

From (2.12)–(2.14), (3.99) and the Bayes' rule, we can see that the Bayesian equalizer is actually an optimal classifier. However, its *a posteriori* probability function is rather complex. Thus, we can approximate it with a classifier with two discriminant functions. We can have the decision rule shown below.

$$i = \arg \max_l f_l(\mathbf{r}(n), \Lambda_l), \quad l = 1, 2 \quad (5.1)$$

$$\hat{x}(n - D) = \begin{cases} 1, & \text{if } i = 1 \\ -1, & \text{if } i = 2 \end{cases} \quad (5.2)$$

To have the same performance as the Bayesian equalizer, the discriminant functions must be able to form the same decision boundary as that by the likelihood functions $\chi^{(+)}(\mathbf{r}(n))$ and $\chi^{(-)}(\mathbf{r}(n))$ in (2.13). Since these functions are highly nonlinear, the discriminant functions that can give the similar performance have to be the nonlinear also. In [43] and [44], a family of MLP networks was proposed as the discriminant functions. However, since the number of required parameters is large, its computational complexity is high.

As we shown, the linear discriminant function has low computational complexity and easy to apply. However, its performance is usually not satisfactory in the application here. This is because the *a posteriori* probability is non-Gaussian; it is a mixture of many

Gaussian components. It is possible to re-formulate the Bayesian equalization problem such that linear discriminant functions can be applied.

Recall the Bayesian equalizer

$$\hat{x}(n-D) = \begin{cases} +1, & f_B(\mathbf{r}(n)) > 0 \\ -1, & f_B(\mathbf{r}(n)) \leq 0 \end{cases}, \quad (5.3)$$

where

$$f_B(\mathbf{r}(n)) = \sum_{\mathbf{s}_i \in \mathcal{S}^+} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma_v^2}\right) - \sum_{\mathbf{s}_i \in \mathcal{S}^-} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma_v^2}\right). \quad (5.4)$$

If σ is small (high SNR), the summation operation can be approximated by a maximum operation.

$$\begin{aligned} \sum_{\mathbf{s}_i \in \mathcal{S}^\pm} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma^2}\right) &\approx \max_{\mathbf{s}_i \in \mathcal{S}^\pm} \exp\left(-\frac{\|\mathbf{r}(n) - \mathbf{s}_i\|^2}{2\sigma^2}\right) \\ &= \max_{\mathbf{s}_i \in \mathcal{S}^\pm} \left(\mathbf{s}_i^T \mathbf{r}(n) - \frac{\|\mathbf{s}_i\|^2}{2}\right). \end{aligned} \quad (5.5)$$

Using (5.5), we can re-formulate the decision rule in (5.3) and (5.4) as

$$\hat{x}(n-D) = \begin{cases} +1, & \mathbf{s}_k \in \mathcal{S}^+ \\ -1, & \mathbf{s}_k \in \mathcal{S}^- \end{cases}, \quad (5.6)$$

where

$$k = \arg \max_l \mathbf{s}_l^T \mathbf{r}(n) - \frac{\|\mathbf{s}_l\|^2}{2}, \quad 1 \leq l \leq N_a. \quad (5.7)$$

We now can define a set of linear discriminant functions as

$$f_l(\mathbf{r}(n), \Lambda_l) = \mathbf{s}_l^T \mathbf{r}(n) - \frac{\|\mathbf{s}_l\|^2}{2}, \quad 1 \leq l \leq N_a, \quad (5.8)$$

where $\Lambda_l = [\mathbf{s}_l^T, \|\mathbf{s}_l\|^2/2]^T$. From (5.7), we can re-write the decision rule as

$$\hat{x}(n-D) = \begin{cases} +1, & \mathbf{s}_k \in \mathcal{S}^+ \\ -1, & \mathbf{s}_k \in \mathcal{S}^- \end{cases}, \quad (5.9)$$

where

$$k = \arg \max_l f_l(\mathbf{r}(n), \Lambda_l) \quad \text{for } 1 \leq l \leq N_a. \quad (5.10)$$

The equalizer in (5.9) can asymptotically ($\sigma \rightarrow 0$) approximate the Bayesian equalizer. With this approximation, we now have a different view for the equalizer problem. The

main difference is that the equalization problem is no longer considered as a two-class classification problem (for BPSK signal). It is a N_a -class classification problem. Or, we may say that we still have a two-class classification problem. But now, each class has $N_a/2$ subclasses and classification is performed at the subclass level. With this new perspective, we can have a completely new equalization concept. However, since N_a is usually large, the computational complexity is still high. In what follows, we will propose a new approach to solve this problem.

5.2 Proposed Equalizer

The main idea of our approach is to treat equalization as a multiple-class instead of a two-class classification problem. By doing so, we can use a set of linear discriminant functions instead of two highly nonlinear discriminant functions. As we have seen, the number of the linear discriminant function can be large. We now propose a method to solve the problem. We first consider the scenario without decision feedback. Rewrite the data vector $\mathbf{x}(n)$ as a combination three vectors.

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T, \quad (5.11)$$

where

$$\begin{aligned} \mathbf{x}_1(n) &= [x(n), x(n-1), \dots, x(n-P+1)]^T, \\ \mathbf{x}_2(n) &= [x(n-P), x(n-P-1), \dots, x(n-Q+1)]^T, \\ \mathbf{x}_3(n) &= [x(n-Q), x(n-Q-1), \dots, x(n-L_c-L_e+2)]^T, \end{aligned} \quad (5.12)$$

where $Q > D \geq P \geq 0$ and $L_c + L_e - 2 \geq Q \geq 1$. Note that $\mathbf{x}_2(n)$ includes $x(n-D)$ and its length is $Q - P$. Depending on the value of D , $\mathbf{x}_1(n)$ or $\mathbf{x}_3(n)$ may or may not exist. For example, if $D = 0$, we do not have $\mathbf{x}_1(n)$. Let $M = 2^{Q-P}$. We then have M possible vector values for $\mathbf{x}_2(n)$. Denote these vectors as \mathbf{x}_i , $1 \leq i \leq M$. Now, we can divide the signal state set according to the value of $\mathbf{x}_2(n)$:

$$\mathcal{S}^i \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_2(n) = \mathbf{x}_i\}, \quad 1 \leq i \leq M \quad (5.13)$$

and

$$\mathcal{S} = \bigcup_{1 \leq i \leq M} \mathcal{S}^i. \quad (5.14)$$

Since we have M signal subsets, we can then partition the received signal space into M classes using M discriminant functions denoted by

$$f_i(\mathbf{r}(n), \Lambda_i) = \mathbf{w}_i^T \mathbf{r}(n) + b_i, \quad (5.15)$$

where $\Lambda_i = [\mathbf{w}_i^T, b_i]^T$. How to identify the unknown parameters will be discussed in the next section. For the time being, we simply assume that the parameters in these discriminant functions are known. Define \mathcal{I}^\pm as a set with indexes such that

$$\mathcal{I}^\pm \triangleq \{i \in \mathcal{N} : J_D^T \mathbf{x}_i = \pm 1\}. \quad (5.16)$$

where \mathbf{J}_D is $(Q - P) \times 1$ vector and

$$\mathbf{J}_D = [\underbrace{0 \ \cdots \ 0 \ 1}_{D-P+1} \ 0 \ \cdots \ 0]^T. \quad (5.17)$$

Thus, \mathcal{I}^+ is the index set corresponding to the subsets of \mathcal{S}^+ . Then, the final decision can be obtained as

$$i = \arg \max_l f_l(\mathbf{r}(n), \Lambda_l), \quad (5.18)$$

$$\hat{x}(n - D) = \begin{cases} +1, & i \in \mathcal{I}^+ \\ -1, & i \in \mathcal{I}^- \end{cases}. \quad (5.19)$$

We now discuss what the decision region the proposed method can form and how it can approximate the Bayesian decision boundary. If $\mathbf{r}(n)$ is in the decision region of Class i , then

$$f_i(\mathbf{r}(n), \Lambda_i) \geq f_k(\mathbf{r}(n), \Lambda_k), \quad 1 \leq k \leq M \text{ and } i \neq k. \quad (5.20)$$

Note that each inequality in (5.20) forms a hyperplane between Class i and k . If we pretend that only these two classes exist, this gives a decision region for Class i . For the same Class i , there are $M - 1$ such regions ($k = 1, 2, \dots, M, k \neq i$) and the true decision region for Class i is the intersection of these regions. Let $\mathcal{R}_d^{i,k}$ be the decision region formed by Class i and k (with respect to Class i) and \mathcal{R}_d^i is the decision region for Class i . Then,

$$\mathcal{R}_d^{i,k} \triangleq \{ \mathbf{r}(n) \in \mathcal{R}^{L_e} : f_i(\mathbf{r}(n), \Lambda_i) \geq f_k(\mathbf{r}(n), \Lambda_k) \}, \quad (5.21)$$

and

$$\mathcal{R}_d^i \triangleq \bigcap_{k \neq i} \mathcal{R}_d^{i,k}. \quad (5.22)$$

Then the final decision region \mathcal{R}_d^\pm corresponding to $x(n-D) = \pm 1$ is

$$\mathcal{R}_d^\pm \triangleq \bigcup_{i \in \mathcal{I}^\pm} \mathcal{R}_d^i. \quad (5.23)$$

The proposed method shown above partition the signal space into 2^I where I is an integer. And, we apply 2^I discriminant functions in the equalizer.

5.3 Discriminative Learning Algorithm

We have transformed the equalization problem into a multiple-class classification problem and proposed to use a linear discriminant function approach. The remaining work is how to obtain the parameters in the discriminant functions, i.e., \mathbf{w}_i and b_i in $f_i(\mathbf{r}(n), \Lambda_i)$ for $1 \leq i \leq M$. As described in Chapter 3, [45] proposed a discriminative learning process to solve the problem. The advantage of this method is that it can yield the minimum classification error rate. In this paper, we then use this method to identify our discriminant functions. The cost function in [45] can be developed in three steps. The first step is to select discriminant functions $f_i(\mathbf{r}(n), \Lambda_i)$ and transform them into positive functions. Let the transform function be $\phi(\cdot)$ and the transformed discriminant function be $g_i(\mathbf{r}(n), \Lambda_i)$. Then, $g_i(\mathbf{r}(n), \Lambda_i) = \phi(f_i(\mathbf{r}(n), \Lambda_i))$. The second step is to define a mis-classification measure. Let $\mathbf{r}(n)$ belong to i th class. Then, the mis-classification measure is defined as

$$d_i(\mathbf{r}(n)) = -g_i(\mathbf{r}(n), \Lambda_i) + \left[\frac{1}{M-1} \sum_{k \neq i} g_k(\mathbf{r}(n), \Lambda_k)^\eta \right]^{1/\eta}, \quad (5.24)$$

where η is a positive number. Finally, define the cost function as

$$J(n) = E[L(d_i(\mathbf{r}(n)))], \quad (5.25)$$

where

$$L(d_i(\mathbf{r}(n))) = \frac{1}{1 + e^{-\varepsilon(d_i(\mathbf{r}(n)) + \alpha)}}, \quad (5.26)$$

where $E[\cdot]$ denotes the statistical expectation, $\varepsilon > 0$, and $\alpha > 0$. By the virtue of the discriminant function, if an observation belongs to the class i , $g_i(\mathbf{r}(n), \Lambda_i)$ will be large,

and $g_j(\mathbf{r}(n), \Lambda_i)$, $j \neq i$, will be small. Thus, $L(d_i(\mathbf{r}(n)))$ will approach zero. On the contrary, if the observation does not belong to the class i , $g_i(\mathbf{r}(n), \Lambda_i)$ will be small, and $g_j(\mathbf{r}(n), \Lambda_i)$, $j \neq i$, will be large. And, $L(d_i(\mathbf{r}(n)))$ will approach one. It was proved in [45] that if a cost function has this “zero-one” characteristics, a classifier trained using the cost function will achieved a minimum classification error rate.

For the proposed method, we choose linear discriminant functions and a transformation function given by

$$\phi(x) = \frac{1}{1 + e^{-x}} \quad (5.27)$$

Also, the class is defined according to the $\mathbf{x}_2(n)$ pattern. In other words,

$$\mathbf{r}(n) \in C_i, \text{ if } \mathbf{x}_2(n) = \mathbf{x}_i, \quad 1 \leq i \leq M, \quad (5.28)$$

where C_i denotes the i th class. Parameters to be determined are ε , α , and η . Among them, η is the most critical one. We now state how to choose that. As shown in (2.7), the observation is Gaussian distributed in the signal space. The density function decay fast deviating from signal states. For a given set of discriminant function parameters, the classifier will make an correct decision if an observation belongs to the class i and $g_i(\cdot) > g_k(\cdot)$, $k \neq i$. Thus, in a high SNR environment, only the largest discriminant function in $g_k(\cdot)$'s, $k \neq i$ may affect the decision. Thus, we propose to let $\eta = \infty$. In this case, the second term at right hand side of (5.24) becomes

$$\left[\frac{1}{M-1} \sum_{k \neq i} g_k(\mathbf{r}(n), \Lambda_k)^\eta \right]^{1/\eta} = \max_{k \neq i} g_k(\mathbf{r}(n), \Lambda_k). \quad (5.29)$$

Thus, (5.24) can be re-written as

$$d_i(\mathbf{r}(n)) = -g_i(\mathbf{r}(n), \Lambda_i) + g_k(\mathbf{r}(n), \Lambda_k), \quad (5.30)$$

where

$$k = \arg \max_{l \neq i} g_l(\mathbf{r}(n), \Lambda_l). \quad (5.31)$$

We found that this choice not only yields satisfactory results, but also reduces the computational complexity for parameter adaptation.

Since (5.25) is a nonlinear cost function, it is difficult to obtain the optimal solution directly. Here, we employ the adaptive method to solve the problem. The commonly used method is the steepest descent method. The adaptive algorithm is given by

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) - \mu \nabla_{\mathbf{w}_i} J(n), \quad (5.32)$$

$$b_i(n+1) = b_i(n) - \mu \nabla_{b_i} J(n). \quad (5.33)$$

where $\nabla_{\mathbf{w}_i} J(n)$ denotes the gradient vector of $J(n)$ with respect to \mathbf{w}_i . In reality, however, exact measurement of the gradient vector is not possible. The gradient vector must be estimated from the available data. The simplest estimate is the stochastic gradient in which the expectation operations in (5.32)-(5.33) are ignored. After some algebra calculation, we obtain the following adaptive algorithm.

For notation simplicity, we use $g_i(n)$ and $L_i(n)$ to denote $g_i(\mathbf{r}(n), \Lambda_i)$ and $L(d_i(\mathbf{r}(n)))$, respectively. In the training mode, if $\mathbf{x}_2(n)$ of $\mathbf{x}(n)$ is \mathbf{x}_i (the corresponding received signal vector is $\mathbf{r}(n)$), we then have the following adaptive algorithm ($k = \arg \max_{l \neq i} g_l(n)$):

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \mu L_i(n)(1 - L_i(n))g_i(n)(1 - g_i(n))\mathbf{r}(n), \quad (5.34)$$

$$b_i(n+1) = b_i(n) + \mu L_i(n)(1 - L_i(n))g_i(n)(1 - g_i(n)) \cdot 1, \quad (5.35)$$

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) - \mu L_i(n)(1 - L_i(n))g_k(n)(1 - g_k(n))\mathbf{r}(n), \quad (5.36)$$

$$b_k(n+1) = b_k(n) - \mu L_i(n)(1 - L_i(n))g_k(n)(1 - g_k(n)) \cdot 1. \quad (5.37)$$

Since the cost function is a nonlinear function, the adaptive algorithm may converge to a local minimum. Thus, a proper choice of initial value is important. Based on the the relationship of signal state and the linear discriminant function shown in (5.5), (5.8), we suggest using the clustering method [28] to obtain a reliable initial value. The computational complexity requirement for the method is low. The clustering method is given as follows. If a received signal vector $\mathbf{r}(n)$ with $\mathbf{x}_2(n) = \mathbf{x}_i$, then

$$\mathbf{w}_i(n+1) = \text{counter}_i \times \mathbf{w}_i(n) + \mathbf{r}(n), \quad (5.38)$$

$$\text{counter}_i = \text{counter}_i + 1, \quad (5.39)$$

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n+1)/\text{counter}_i, \quad (5.40)$$

$$b_i(n+1) = -\frac{\|\mathbf{w}_i(n+1)\|^2}{2}. \quad (5.41)$$

5.4 Decision Feedback Equalization

The procedure developing decision feedback equalization is similar to that in 4.4. For completeness, we re-describe the procedure. Define the input data vector $\mathbf{x}(n)$ be a combination four vectors

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n), \mathbf{x}_4^T(n)]^T, \quad (5.42)$$

where

$$\begin{aligned} \mathbf{x}_1(n) &= [x(n), x(n-1), \dots, x(n-P+1)]^T, \\ \mathbf{x}_2(n) &= [x(n-P), x(n-P-1), \dots, x(n-Q+1)]^T, \end{aligned} \quad (5.43)$$

$$\mathbf{x}_3(n) = [x(n-Q), x(n-Q-1), \dots, x(n-S+1)]^T, \quad (5.44)$$

$$\mathbf{x}_4(n) = [x(n-S), x(n-S-1), \dots, x(n-L_c-L_e+2)]^T, \quad (5.45)$$

$$(5.46)$$

where $S \geq Q > D \geq P \geq 0$ and $L_c + L_e - Q - 2 \geq S \geq Q \geq 1$. The fourth component will be obtained from decision feedback. Let $\hat{\mathbf{x}}_b(n)$ be the feedback vector and decisions be correct. Then, $\mathbf{x}_4(n) = \hat{\mathbf{x}}_b(n)$. Since the length of $\hat{\mathbf{x}}_b(n)$ is $L_b = L_c + L_e - S - 1$, there are $N_b = 2^{L_b}$ possible decision pattern. As defined, we call each pattern a decision state and its possible value is $\hat{\mathbf{x}}_j$, $1 \leq j \leq N_b$. We then define the input data vector given the decision state j as

$$\mathbf{x}_j(n) = [\mathbf{x}_{1,j}^T(n), \mathbf{x}_{2,j}^T(n), \mathbf{x}_{3,j}^T(n), \hat{\mathbf{x}}_b^T(n) = \hat{\mathbf{x}}_j^T]^T. \quad (5.47)$$

Note that $\mathbf{x}_{2,j}(n)$ includes $x(n-D)$ and its length is $Q-P$. Similar to the feedforward equalization scenario, $\mathbf{x}_{1,j}(n)$ and $\mathbf{x}_{2,j}(n)$ may or may not exist. We then have $M = 2^{P+Q+1}$ possible vector values for $\mathbf{x}_{2,j}(n)$. Denote these vector as $\mathbf{x}_{i,j}$, $1 \leq i \leq M$. Thus, we can divide the received signal space into M classes according to $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$ (for the j th decision feedback state). Assume that the decision feedback is correct then we can divide received signal space into M classes according to $\mathbf{x}_{2,j}(n) = \mathbf{x}_{i,j}$. Then we use M linear discriminant functions to discriminate these classes

$$f_{i,j}(\mathbf{r}(n), \Lambda_{i,j}) = \mathbf{w}_{i,j}^T \mathbf{r}(n) + b_{i,j}, \quad 1 \leq i \leq M, \quad 1 \leq j \leq N_b \quad (5.48)$$

where $\Lambda_{i,j} = [\mathbf{w}_{i,j}^T, b_{i,j}]^T$ and $N_b = 2^{L_b}$. Define \mathcal{I}_j^\pm as a set with indexes such that

$$\mathcal{I}_j^\pm \triangleq \{i \in \mathcal{N} : J_D^T \mathbf{x}_{i,j} = \pm 1\}, \quad (5.49)$$

where \mathbf{J}_D is $(Q - P) \times 1$ vector and

$$\mathbf{J}_D = \underbrace{[0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]}_{D-P+1}^T. \quad (5.50)$$

Thus, \mathcal{I}_j^\pm is the index set corresponding to the subset in \mathcal{S}_j^\pm . Then, the final decision can be obtained as

$$i = \arg \max_l f_{l,j}(\mathbf{r}(n), \Lambda_{i,j}), \quad (5.51)$$

$$\hat{x}(n - D) = \begin{cases} 1, & i \in \mathcal{I}_j^+ \\ -1, & i \in \mathcal{I}_j^- \end{cases}. \quad (5.52)$$

The overall structure of the proposed nonlinear equalizer shown in Fig. 4.5. Note that, if we do not use the feedback signal, the selection part can be ignored.

We need also to decide the unknown parameters $\mathbf{w}_{i,j}$ and $b_{i,j}$. Using the similar cost function, we can identify the parameters. Thus, we have

$$\min J_j(n) = E[L(d_{i,j}(\mathbf{r}(n)))], \quad (5.53)$$

where

$$L_j(d_{i,j}(\mathbf{r}(n))) = \frac{1}{1 + e^{-\varepsilon(d_{i,j}(\mathbf{r}(n)) + \alpha)}}, \quad (5.54)$$

$$d_{i,j}(\mathbf{r}(n)) = -g_{i,j}(\mathbf{r}(n), \Lambda_{i,j}) + g_{k,j}(\mathbf{r}(n), \Lambda_{k,j}), \quad (5.55)$$

$$k = \arg \max_{l \neq i} g_{l,j}(\mathbf{r}(n), \Lambda_{l,j}), \quad (5.56)$$

and $g_{i,j}(\mathbf{r}(n), \Lambda_{i,j}) = \phi(f_{i,j}(\mathbf{r}(n)), \Lambda_{i,j})$. The adaptive algorithm, summarized in Table 5.1, can then be derived accordingly. For simplicity, we use $g_{i,j}(n)$ and $L_{i,j}(n)$ to denote $g_{i,j}(\mathbf{r}(n), \Lambda_{i,j})$ and $L_j(d_{i,j}(\mathbf{r}(n)))$, respectively. Also, we set $\alpha = 0$.

Since there are N_b feedback states, we then will have N_b sets of discriminant functions. Thus, we have to store all these functions and the memory size may be large. Here, we propose a method to alleviate this problem. As mentioned above, $\mathbf{s}_{i,j}$ denotes one possible signal state given $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$. More accurately, $\mathbf{s}_{i,j}$ in (2.17) is a mapping of the i th input vector $[\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T$ given $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$. Consider the linear channel scenario, i.e.,

$$g(x(n), \cdots, x(n - L_c + 1)) = \sum_{i=0}^{L_c-1} c_i x(n - i), \quad (5.57)$$

Table 5.1: The adaptive algorithm of proposed equalizer with decision feedback.

-
1. Initialize variables: $n=0$,
 $\mathbf{w}_{i,j}(0) = \mathbf{0}$ and $b_{i,j}(0) = 0$ for $i = 1, \dots, M$ and $j = 1, \dots, N_b$
 2. Clustering method (find the initial value), $n = 1, 2, \dots$
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 $\mathbf{w}_{i,j}(n+1) = \text{counter}_{i,j} \times \mathbf{w}_{i,j}(n) + \mathbf{r}(n)$,
 $\text{counter}_{i,j} = \text{counter}_{i,j} + 1$,
 $\mathbf{w}_{i,j}(n+1) = \mathbf{w}_{i,j}(n+1) / \text{counter}_{i,j}$,
 $b_{i,j}(n+1) = -\frac{\|\mathbf{w}_{i,j}(n+1)\|^2}{2}$.
 end
 3. Adaptive algorithm
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 for $k = 1 : M$
 $g_{k,j}(n) = \phi(\mathbf{w}_{k,j}^T(n)\mathbf{r}(n) + b_{k,j}(n))$,
 end
 $k = \arg \max_{l \neq i} g_{l,j}(n)$
 $L_{i,j}(n) = \frac{1}{1 + e^{-\varepsilon(-g_{i,j}(n) + g_{k,j}(n))}}$
 $\mathbf{w}_{i,j}(n+1) = \mathbf{w}_{i,j}(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,j}(n)(1 - g_{i,j}(n))\mathbf{r}(n)$,
 $b_{i,j}(n+1) = b_{i,j}(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,j}(n)(1 - g_{i,j}(n)) \cdot 1$,
 $\mathbf{w}_{k,j}(n+1) = \mathbf{w}_{k,j}(n) - \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,k}(n)(1 - g_{i,k}(n))\mathbf{r}(n)$,
 $b_{k,j}(n+1) = b_{k,j}(n) - \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,k}(n)(1 - g_{i,k}(n))\mathbf{r}(n) \cdot 1$.
 end

where c_i 's are the channel input responses. The signal state $\mathbf{s}(n)$ can be written as

$$\mathbf{s}(n) = \mathbf{C}\mathbf{x}(n), \quad (5.58)$$

where \mathbf{C} is a Toeplitz matrix with the channel responses c_i 's and

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots & 0 \\ 0 & c_0 & c_1 & \cdots & c_{L_c-1} & 0 & \cdots \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{L_c-1} \end{bmatrix}. \quad (5.59)$$

Decompose $\mathbf{x}(n)$ as $\mathbf{x}(n) = [\mathbf{x}_f^T(n), \hat{\mathbf{x}}_b^T(n)]^T$ where $\mathbf{x}_f(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \mathbf{x}_3^T(n)]^T$. The matrix \mathbf{C} can then be decomposed accordingly, i.e., $\mathbf{C} = [\mathbf{C}_f, \mathbf{C}_b]$. Thus, (5.58) can be rewritten as

$$\mathbf{s}(n) = \mathbf{C}_f \mathbf{x}_f(n) + \mathbf{C}_b \hat{\mathbf{x}}_b(n). \quad (5.60)$$

From (5.60), we can see that $\mathbf{s}_{i,j} - \mathbf{s}_{i,k} = \mathbf{C}_b(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_k)$, $j \neq k$, for $1 \leq i \leq N_d$. This is to say that signal states for two feedback states has a common difference vector. Thus, we

can describe all signal states (for different given feedback states) as

$$\mathbf{s}_{i,j} = \mathbf{s}_{i,1} + \mathbf{d}_j, \quad 2 \leq j \leq N_b, \quad (5.61)$$

where \mathbf{d}_j is a difference vector and $\mathbf{d}_1 = \mathbf{0}$. Using the relationship of the signal state and the discriminant function in (5.7), we can find that each pair parameters $\mathbf{w}_{i,j}$ and $\mathbf{w}_{i,k}$ have the same relationship as that in (5.61). Thus,

$$\mathbf{w}_{i,j} = \mathbf{w}_{i,1} + \mathbf{d}_j, \quad 1 \leq i \leq M, \quad 2 \leq j \leq N_b, \quad (5.62)$$

and $\mathbf{d}_1 = \mathbf{0}$. Using this approach, all the parameters needed to identify are $\mathbf{w}_{i,1}$, $b_{i,j}$ and \mathbf{d}_j , $1 \leq i \leq M$, $2 \leq j \leq N_b$. Thus, the number of vectors to be identified is reduced from MN_b vectors to $M + N_b - 1$. Using the same cost function in (5.53), we derive the corresponding weight update equations shown in Table 5.2. For nonlinear channels, (5.61) is not valid in general.

Since in the nonlinear channel, each pair signal state $\mathbf{s}_{i,j}$ and $\mathbf{s}_{i,k}$ do not have the same difference. It will be shown later that the performance loss is small by using the reducing method in nonlinear channel. The two algorithms shown in Table 5.1 and 5.2 update parameters every N_b iterations and the convergence may not fast enough. We then propose another algorithm to solve the problem. The idea is to let the difference vectors \mathbf{d}_j 's be non-adaptive. In this case, we can update $\mathbf{w}_{i,1}$ every iteration and this can effectively accelerate the convergence rate. Note that the difference vectors are estimated (without extra cost) in the initialization stage using the clustering method. We summarize the adaptive algorithm for this approach in Table 5.3.

The computational complexity for the propose algorithms in this chapter is summarized in Table 5.4 and 5.5, respectively. Roughly speaking, the complexity of the proposed equalizers is M times higher than the conventional DFE. The actual choice of M is dependent on the desired performance. In our design, the decision region for each subset is automatically and adaptively formed, and these decision regions approximate the Bayesian decision regions. In principle, if the Bayesian decision boundary has a complex shape, we need more subsets. In many cases, however, only a small number of subsets is sufficient. The computational complexity of the proposed equalizer can be much lower than that of

the Bayesian equalizers. Note that if a time-varying channel is considered, decisions can be used to train the proposed equalizer continuously such that channel variations can be properly tracked.

5.5 A New cost function

In the previous section, we have used the cost function in [45] to train discriminant functions. Note that noise is Gaussian. Its distribution is an exponential function. However, the mis-classification measure, $g_k(\cdot)$'s, $k \neq i$, is a polynomial function. As a result, the cost in (5.24) may not be able to reflect the Gaussian characteristics. Thus, we let $\eta \rightarrow \infty$ in the previous section. Using the clustering algorithm, we can obtain proper initial values such that the adaptive algorithm (5.34)–(5.37) can converge to the global minimum. However, there are some circumstances, the clustering cannot be applied (for the applications in the next chapter for example). In these scenarios, the adaptive algorithms may converge to local minimums. In this section, we propose a new cost function that can alleviate this problem. From extensive simulations, we found that this cost function can always let the adaptive algorithm converge to the global minimum without specific initials.

Stimulating by the method in the previous chapter, we first define a cost for each pair $g_i(\cdot)$ and $g_k(\cdot)$, $k \neq i$ and then combine these costs to form a overall cost function. Let $\eta = 1$ in (5.24), and write (5.24) in an alternative form as

$$d_i(\mathbf{r}(n)) = \frac{1}{M-1} \sum_{k \neq i} [-g_i(\mathbf{r}(n), \Lambda_i) + g_k(\mathbf{r}(n), \Lambda_k)]. \quad (5.63)$$

Each term in the bracket of (5.63) can define its individual cost as

$$J_{i,k}(n) = \frac{1}{1 + \exp(-\epsilon(-g_i(\mathbf{r}(n), \Lambda_i) + g_k(\mathbf{r}(n), \Lambda_k)))}. \quad (5.64)$$

Examining (5.64), we can find that if $g_i(\mathbf{r}(n), \Lambda_i) \geq g_k(\mathbf{r}(n), \Lambda_k)$ and ϵ is large then the cost will approach to zero. On the contrary, if $g_i(\mathbf{r}(n), \Lambda_i) < g_k(\mathbf{r}(n), \Lambda_k)$ the cost will approach to one. Note that ϵ corresponds to the noise level. Finally, we combine these costs to form the overall cost as below. If $\mathbf{x}_2(n) = \mathbf{x}_i$ then

$$J(n) = \frac{1}{M-1} \sum_{k \neq i} E[J_{i,k}(n)], \quad (5.65)$$

where

$$J_{i,k}(n) = \frac{1}{1 + \exp(-\epsilon(-g_i(\mathbf{r}(n), \Lambda_i) + g_k(\mathbf{r}(n), \Lambda_k)))}. \quad (5.66)$$

Examining (5.65) again, we find that if the received signal $\mathbf{r}(n)$ belongs to i th class and $g_i(\mathbf{r}(n), \Lambda_i) \geq g_k(\mathbf{r}(n), \Lambda_k)$, $1 \leq k \leq M$, $k \neq i$, the decision is correct and the cost function is small. On the contrary, if the received signal $\mathbf{r}(n)$ belongs to i th class, but any one $g_k(\mathbf{r}(n), \Lambda_k)$ is large than $g_i(\mathbf{r}(n), \Lambda_i)$, the decision will be not correct and the cost function will approach one. Thus, this cost function approach to a smoothed zeros-one function. The shape of the function is controlled by the parameter ϵ . The cost function in (5.65) reflects the Gaussian characteristics better than that in (5.24).

The adaptive algorithm for (5.65) is given as follows. If a received signal vector $\mathbf{r}(n)$ with $\mathbf{x}_2(n) = \mathbf{x}_i$ we than have the following adaptive algorithm ($i \neq k$)

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \mu \sum_{k \neq i} J_{i,k}(n)(1 - J_{i,k}(n))g_i(\mathbf{r}(n))(1 - g_i(\mathbf{r}(n)))\mathbf{r}(n), \quad (5.67)$$

$$b_i(n+1) = b_i(n) + \mu \sum_{k \neq i} J_{i,k}(n)(1 - J_{i,k}(n))g_i(\mathbf{r}(n))(1 - g_i(\mathbf{r}(n))) \cdot 1, \quad (5.68)$$

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) - \mu J_{i,k}(n)(1 - J_{i,k}(n))g_k(\mathbf{r}(n))(1 - g_k(\mathbf{r}(n)))\mathbf{r}(n), \quad (5.69)$$

$$b_k(n+1) = b_k(n) - \mu J_{i,k}(n)(1 - J_{i,k}(n))g_k(\mathbf{r}(n))(1 - g_k(\mathbf{r}(n))) \cdot 1. \quad (5.70)$$

If we let $\epsilon \rightarrow \infty$, the adaptive algorithm will become identical to that in the previous section. As mentioned, the adaptive algorithm (5.67)–(5.70) can converge to the global minimum even without proper initials. One disadvantage using this cost function is that the computational complexity for its adaptive algorithm is higher. With proper initials, both cost functions give the same performance.

5.6 Simulation Results

we consider a nonlinear channel which a linear channel followed by a memoryless nonlinear function. The input-output relationship of the linear channel is given by

$$q(n) = 0.1x(n) + 0.4084x(n-1) + 0.8164x(n-2) + 0.4084x(n-3) + 0.1x(n-4). \quad (5.71)$$

The output of the memoryless nonlinear function is given by

$$r(n) = q(n) + 0.2q^2(n) + 0.1q^3(n) + v(n). \quad (5.72)$$

For this scenario, we compare the conventional, the Bayesian, and the proposed DFEs. From the equations shown above, we can see that $L_c = 5$. We set $L_e = 6$, $P = 2$, $D = 6$, $Q = 0$ and $S = 0$. The decision feedback vector is then $\hat{\mathbf{x}}_4(n) = [\hat{x}(n-7), \hat{x}(n-8), \hat{x}(n-9)]^T$. For the proposed equalizer, we let $M = 4$, $M = 8$ and $M = 16$, in which $\mathbf{x}_2(n) = [x(n-6), x(n-5)]^T$, $\mathbf{x}_2(n) = [x(n-6), x(n-5), x(n-4)]^T$ and $\mathbf{x}_2(n) = [x(n-6), x(n-5), x(n-4), x(n-3)]^T$, respectively. The results are shown in Fig. 5.1 and Fig. 5.2.

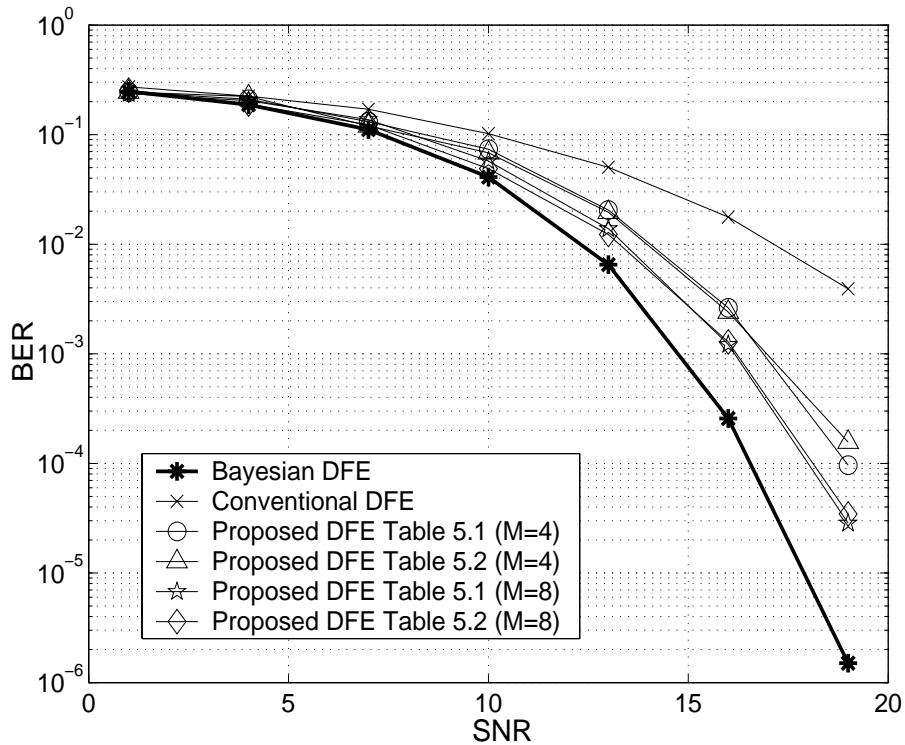


Figure 5.1: BER comparison for the conventional, proposed (Table 5.1 and Table 5.2), and Bayesian DFEs.

Both figures show that all equalizers performed similarly as that in the linear channels. From Fig. 5.1, we can see that the reduced storage method in Table 5.2 gives almost the same performance as the method in Table 5.1. As mentioned, the reduced storage method will have performance loss in nonlinear channels. The result here show that the performance loss is almost negligible. Also, from Fig. 5.2, we found that the fast convergent algorithm in Table 5.3 performs somewhat worse than that in Table 5.1. This is because the difference vectors estimated using the clustering method is not accurate

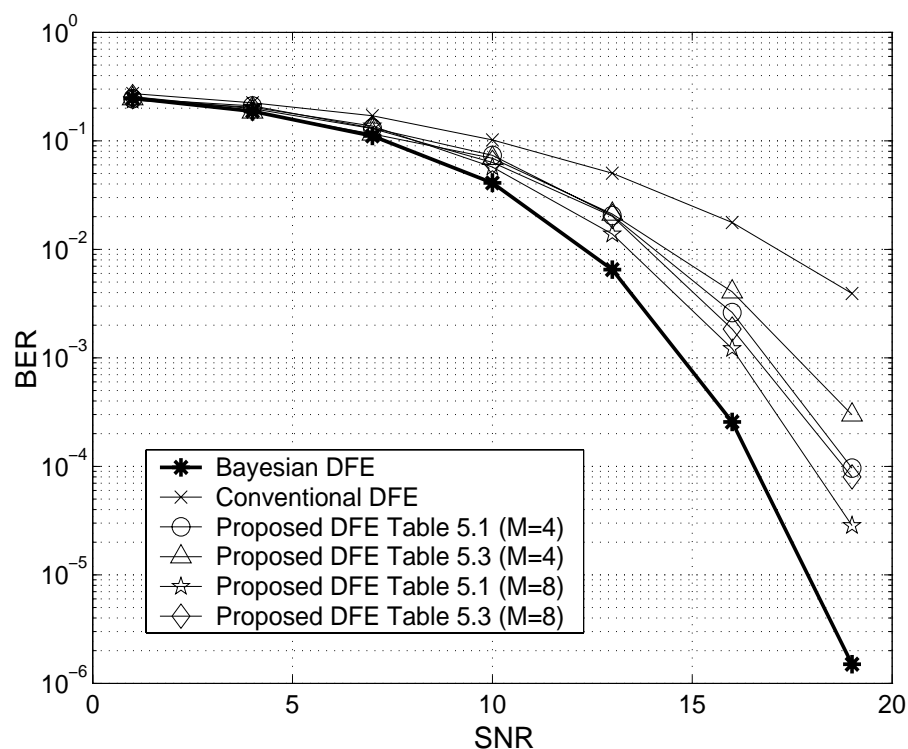


Figure 5.2: BER comparison for the conventional, proposed (Table 5.1 and Table 5.3), and Bayesian DFEs.

enough. This problem can be alleviated by increasing the training period for the clustering algorithm. The learning curves for the algorithms in Table 5.1, 5.2 and 5.3 are depicted in Fig. 5.3 (This set of simulations was run at SNR= 16dB and $M = 8$).

It is clear that the fast convergent algorithm in Table 5.3 does have the fastest convergence rate. Also note that the convergence of the reduced storage algorithm in Table 5.2 is slower than that of the algorithm in Table 5.1. This is the price to pay for storage reduction. Finally, we show the computational complexity and storage comparison for all equalizers in Table 5.6 for $M = 8$. From these tables and Fig. 5.1, Fig. 5.2, it is apparent that while the performance of the proposed DFEs (Table 5.2 and 5.3) are comparable to that of the Bayesian DFE, the computational complexity and the storage requirement for the proposed DFEs algorithm is significantly lower.

We then consider the nonlinear channel used previously. The input-output relationship of the linear channel was given by

$$q(n) = -0.227x(n) + 0.460x(n-1) + 0.848x(n-2) + 0.460x(n-3) - 0.227x(n-4). \quad (5.73)$$

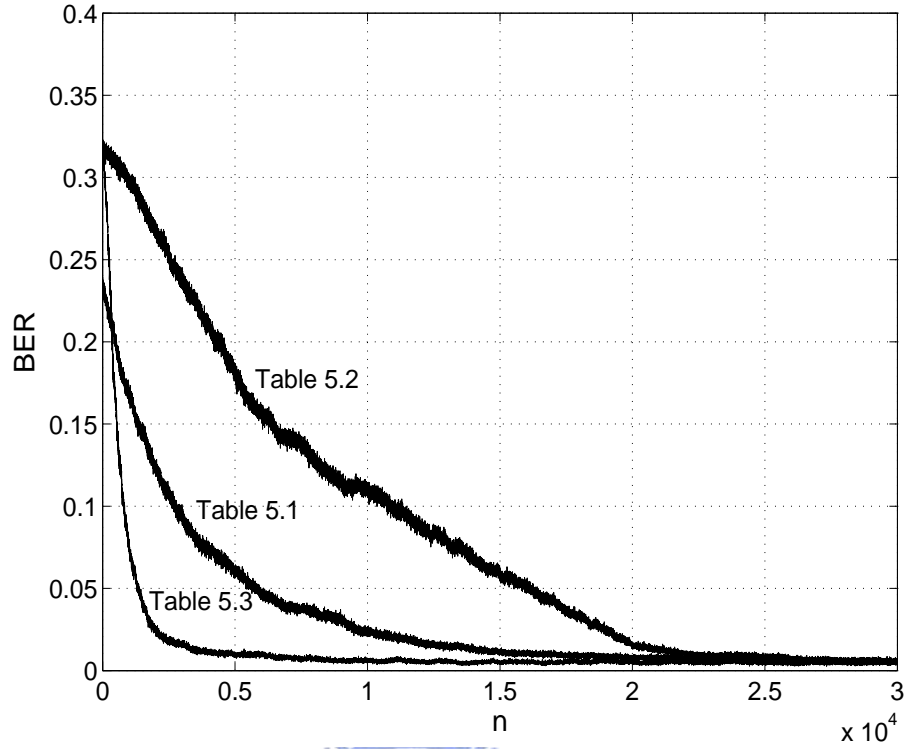
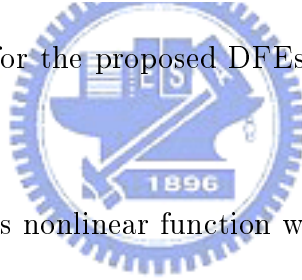


Figure 5.3: Learning curves for the proposed DFEs in Table 5.1, 5.2 and 5.3 (SNR= 16 dB, $M = 8$).



The output of the memoryless nonlinear function was given by

$$r(n) = q(n) + 0.156q^2(n) - 0.031q^3(n) + v(n). \quad (5.74)$$

From the equations shown above, we can see that $L_c = 5$. We set $L_e = 6$, $P = 2$, $D = 6$, $Q = 0$ and $S = 0$. The decision feedback vector is then $\hat{\mathbf{x}}_4(n) = [\hat{x}(n-7), \hat{x}(n-8), \hat{x}(n-9)]^T$. For the proposed equalizer, we let $M = 4$, $M = 8$ and $M = 16$, in which $\mathbf{x}_2(n) = [x(n-6), x(n-5)]^T$, $\mathbf{x}_2(n) = [x(n-6), x(n-5), x(n-4)]^T$ and $\mathbf{x}_2(n) = [x(n-6), x(n-5), x(n-4), x(n-3)]^T$, respectively.

The BER performance results are shown in 5.4 and 5.5. We can see that the performance degradation for the algorithms in Table 5.2 and 5.3 becomes slightly larger.

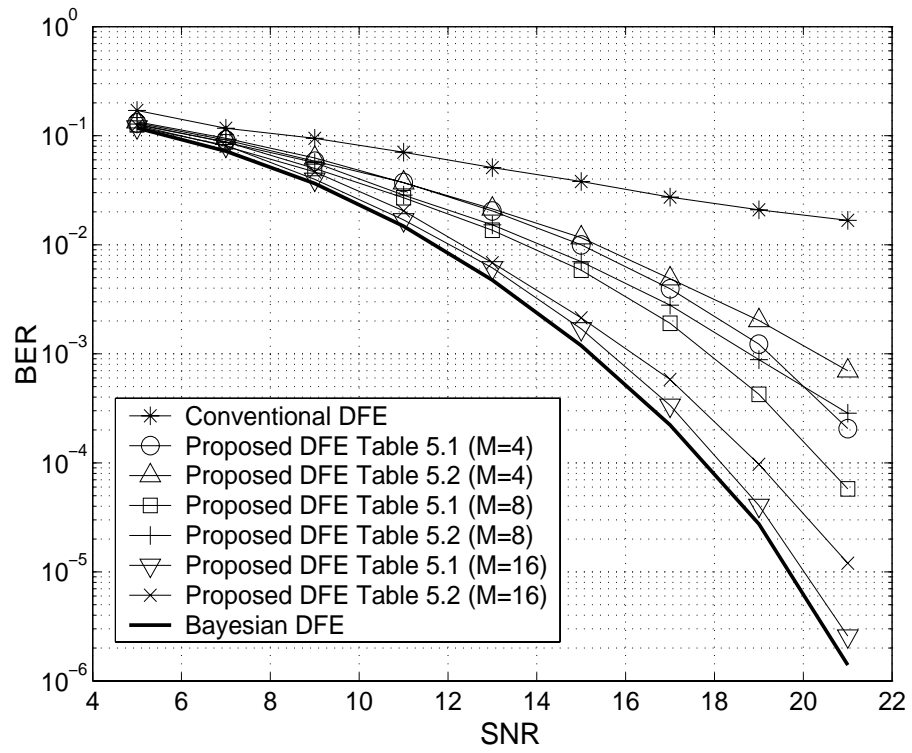


Figure 5.4: BER comparison for the proposed DFEs in Table 5.1 and 5.2 (SNR= 16 dB, $M = 8$).

Table 5.2: The adaptive algorithm for the proposed reduced storage DFE.

-
1. Initialize variables: $n=0$,
 $\mathbf{w}_{i,1}(0) = \mathbf{0}$, $b_{i,j}(0) = 0$ and $\mathbf{d}_j = \mathbf{0}$ for $i = 1, \dots, M$ and $j = 1, \dots, N_b$
 2. Clustering method (find the initial value), $n = 1, 2, \dots$
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 if $j = 1$
 $\mathbf{w}_{i,1}(n+1) = \text{counter}_{i,1} \times \mathbf{w}_{i,1}(n) + \mathbf{r}(n)$,
 $\text{counter}_{i,1} = \text{counter}_{i,1} + 1$,
 $\mathbf{w}_{i,1}(n+1) = \mathbf{w}_{i,1}(n+1) / \text{counter}_{i,1}$,
 $b_{i,1}(n+1) = -\frac{\|\mathbf{w}_{i,1}(n+1)\|^2}{2}$,
 else
 $\text{tmp} = \text{counter}_{i,j}(n) \times (\mathbf{w}_{i,1}(n) + \mathbf{d}_j(n)) + \mathbf{r}(n)$,
 $\text{counter}_{i,j} = \text{counter}_{i,j} + 1$,
 $\mathbf{d}_j(n+1) = \text{tmp} / \text{counter}_{i,j} - \mathbf{w}_{i,1}(n)$.
 end
 end
 3. Adaptive algorithm
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 for $k = 1 : M$
 $g_{k,j}(n) = \phi((\mathbf{w}_{k,1}(n) + \mathbf{d}_j(n))^T \mathbf{r}(n) + b_{k,j}(n))$,
 end
 $k = \arg \max_{l \neq i} g_{l,j}(n)$
 $L_{i,j}(n) = \frac{1}{1 + e^{-\varepsilon(-g_{i,j}(n) + g_{k,j}(n))}}$
 if $j = 1$
 $\mathbf{w}_{i,1}(n+1) = \mathbf{w}_{i,1}(n) + \mu L_{i,1}(n)(1 - L_{i,1}(n))g_{i,1}(n)(1 - g_{i,1}(n))\mathbf{r}(n)$,
 $b_{i,1}(n+1) = b_{i,1}(n) + \mu L_{i,1}(n)(1 - L_{i,1}(n))g_{i,1}(n)(1 - g_{i,1}(n)) \cdot 1$,
 $\mathbf{w}_{k,1}(n+1) = \mathbf{w}_{k,1}(n) - \mu L_{i,1}(n)(1 - L_{i,1}(n))g_{k,1}(n)(1 - g_{k,1}(n))\mathbf{r}(n)$,
 $b_{k,1}(n+1) = b_{k,1}(n) - \mu L_{i,1}(n)(1 - L_{i,1}(n))g_{k,1}(n)(1 - g_{k,1}(n)) \cdot 1$,
 else
 $\mathbf{d}_j(n+1) = \mathbf{d}_j(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))(g_{i,j}(n)(1 - g_{i,j}(n)) - g_{k,j}(n)(1 - g_{k,j}(n)))\mathbf{r}(n)$
 $b_{i,j}(n+1) = b_{i,j}(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,j}(n)(1 - g_{i,j}(n)) \cdot 1$,
 $b_{k,j}(n+1) = b_{k,j}(n) - \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{k,j}(n)(1 - g_{k,j}(n)) \cdot 1$,
 end

Table 5.3: The adaptive algorithm for the proposed fast convergent DFE.

1. Initialize variables: $n=0$,
 $\mathbf{w}_{i,1}(0) = \mathbf{0}$, $b_{i,j}(0) = 0$ and $\mathbf{d}_j(0) = \mathbf{0}$ for $i = 1, \dots, M$ and $j = 1, \dots, N_b$
2. Clustering method (find the initial value), $n = 1, 2, \dots$
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 if $j = 1$
 $\mathbf{w}_{i,1}(n+1) = \text{counter}_{i,1} \times \mathbf{w}_{i,1}(n) + \mathbf{r}(n)$,
 $\text{counter}_{i,1} = \text{counter}_{i,1} + 1$,
 $\mathbf{w}_{i,1}(n+1) = \mathbf{w}_{i,1}(n+1) / \text{counter}_{i,1}$,
 $b_{i,1}(n+1) = -\frac{\|\mathbf{w}_{i,1}(n+1)\|^2}{2}$,
 else
 $\text{tmp} = \text{counter}_{i,j} \times (\mathbf{w}_{i,1}(n) + \mathbf{d}_j(n)) + \mathbf{r}(n)$,
 $\text{counter}_{i,j} = \text{counter}_{i,j} + 1$,
 $\mathbf{d}_j(n+1) = \text{tmp} / \text{counter}_{i,j} - \mathbf{w}_{i,1}(n)$.
 end
 end
 end
3. Adaptive algorithm
 if $\mathbf{x}_{c,2}(n) = \mathbf{x}_i$ and $\hat{\mathbf{x}}_b(n) = \hat{\mathbf{x}}_j$
 for $k = 1 : M$
 $g_{k,j}(n) = \phi((\mathbf{w}_{k,1}(n) + \mathbf{d}_j(n))^T \mathbf{r}(n) + b_{k,j}(n))$,
 end
 $k = \arg \max_{l \neq i} g_{l,j}(n)$
 $L_{i,j}(n) = \frac{1}{1 + e^{-\varepsilon(-g_{i,j}(n) + g_{k,j}(n))}}$
 $\mathbf{w}_{i,j}(n+1) = \mathbf{w}_{i,j}(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,j}(n)(1 - g_{i,j}(n))\mathbf{r}(n)$,
 $b_{i,j}(n+1) = b_{i,j}(n) + \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{i,j}(n)(1 - g_{i,j}(n)) \cdot 1$,
 $\mathbf{w}_{k,j}(n+1) = \mathbf{w}_{k,j}(n) - \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{k,j}(n)(1 - g_{k,j}(n))\mathbf{r}(n)$,
 $b_{k,j}(n+1) = b_{k,j}(n) - \mu L_{i,j}(n)(1 - L_{i,j}(n))g_{k,j}(n)(1 - g_{k,j}(n)) \cdot 1$,
 end
 end
 end

Table 5.4: Computational complexity comparison for the DFE and proposed equalizers with decision feedback signal in the training phase.

	Proposed		DFE
	Initial	Adaptive	
Multiplications	L_e	$M \times (L_e + 1) + 2L_e + 10$	$2 \times (L_e + L_b)$
Additions	L_e	$M \times (L_e + 1) + 2L_e + 8$	$2 \times (L_e + L_b + 1)$
Others		$M + 1 \times \exp(\cdot)$	

Table 5.5: Computational complexity comparison for the linear, proposed, and Bayesian DFE in the decision phase.

	Bayesian DFE	Proposed DFE	DFE
Multiplications	$N_d \times (L_e + 1)$	ML_e	L_e
Additions	$2L_e N_d - 1$	ML_e	$L_e - 1 + L_b$
Others	$N_d \times \exp(\cdot)$	Compare logics	

Table 5.6: Storage comparison for the linear, proposed, and Bayesian DFEs ($M=8$).

	Bayesian DFE	Proposed DFE	Proposed DFE (reduced)	DFE
No. coefficients	6144	448	154	9

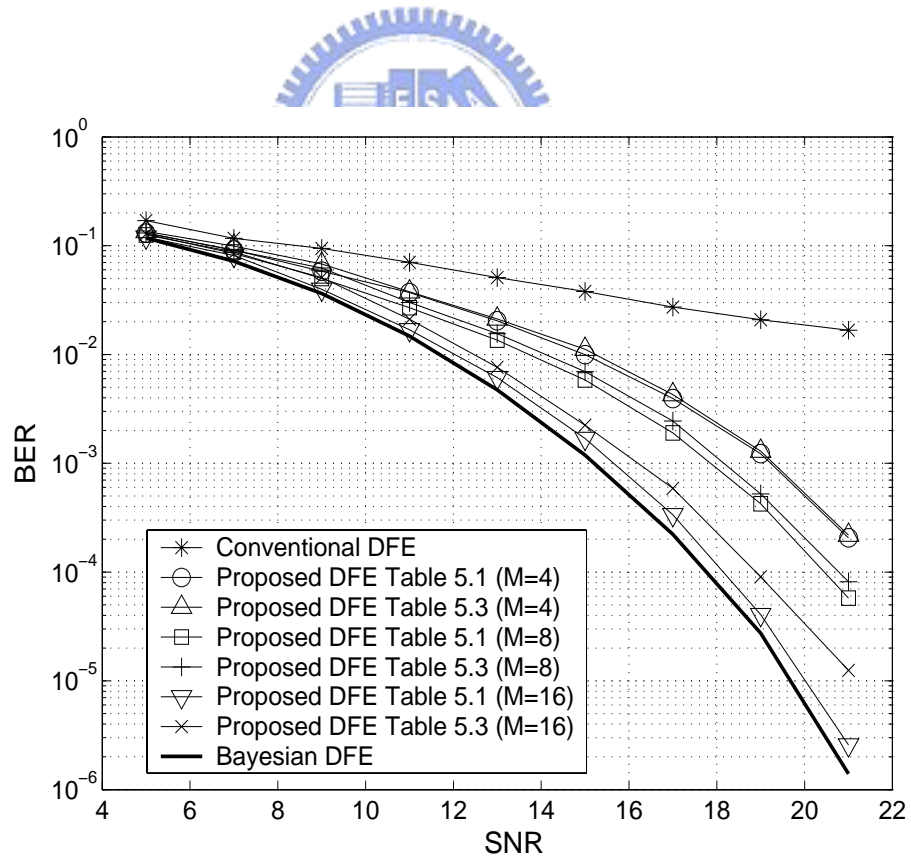


Figure 5.5: BER comparison for the proposed DFEs in Table 5.1 and 5.3 (SNR= 16 dB, $M = 8$).

Chapter 6

Adaptive Nonlinear Spatial and Temporal Equalization

In this chapter, we consider antenna array communication system. Specially, we consider the single-input-multiple-output (SIMO) systems. This kind of system has a transit antenna in the transmitter side, and an antenna array in the receiver side. Thus, we can perform spatial and temporal equalization in the receiver [46]–[50]. The main advantage of additional spatial processing is its capability to reject interference from other transmitted antennas. For this application, three kinds of equalization algorithms have been developed. The first one is the MLSE equalizer [62]–[63]. Similar to its temporal counterpart, this type of equalizer has the best performance and requires the highest computational complexity. The second one is the broad-band beamformer [64]. In this method, each antenna is followed by a temporal filter. In other words, if we have U array elements, there are U temporal filters. Adaptive algorithms such as the LMS, the recursive least square (RLS) and the constant modulus algorithm (CMA) [65] have been proposed to derive the filter weights. The last type of equalizers is the cascade of a narrow-band beamformer and a temporal filter [48]. For this type of equalization, spatial and temporal processing are performed separately.

We mainly focus on the second and third type of the spatial-temporal equalizers. Existing algorithms [48], [64] use the MMSE criterion to identify equalizer coefficients. From the discussion in previous chapters, we know that the coefficients computed using the MMSE criterion does not achieve the MBER performance. In this chapter, we treat the equalization problem as a classification problem and apply the Bayesian equalizer

to obtain optimal results. As mentioned, the Bayesian equalizer requires high computational complexity. Thus, many nonlinear equalization algorithms were developed to approximate the Bayesian equalizer (as described in Chapter 3). For reasons discussed previously, these algorithms are either computationally not efficient or not suitable for time-varying environments. The proposed algorithms described in Chapter 4 and 5 can be easily extended to SIMO systems. In this chapter, we detail the extension and show the performance enhancement of this approach.

6.1 Spatial and Temporal Channel Model

A discrete-time SIMO multipath channel was shown in Fig. 6.1. The number of antennas

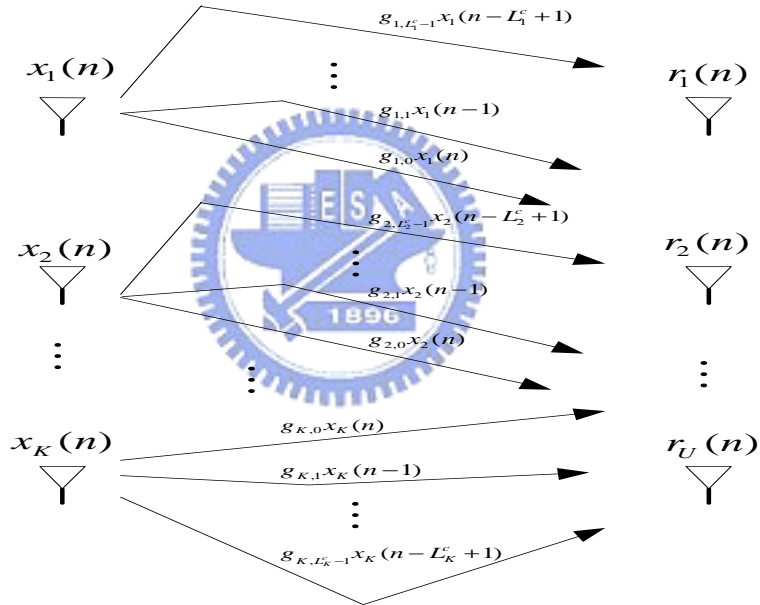


Figure 6.1: Discrete-time spatial and temporal channel model.

at the transmit and receive sides are denoted as K and U , respectively. Note here that each user only uses one transmit antenna. In other words, there are K users transmit signals simultaneously. For the desired user, there are $K - 1$ interferences. Let the transmitted signal at k th transmitted antenna and the received signal at u th received antenna be denoted as $x_k(n)$ and $r_u(n)$, respectively. Without loss of generality, we deem the signal from the first antenna is the desired signal and signals from other antenna are considered as interferences. Also, we assume that the channel between $x_k(n)$ and $r_u(n)$

is a multi-path channel. Our spatial and temporal channel model is shown in Fig. 6.1. Let $\varphi_{k,l}$ be the phase difference between the received signals at adjacent antenna elements (for the l -th channel path of the k -th user). Then,

$$\varphi_{k,l} = \frac{2\pi s \sin \theta_{k,l}}{\lambda}, \quad (6.1)$$

where λ , s , and $\theta_{k,l}$ are the wave-length of the transmitted signal $x_k(n)$, the distance between adjacent antenna elements, and the direction-of-arrival (DoA) of the received signal $r_u(n)$, respectively. Also, let $h_{u,k,l}$ be the l -path response for the k -th user at the u -th received antenna be

$$h_{u,k,l} = g_{k,l} e^{i(u-1)\varphi_{k,l}}, \quad (6.2)$$

where $g_{k,l}$ denotes the complex path gain in Fig. 6.1. We can then denote the multi-path channel for the k -th user observed by the u -th received antenna as

$$\mathbf{h}_{u,k} = [h_{u,k,0}, h_{u,k,1}, \dots, h_{u,k,L_k^c-1}]^T, \quad (6.3)$$

where L_k^c denotes the number of channel paths for the k th user signal. Let

$$\mathbf{h}_u = [\mathbf{h}_{u,1}^T, \mathbf{h}_{u,2}^T, \dots, \mathbf{h}_{u,K}^T]^T, \quad (6.4)$$

$$\mathbf{x}_k(n) = [x_k(n), x_k(n-1), \dots, x_k(n-L_k^c+1)]^T, \quad (6.5)$$

and

$$\mathbf{x}'(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \dots, \mathbf{x}_K^T(n)]^T. \quad (6.6)$$

We can then represent the signal received in the k -th antenna as

$$r_u(n) = \mathbf{h}_u^H \mathbf{x}'(n) + v_u(n), \quad (6.7)$$

where $v_u(n)$ is an AWGN with zero mean and variance σ_v^2 .

6.2 Adaptive Nonlinear Spatial-Temporal Equalization for Structure I

As mentioned, we consider two types of spatial-temporal equalization algorithm. We name the broad-band beamformer as Structure I, which is shown in Fig. 6.2, [49]–[50].

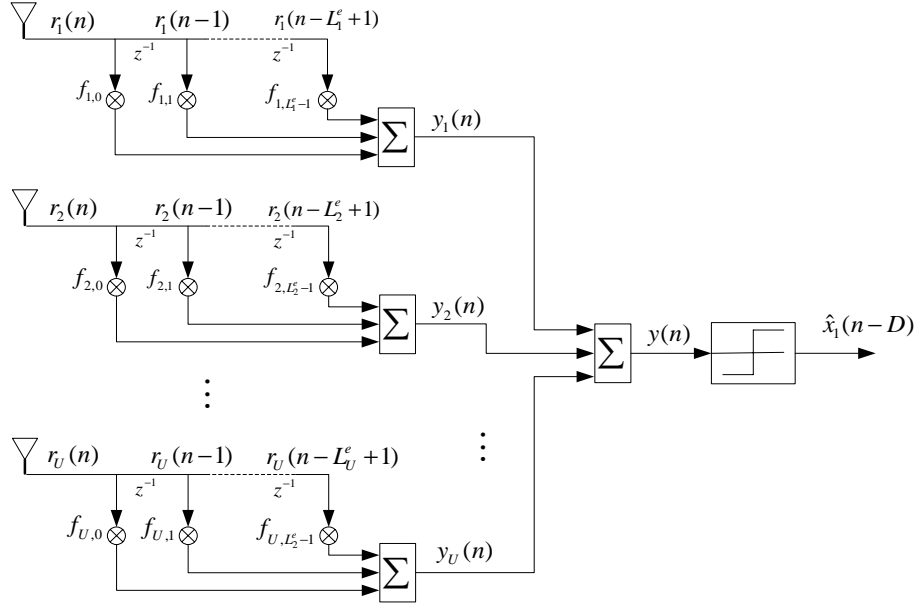


Figure 6.2: Structure I spatial-temporal equalizer.

As shown in the figure, $x_1(n - D)$ is the desired signal and a linear equalizer (with length L_u^e) follows each antenna element. Thus, we have U equalizers in total. For simplicity, we consider the BPSK transmit signal and the real channel. Thus, we can re-write $h_{u,k,l}$ as

$$h_{u,k,l} = g_{k,l} \cos((u - 1)\varphi_{k,l}). \quad (6.8)$$

Let the weight and the input vectors of the u -th equalizer be

$$\mathbf{f}_u = [f_{u,1}, f_{u,2}, \dots, f_{u,L_u^e-1}]^T, \quad (6.9)$$

and

$$\mathbf{r}_u(n) = [r_u(n), r_u(n - 1), \dots, r_u(n - L_u^e + 1)]^T. \quad (6.10)$$

Also, collect all the weight and input vectors into matrices

$$\mathbf{f} = [\mathbf{f}_1^T, \mathbf{f}_2^T, \dots, \mathbf{f}_U^T]^T, \quad (6.11)$$

and

$$\mathbf{r}(n) = [\mathbf{r}_1^T(n), \mathbf{r}_2^T(n), \dots, \mathbf{r}_U^T(n)]^T. \quad (6.12)$$

We can then have the output of the spatio-temporal equalizer as

$$y(n) = \mathbf{f}^T \mathbf{r}(n), \quad (6.13)$$

Conventional approaches [49]–[50] use the linear structure and the MMSE criterion to perform equalization. To reduce computational complexity, the adaptive algorithms such as the LMS, the recursive least squares (RLS), and the constant modulus algorithm (CMA) [65] have been proposed to use in this application. In many scenarios, the performance of the linear spatial-temporal equalizer is not satisfactory. Nonlinear equalizers are then required.

The received signal matrix $\mathbf{r}(n)$ defined in (6.12) is what we observe and we will use it to estimate $\mathbf{x}_1(n - D)$. Let $\mathbf{s}(n)$ be the received signal matrix $\mathbf{r}(n)$ when noise is absent. Let

$$\bar{\mathbf{x}}_k(n) = [x_k(n), x_k(n - 1), \dots, x_k(n - L_k^c - L_m + 2)]^T, \quad (6.14)$$

where $L_m = \max\{L_1^e, L_2^e, \dots, L_U^e\}$, and

$$\mathbf{x}(n) = [\bar{\mathbf{x}}_1(n), \bar{\mathbf{x}}_2(n), \dots, \bar{\mathbf{x}}_K(n)]^T, \quad (6.15)$$

From (6.3)–(6.7), we know that $\mathbf{s}(n)$ can be obtained by a mapping from $\mathbf{x}(n)$, i.e.,

$$\mathbf{s}(n) = \mathbf{h}(\mathbf{x}(n)), \quad (6.16)$$

where $\mathbf{h}(\cdot)$ is a matrix mapping function.

Since the number of elements in $\mathbf{x}(n)$ is given by

$$L_a = \sum_{k=1}^K (L_k^c + L_m - 1), \quad (6.17)$$

the possible vector values for $\mathbf{s}(n)$ is $N_a = 2^{L_a}$. We call those values signal states and denote them as \mathbf{s}_i , $1 \leq i \leq N_a$. We set L_r be the total dimension of the received signal. Then,

$$L_r = \sum_{u=1}^U L_u^e. \quad (6.18)$$

Similar the approach in Chapter 4, we collect all the possible signal states in the signal state set:

$$\mathcal{S} \triangleq \{\mathbf{s}_i \in \mathcal{R}^{L_r}, 1 \leq i \leq N_a\}. \quad (6.19)$$

Within the set, we can define two subsets as

$$\begin{aligned} \mathcal{S}^\pm &\triangleq \{\mathbf{s}_i \in \mathcal{S} : x_1(n - D) = \pm 1\}, \\ \mathcal{S} &= \mathcal{S}^+ \cup \mathcal{S}^-. \end{aligned} \quad (6.20)$$

To facilitate description, we use a simple example to depict the signal states.

Example :

Consider the scenario such that $K = 2$, $U = 2$, $D = 0$, $L_{c,1} = 2$, $L_{c,2} = 2$, $L_{e,1} = 1$, $L_{e,2} = 1$, $[g_{1,0}, g_{1,1}] = [1, 0.5]$, $[g_{2,0}, g_{2,1}] = [0.3, 1]$ and $2\pi s/\lambda = 1$. Thus, the received signal $\mathbf{r}(n)$ can be expressed as

$$\begin{aligned} \begin{bmatrix} r_1(n) \\ r_2(n) \end{bmatrix} &= \begin{bmatrix} g_{1,0} \cos(0 \cdot \sin(\theta_{1,0})) & g_{1,1} \cos(0 \cdot \sin(\theta_{1,1})) \\ g_{1,0} \cos(1 \cdot \sin(\theta_{1,0})) & g_{1,1} \cos(1 \cdot \sin(\theta_{1,1})) \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_1(n-1) \end{bmatrix} \\ &+ \begin{bmatrix} g_{2,0} \cos(0 \cdot \sin(\theta_{2,0})) & g_{2,1} \cos(0 \cdot \sin(\theta_{2,1})) \\ g_{2,0} \cos(1 \cdot \sin(\theta_{2,0})) & g_{2,1} \cos(1 \cdot \sin(\theta_{2,1})) \end{bmatrix} \begin{bmatrix} x_2(n) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} \\ &= \mathbf{H} \begin{bmatrix} x_1(n) \\ x_1(n-1) \\ x_2(n) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix}, \end{aligned} \quad (6.21)$$

where

$$\mathbf{H} = \begin{bmatrix} g_{1,0} \cos(0 \cdot \sin(\theta_{1,0})) & g_{1,1} \cos(0 \cdot \sin(\theta_{1,1})) & g_{2,0} \cos(0 \cdot \sin(\theta_{2,0})) & g_{2,1} \cos(0 \cdot \sin(\theta_{2,1})) \\ g_{1,0} \cos(1 \cdot \sin(\theta_{1,0})) & g_{1,1} \cos(1 \cdot \sin(\theta_{1,1})) & g_{2,0} \cos(1 \cdot \sin(\theta_{2,0})) & g_{2,1} \cos(1 \cdot \sin(\theta_{2,1})) \end{bmatrix}. \quad (6.22)$$

In this example, the vector mapping function $\mathbf{h}(\cdot)$ corresponds to a linear transformation which can be represented by a matrix \mathbf{H} . From (6.21), we can know that the number of signal states are $N_a = 16$. We set DoA's as $\theta_{1,0} = 0$, $\theta_{1,1} = \pi/2$, $\theta_{2,0} = \pi/2$, and

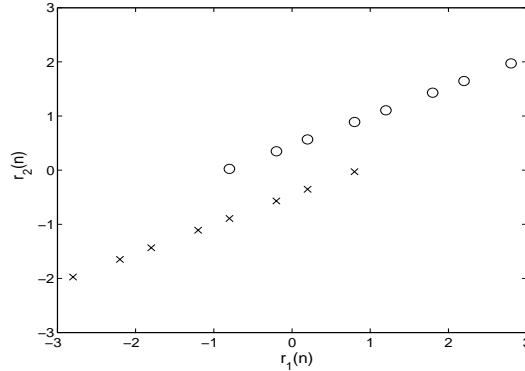


Figure 6.3: The signal states ($\theta_{1,0} = 0$, $\theta_{1,1} = \pi/2$, $\theta_{2,0} = \pi/2$, and $\theta_{2,1} = \pi/2$).

$\theta_{2,1} = \pi/2$, and show the signal states in the Fig. 6.3. The symbol 'o' denotes the signal state corresponding to $x_1(n) = +1$, and 'x' denotes the signal state corresponding to $x_1(n) = -1$. As shown in Fig. 6.3, a straight line in the $r_1(n)$ - $r_2(n)$ plane can successfully

separate the signal states in \mathcal{S}^+ and \mathcal{S}^- . This is because all the interference have the same DoA and the DoA is sufficiently different from that of the desired user. Thus, the spatial equalizer can form a null at that interfering DoA. We now show another example where the DoA of the desired user is the same as that of interfering users. Let $[g_{1,0}, g_{1,1}] = [1, 0.4]$, $[g_{2,0}, g_{2,1}] = [-0.5, 1.2]$, $\theta_{1,0} = 0$, $\theta_{1,1} = \pi/2$, $\theta_{2,0} = \pi/10$ and $\theta_{2,1} = \pi/4$. The result is shown in Fig. 6.4. From the figure, it is clear that the signal

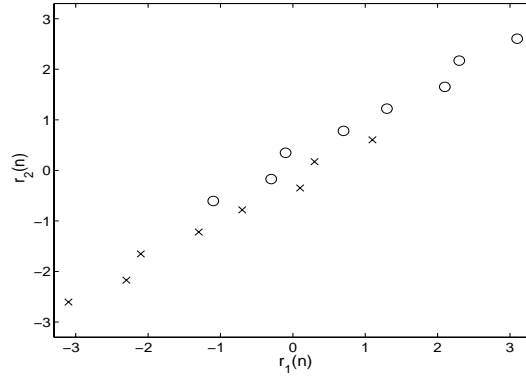


Figure 6.4: The signal states ($\theta_{1,0} = 0$, $\theta_{1,1} = \pi/2$, $\theta_{2,0} = \pi/10$ and $\theta_{2,1} = \pi/4$).

states now are not linearly separable. The conventional linear equalizer will perform poorly in this scenario. If we process the received data $r_1(n)$ and $r_2(n)$ by some nonlinear way, it is possible to separate the signal states in \mathcal{S}^+ and \mathcal{S}^- .

We then extend the methods in Chapter 5 to obtain an adaptive asymptotic spatial-temporal Bayesian equalizer. We divide the $\bar{\mathbf{x}}_1(n)$ be a combination of three vectors

$$\bar{\mathbf{x}}_1(n) = [\bar{\mathbf{x}}_{1,\alpha}^T(n), \bar{\mathbf{x}}_{1,\beta}^T(n), \bar{\mathbf{x}}_{1,\gamma}^T(n)]^T, \quad (6.23)$$

where

$$\begin{aligned} \bar{\mathbf{x}}_{1,\alpha}(n) &= [x_1(n), x_1(n-1), \dots, x_1(n-P+1)]^T, \\ \bar{\mathbf{x}}_{1,\beta}(n) &= [x_1(n-P), x_1(n-P-1), \dots, x_1(n-Q+1)]^T, \\ \bar{\mathbf{x}}_{1,\gamma}(n) &= [x_1(n-Q), x_1(n-Q-1), \dots, x_1(n-L_1^c - L_m + 2)]^T, \end{aligned} \quad (6.24)$$

where $Q > D \geq P \geq 0$ and $L_1^c + L_m - 2 \geq Q \geq 1$. Note that $\bar{\mathbf{x}}_{1,\beta}(n)$ includes $x_1(n-D)$ and its length is $Q - P$. Let $M = 2^{Q-P}$. We then have M possible vector values for

$\bar{\mathbf{x}}_{1,\beta}(n)$. Denote these vectors as \mathbf{x}_i , $i = 1, 2, \dots, M$. Now, we can divide the signal state set according to the value of $\bar{\mathbf{x}}_{1,\beta}(n)$:

$$\mathcal{S}^i \triangleq \{\mathbf{s}_i \in \mathcal{S} : \bar{\mathbf{x}}_{1,\beta}(n) = \mathbf{x}_i\}, \quad 1 \leq i \leq M, \quad (6.25)$$

and

$$\mathcal{S} = \bigcup_{1 \leq i \leq M} \mathcal{S}^i. \quad (6.26)$$

Employing the method in Chapter 5, we can use M linear discriminant functions to perform equalization. The detailed derivation is the same as the previous chapters and is omitted here. Note that the cost function (for parameter adaptation) used here is the one developed in Section 5.5. This is because the state does not consists of the desired signal only. The clustering method is less valid here. The proposed equalizer can also be extended to include decision information. Recall the method describe previously and divide $\bar{\mathbf{x}}_1(n)$ as a combination four vectors

$$\bar{\mathbf{x}}_1(n) = [\bar{\mathbf{x}}_{1,\alpha}^T(n), \bar{\mathbf{x}}_{1,\beta}^T(n), \bar{\mathbf{x}}_{1,\gamma}^T(n), \bar{\mathbf{x}}_{1,\rho}^T(n)]^T, \quad (6.27)$$

where

$$\bar{\mathbf{x}}_{1,\alpha}(n) = [x_1(n), x_1(n-1), \dots, x_1(n-P+1)]^T,$$

$$\bar{\mathbf{x}}_{1,\beta}(n) = [x_1(n-P), x_1(n-P-1), \dots, x_1(n-Q+1)]^T, \quad (6.28)$$

$$\bar{\mathbf{x}}_{1,\gamma}(n) = [x_1(n-Q), x_1(n-Q-1), \dots, x_1(n-S+1)]^T, \quad (6.29)$$

$$\bar{\mathbf{x}}_{1,\rho}(n) = [x_1(n-S), x_1(n-S-1), \dots, x_1(n-L_1^c - L_m + 2)]^T, \quad (6.30)$$

$$(6.31)$$

where $S \geq Q > D \geq P \geq 0$ and $L_c + L_e - Q - 2 \geq S \geq Q \geq 1$. As previsouly, the fourth component is obtained from feedback decisions. Let $\hat{\mathbf{x}}_1^b(n)$ be the feedback vector and decisions be correct. Then, $\bar{\mathbf{x}}_{1,\rho}(n) = \hat{\mathbf{x}}_1^b(n)$. Since the size of $\hat{\mathbf{x}}_1^b(n)$ is $L_1^b = L_1^c + L_m - S - 1$, there will be $N_1^b = 2^{L_1^b}$ possible decision patterns. As defined, we call each pattern a decision state and its possible value is $\hat{\mathbf{x}}_{1,j}^b$, $1 \leq j \leq N_1^b$. We then define $\bar{\mathbf{x}}_1(n)$ given the decision state j as

$$\bar{\mathbf{x}}_{1,j}(n) = [\bar{\mathbf{x}}_{1,\alpha,j}^T(n), \bar{\mathbf{x}}_{1,\beta,j}^T(n), \bar{\mathbf{x}}_{1,\gamma,j}^T(n), \bar{\mathbf{x}}_{1,\rho}^T(n) = \hat{\mathbf{x}}_{1,j}^{bT}]^T. \quad (6.32)$$

Note that $\bar{\mathbf{x}}_{1,\beta,j}^T(n)$ includes $x_1(n-D)$ and its length is $Q-P$. The remaining work for the equalizer design is identical to that in 5.4 and is omitted too.

We next report some simulation results to demonstrate the advantage of the proposed spatio-temporal equalizer. We consider the following scenario: $K=2$, $U=3$, $D=1$, $L_1^c=3$, $L_2^c=3$, $L_1^e=4$, $L_2^e=4$, $[g_{1,0}, g_{1,1}, g_{1,2}] = [0.3482, 0.8764, 0.3482]$, $[g_{2,0}, g_{2,1}, g_{2,2}] = [0.8, -0.2, 0.1]$, $2\pi s/\lambda = 1$. The DoAs for channel multipaths are $\theta_{1,0} = \pi/8$, $\theta_{1,1} = \pi/3$, $\theta_{1,2} = \pi/5$, $\theta_{2,0} = \pi/4$, $\theta_{2,1} = \pi/1.1$, and $\theta_{2,2} = 2\pi/3$. Let σ_s^2 be the total transmitted signal power. Then,

$$\sigma_s^2 = \sum_{k=1}^K \sigma_{s,k}^2, \quad (6.33)$$

where $\sigma_{s,k}^2$ denotes the signal power for the k th transmitted user signal. We define the SNR as the ratio of $\sigma_{s,1}^2$ and $U\sigma_v^2$. We compare the performance of the conventional linear MMSE equalizer in Fig. 6.2 and the proposed Structure I equalizer. For the proposed equalizer, we consider three cases corresponding to $M=4$, $M=8$, and $M=16$. In these cases, $\bar{\mathbf{x}}_{1,\beta}(n) = [x_1(n-1), x_1(n-2)]^T$, $\bar{\mathbf{x}}_{1,\beta}(n) = [x_1(n-1), x_1(n-2), x_1(n-3)]^T$, and $\bar{\mathbf{x}}_{1,\beta}(n) = [x_1(n-1), x_1(n-2), x_1(n-3), x_1(n-4)]^T$, respectively. The simulation results for various SNR values is shown in Fig. 6.5. Under the simulation scenario, the signal state set \mathcal{S} is not linearly separable. Thus, the performance of the linear MMSE spatial-temporal equalizer performed poorly. Even the SNR approaches to infinity, the BER does not approach to zero. The proposed spatial-temporal equalizer performed much better. The proposed equalizers with $M=8$ and $M=16$ have almost the same performance. The difference between $M=8$ and $M=4$ is small. This may due to the fact that the signal state set does not only consist of those of the desired transmitted signals. Thus, a larger M does not guarantee better performance. The performance strongly depends on the structure of all signal states. From Fig. 6.5, we can see that for the proposed equalizer, $M=4$ is sufficient.

In Fig. 6.6, we show the result for various antenna number. From the figure, it is apparent that the larger the antenna number, the better the performance (for both equalizer). When the antenna number equals to 5, the signal state set \mathcal{S} become linearly separable. However, the performance of the linear equalizer is only close to that of the

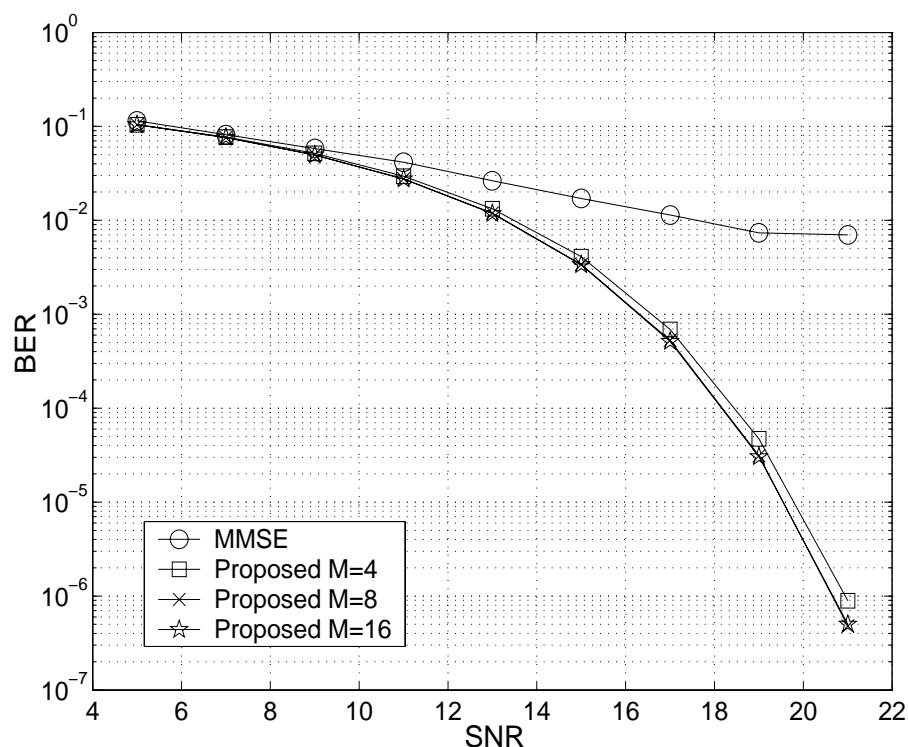


Figure 6.5: BER comparison for the linear MMSE and the proposed Structure I spatial-temporal equalizers.

proposed equalizer with 3 antenna ($M = 8$).

We also compare the conventional DFE and the proposed DFE. We use $\hat{x}_1(n-2)$ and $\hat{x}_1(n-3)$ as the feedback decisions for the conventional DFE, and use $\hat{x}_1(n-3)$ and $\hat{x}_1(n-4)$ as feedback decisions for the proposed DFE. We let $M = 4$ so that $\bar{\mathbf{x}}_{1,\beta}(n) = [x_1(n-1), x_1(n-2)]^T$. The simulation results are shown in Fig. 6.7. As we can see, the results are similar to those in the previous set of simulations.

6.3 Adaptive Nonlinear Spatial-Temporal Equalization for Structure II

Next, we discuss another spatial-temporal equalization structure shown in Fig. 6.8 [48]. In this structure, the spatial-temporal equalizer is a cascade of a narrow-band beamformer and a temporal equalizer. We refer this structure as Structure II.

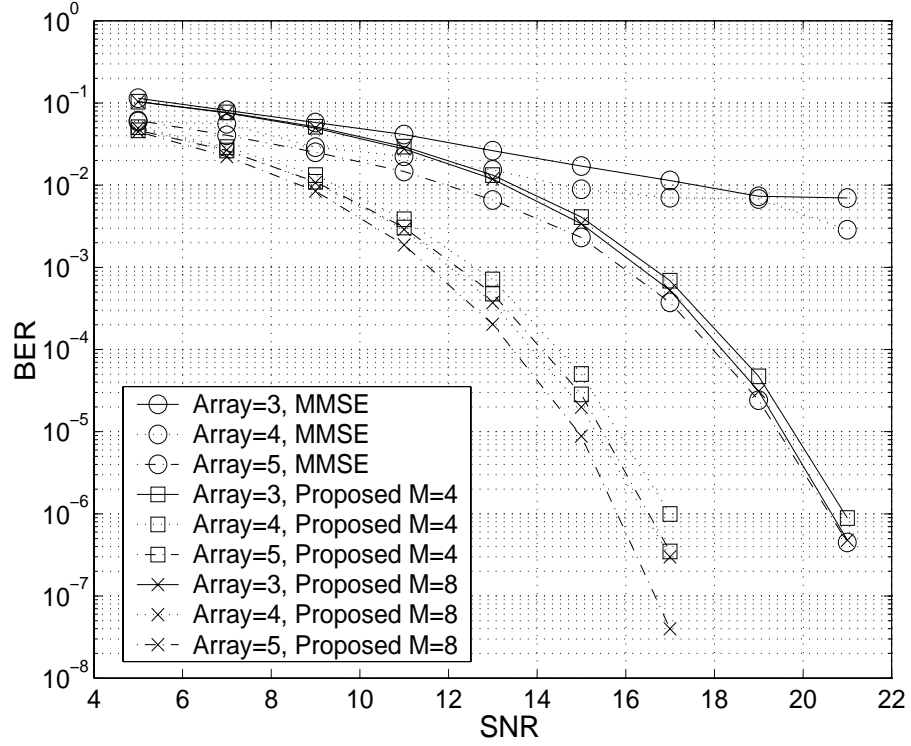


Figure 6.6: BER comparison for the MMSE linear and the proposed Structure I spatio-temporal equalizers ($U=3, 4$ and 5).

Let the weight vector of the beamformer be

$$\mathbf{w} = [w_1, w_2, \dots, w_U]^T, \quad (6.34)$$

and its input vector be

$$\mathbf{r}(n) = [r_1(n), r_2(n), \dots, r_U(n)]^T. \quad (6.35)$$

Also let the weight vector of the temporal equalizer be

$$\mathbf{f} = [f_0, f_1, \dots, f_{L_e-1}]^T. \quad (6.36)$$

where L_e is the length of the temporal equalizer. We then have the beamformer output as

$$y(n) = \mathbf{w}^T \mathbf{r}(n), \quad (6.37)$$

and the temporal equalizer output as

$$o(n) = \mathbf{f}^T \mathbf{y}(n), \quad (6.38)$$

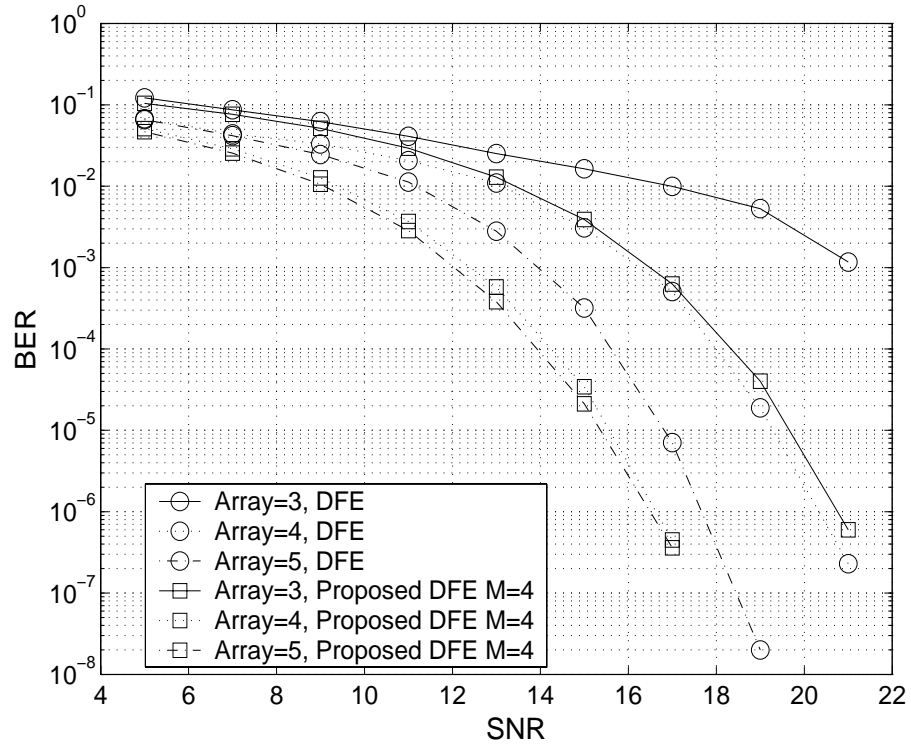
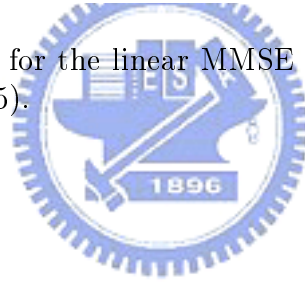


Figure 6.7: BER comparison for the linear MMSE and the proposed Structure I spatio-temporal DFE ($U=3, 4$ and 5).



where

$$\mathbf{y}(n) = [y(n), y(n-1), \dots, y(n-L^e+1)]^T. \quad (6.39)$$

We can re-write the output $o(n)$ in a compact matrix form as

$$o(n) = \bar{\mathbf{f}}^H \bar{\mathbf{r}}(n), \quad (6.40)$$

where

$$\bar{\mathbf{f}} = \mathbf{w}^T \otimes \mathbf{f}^T, \quad (6.41)$$

$$\bar{\mathbf{r}}(n) = [\mathbf{r}^T(n), \mathbf{r}^T(n-1), \dots, \mathbf{r}^T(n-L^e+1)]^T, \quad (6.42)$$

where the symbol \otimes indicates the Kronecker product operation.

It is straightforward to see that the spatio-temporal equalizer in Fig. 6.8 has lower computational complexity than that in Fig. 6.2. However, there is a price to pay for the simplicity. From (6.40), we can see that the final output is a nonlinear function of the

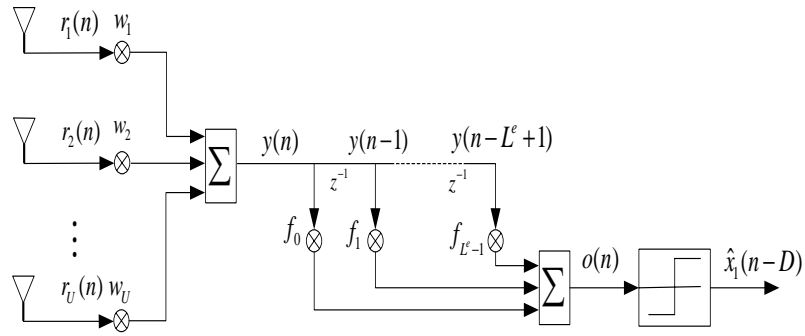


Figure 6.8: Structure II spatial-temporal equalizer.

beamformer and the equalizer weights. The MMSE cost function may exist local minima and this will adversely affect weight identification. In [48], a method was proposed to avoid the local minimum problem. Here, we do not specifically concern the local minimum problem since the method in [48] can always be employed whenever necessary. We then propose to replace the temporal equalizer in Fig. 6.8 using methods developed in Chapter 5.

Let $\mathbf{s}(n)$ denote $\mathbf{y}(n)$ when noise is absent, and

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \dots, \mathbf{x}_K^T(n)]^T, \quad (6.43)$$

where

$$\mathbf{x}_k(n) = [\mathbf{x}_k(n), \mathbf{x}_k(n-1), \dots, \mathbf{x}_k(n-L_k^c-L^e+2)]^T. \quad (6.44)$$

From Fig. 6.8, we know that $\mathbf{s}(n)$ is a mapping from $\mathbf{x}(n)$, i.e.,

$$\mathbf{s}(n) = \mathbf{h}(\mathbf{x}(n), \mathbf{w}), \quad (6.45)$$

where $\mathbf{h}(\cdot, \mathbf{w})$ is a vector mapping function. We use \mathbf{w} here to indicate that the mapping function depends on \mathbf{w} . Since the length of $\mathbf{x}(n)$ is

$$L_a = \sum_{k=1}^K (L_k^c + L^e - 1), \quad (6.46)$$

the possible values for $\mathbf{s}(n)$ is $N_a = 2^{L_a}$. Denote these values as \mathbf{s}_i , $1 \leq i \leq N_a$ and call them signal states. Using the notations in Chapter 4, we denote the signal state set as

$$\mathcal{S} \triangleq \{\mathbf{s}_i \in \mathcal{R}^{L^e}, 1 \leq i \leq N_a\}. \quad (6.47)$$

Within the set, we can define two subsets as

$$\begin{aligned}\mathcal{S}^\pm &\triangleq \{\mathbf{s}_i \in \mathcal{S} : x_1(n-D) = \pm 1\}, \\ \mathcal{S} &= \mathcal{S}^+ \cup \mathcal{S}^-.\end{aligned}\tag{6.48}$$

One thing we have to notice is that the signal states are not longer time-invariant (even the channel response is time-invariant). All the signal states will be changed whenever \mathbf{w} is changed.

We set the $\mathbf{x}_1(n)$ be a combination of three vectors

$$\mathbf{x}_1(n) = [\mathbf{x}_{1,\alpha}^T(n), \mathbf{x}_{1,\beta}^T(n), \mathbf{x}_{1,\gamma}^T(n)]^T,\tag{6.49}$$

where

$$\begin{aligned}\mathbf{x}_{1,\alpha}(n) &= [x_1(n), x_1(n-1), \dots, x_1(n-P+1)]^T, \\ \mathbf{x}_{1,\beta}(n) &= [x_1(n-P), x_1(n-P-1), \dots, x_1(n-Q+1)]^T, \\ \mathbf{x}_{1,\gamma}(n) &= [x_1(n-Q), x_1(n-Q-1), \dots, x_1(n-L_1^c - L_m + 2)]^T,\end{aligned}\tag{6.50}$$

where $Q \geq D > P \geq 0$ and $L_1^c + L_m - 2 \geq Q \geq 1$. Note that $\mathbf{x}_{1,\beta}(n)$ includes $x_1(n-D)$ and its length is $Q - P$. Let $M = 2^{Q-P}$. We then have M possible vector values for $\mathbf{x}_{1,\beta}(n)$. Denote these vectors as $\mathbf{x}_i, i = 1, 2, \dots, M$. Now, we can divide the signal state set according to the value of $\mathbf{x}_{1,\beta}(n)$:

$$\mathcal{S}^i \triangleq \{\mathbf{s}_i \in \mathcal{S} : \mathbf{x}_{1,\beta}(n) = \mathbf{x}_i\}, \quad 1 \leq i \leq M,\tag{6.51}$$

and

$$\mathcal{S} = \bigcup_{1 \leq i \leq M} \mathcal{S}^i.\tag{6.52}$$

We can further partition the space of $\mathbf{y}(n)$ into M classes and use M linear discriminant functions $f_i(\mathbf{y}(n), \Lambda_i), i = 1, \dots, M$ to perform equalization. The discriminant function $f_i(\mathbf{y}(n), \Lambda_i)$ can be expressed as

$$f_i(\mathbf{y}(n), \Lambda_i) = \mathbf{f}_i^T \mathbf{y}(n) + b_i,\tag{6.53}$$

and $\Lambda_i = [\mathbf{w}, \mathbf{f}_i, b_i]$. Note that, the parameters we have to identify are \mathbf{w}, \mathbf{f}_i , and b_i . We use the cost develop in Section 5.5 to identify discriminant functions. The advantage of

the cost function is that the adaptive algorithm yielded does not require specific initials. Define the cost function as

$$J(n) = E[L(d_i(\mathbf{y}(n)))], \quad (6.54)$$

where

$$L(d_i(\mathbf{y}(n))) = \frac{1}{1 + e^{-\varepsilon(d_i(\mathbf{y}(n)) + \alpha)}}, \quad (6.55)$$

and

$$d_i(\mathbf{y}(n)) = \frac{1}{M-1} \sum_{k \neq i} [-g_i(\mathbf{y}(n), \Lambda_i) + g_k(\mathbf{y}(n), \Lambda_k)]. \quad (6.56)$$

where

$$g_i(\mathbf{y}(n), \Lambda_i) = \phi(f_i(\mathbf{y}(n), \Lambda_i)). \quad (6.57)$$

Use the same technique in Chapter 5, we can obtain the following adaptive algorithm ($i \neq k$)

$$\mathbf{f}_i(n+1) = \mathbf{f}_i(n) + \mu L_i(n)(1 - L_i(n)) \sum_{i \neq k} g_i(n)(1 - g_i(n)) \mathbf{y}(n), \quad (6.58)$$

$$b_i(n+1) = b_i(n) + \mu L_i(n)(1 - L_i(n)) \sum_{i \neq k} g_i(n)(1 - g_i(n)) \cdot 1, \quad (6.59)$$

$$\mathbf{f}_k(n+1) = \mathbf{f}_k(n) - \mu L_i(n)(1 - L_i(n)) g_k(n)(1 - g_k(n)) \mathbf{y}(n), \quad (6.60)$$

$$b_k(n+1) = b_k(n) - \mu L_i(n)(1 - L_i(n)) g_k(n)(1 - g_k(n)) \cdot 1, \quad (6.61)$$

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu L_i(n)(1 - L_i(n)) \\ &\quad \sum_{k \neq i} (g_k(n)(1 - g_k(n)) \mathbf{R}(n) \mathbf{f}_k(n) - g_i(n)(1 - g_i(n)) \mathbf{R}(n) \mathbf{f}_i(n)), \end{aligned} \quad (6.62)$$

where

$$\mathbf{R}(n) = [\mathbf{r}(n), \mathbf{r}(n-1), \dots, \mathbf{r}(n-L^e+1)]. \quad (6.63)$$

We can extend this approach to equalization with decision feedback. Recall the method in Chapter 5 and define $\bar{\mathbf{x}}_1(n)$ be a combination four vectors

$$\bar{\mathbf{x}}_1(n) = [\bar{\mathbf{x}}_{1,\alpha}^T(n), \bar{\mathbf{x}}_{1,\beta}^T(n), \bar{\mathbf{x}}_{1,\gamma}^T(n), \bar{\mathbf{x}}_{1,\rho}^T(n)]^T, \quad (6.64)$$

where

$$\begin{aligned}\bar{\mathbf{x}}_{1,\alpha}(n) &= [x_1(n), x_1(n-1), \dots, x_1(n-P+1)]^T, \\ \bar{\mathbf{x}}_{1,\beta}(n) &= [x_1(n-P), x_1(n-P-1), \dots, x_1(n-Q+1)]^T, \end{aligned} \quad (6.65)$$

$$\bar{\mathbf{x}}_{1,\gamma}(n) = [x_1(n-Q), x_1(n-Q-1), \dots, x_1(n-S+1)]^T, \quad (6.66)$$

$$\bar{\mathbf{x}}_{1,\varrho}(n) = [x_1(n-S), x_1(n-S-1), \dots, x_1(n-L_1^c - L_m + 2)]^T, \quad (6.67)$$

$$(6.68)$$

where $S \geq Q \geq D \geq P \geq 0$ and $L_c + L_e - Q - 2 \geq S \geq Q \geq 1$. The fourth component is obtained from feedback decisions. Let $\hat{\mathbf{x}}_1^b(n)$ be the feedback vector and decisions be correct. Then, $\bar{\mathbf{x}}_{1,\varrho}(n) = \hat{\mathbf{x}}_1^b(n)$. Since the length of $\hat{\mathbf{x}}_1^b(n)$ is $L_1^b = L_1^c + L_m - S - 1$, there are $N_1^b = 2^{L_1^b}$ possible decision patterns. As defined, we call each pattern a decision state and denote its possible value as $\hat{\mathbf{x}}_{1,j}^b$, $1 \leq j \leq N_1^b$. We then define $\bar{\mathbf{x}}_1(n)$ given the decision state j as

$$\bar{\mathbf{x}}_{1,j}(n) = [\bar{\mathbf{x}}_{1,\alpha,j}^T(n), \bar{\mathbf{x}}_{1,\beta,j}^T(n), \bar{\mathbf{x}}_{1,\gamma,j}^T(n), \bar{\mathbf{x}}_{1,\varrho}^T(n) = \hat{\mathbf{x}}_{1,j}^{bT}]^T. \quad (6.69)$$

Note that $\bar{\mathbf{x}}_{1,\beta,j}^T(n)$ includes $x_1(n-D)$ and its length is $Q - P$. The remaining works for the equalizer design is identical to that in Section 5.4. We then omit the details here.

We finally use some simulation results to compare the proposed equalizer with that in Fig. 6.8. The simulation scenario is the same as that in the previous section. The result is shown in Fig. 6.9. In this set of simulations, the initial values were set to be $\mathbf{w} = [1, 0, \dots, 0]^T$ and zeros for others. From the figure, we can see that the proposed equalizer outperforms the equalizer in Fig. 6.8. We can easily find that while the computational complexity of the Structure II is lower, its performance is poorer than that in Structure I. Fig. 6.10 shows the performance comparison for these two structures (3 antennas). We also show the performance for Structure II DFEs in Fig. 6.11. We also show the performance for Structure II DFEs in Fig. 6.11. It is clear that the proposed DFE is better than the conventional DFE.

The computational complexity for the conventional and proposed spatio-temporal equalizers are summarized in Table 6.1. Comparing the equalizer with Structure I and Structure II, we can easily find that the complexity of Structure II is approximately U times less than the complexity of Structure I. However, the performance of the Structure

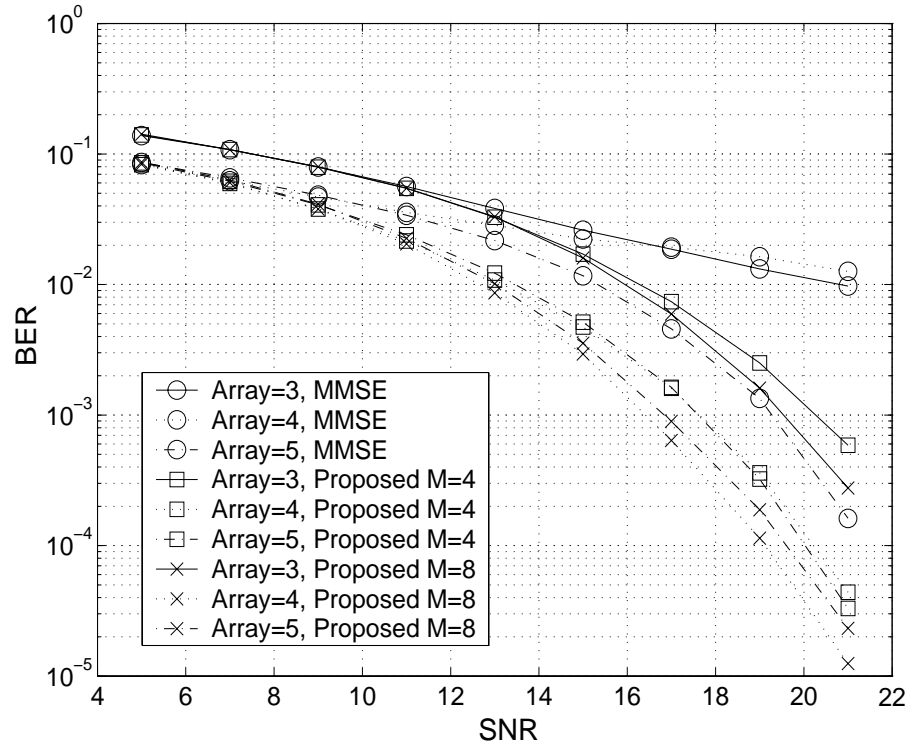


Figure 6.9: BER comparison for the linear MMSE (in Fig. 6.8) and the proposed Structure II spatio-temporal equalizers.

I is better than the Structure II as shown in Fig. 6.10. Also, Structure II may suffer from the local minimum problem. The computational complexity for the proposed equalizers is roughly M times than the conventional linear ones.

Table 6.1: Computational complexity comparison for the linear and the proposed spatio-temporal equalizers

	Structure I	Proposed for Structure I	Structure II	Proposed for Structure II
Multiplications	$\sum_{u=1}^U L_u^e$	$M \sum_{u=1}^U L_u^e$	$U + L^e$	$U + ML^e$
Additions	$\sum_{u=1}^U L_u^e - 1$	$M \sum_{u=1}^U L_u^e$	$U + L^e - 2$	$U + ML^e - 1$
Others		Compare logics		Compare logics

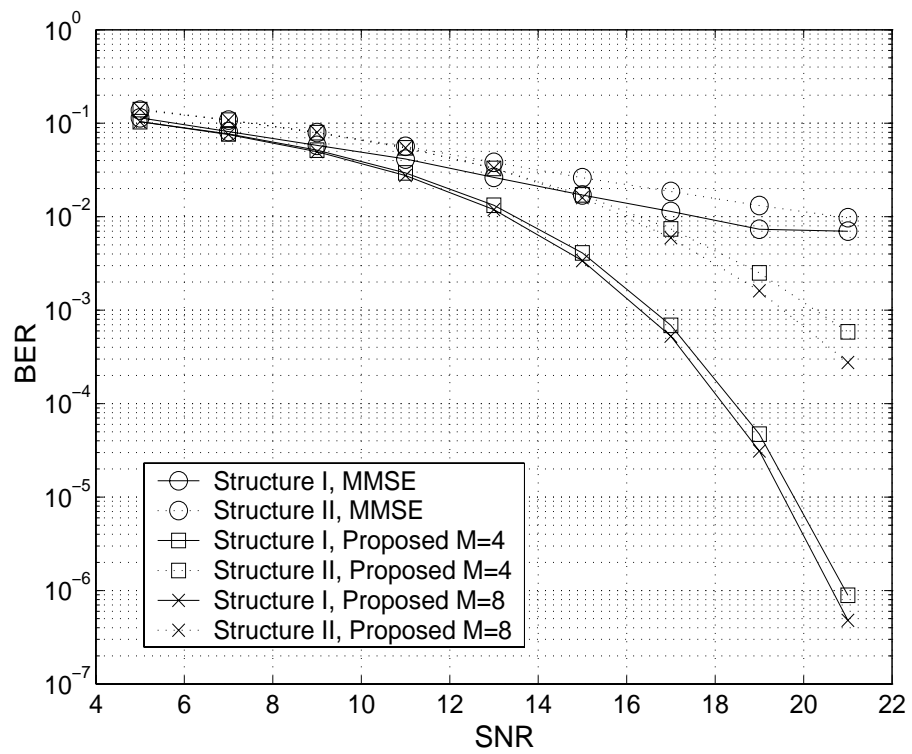


Figure 6.10: BER comparison for the linear MMSE and the proposed spatio-temporal equalizers with Structure I and II (3 antennas).

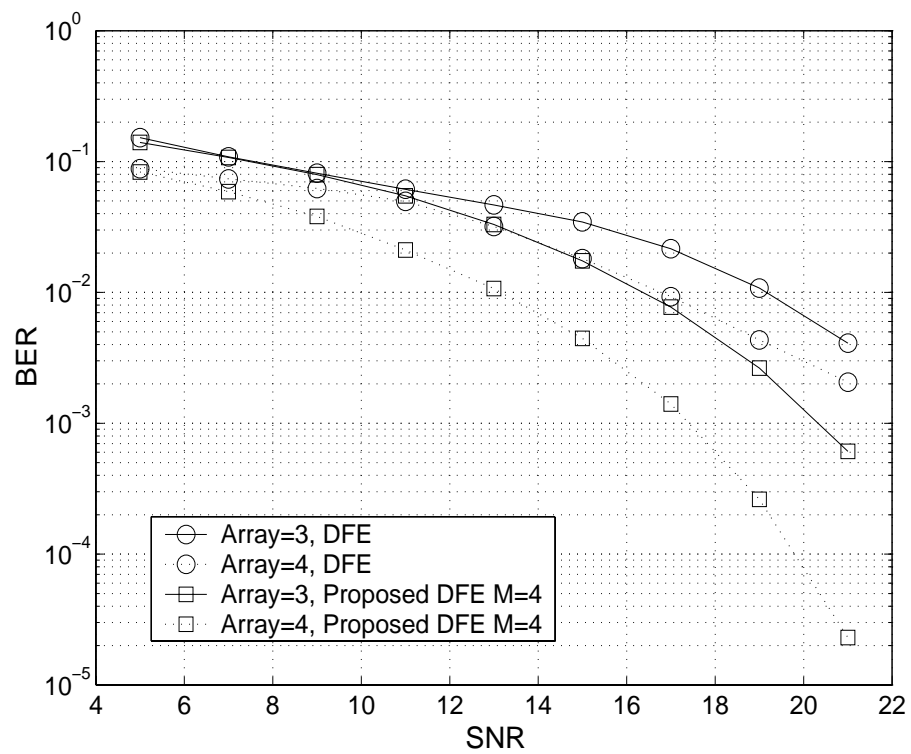


Figure 6.11: BER comparison for the linear MMSE and the proposed Structure II spatio-temporal DFEs.

Chapter 7

Maximum Likelihood Sequence Estimation for Nonlinear Channels

The MLSE equalizer is known to be the optimal sequence estimation equalizer. The computationally efficient Viterbi algorithm is commonly used to implement the MLSE equalizer. One requirement in the Viterbi algorithm is that the channel response has to be known. For linear systems, there is a unique representation for the channel response, i.e., the impulse response. However, for nonlinear systems, there is no such representation. An nonlinear channel response cannot be exactly modeled in general; only approximation is possible. Many nonlinear functions can be employed to approximate the nonlinear channel response. In [66], [67], the Volterra series were used to model nonlinear channel responses. However, the Volterra series general requires many coefficients and this results in complex channel model. And, the computational complexity in the Viterbi algorithm is then higher. Also, when the nonlinear characteristics of the channel is not well known, determination of the expansion order becomes different. In this chapter, we proposed an efficient method to solve this problem. Using this method, no channel modeling is required and the computational complexity can be lower than the approaches in [66], [67]. Also, the nonlinear effect can be exactly taken into account, not approximated.

7.1 Branch Metric Calculation

In Section 2.2, the decision rule for the MLSE can be express as

$$\hat{\mathbf{x}}_a = \arg \min_{\mathbf{x} \in \mathcal{X}} \sum_{n=0}^{L_x-1} \sum_{i=0}^{2^{L_c}-1} (r(n) - \psi_i)^2. \quad (7.1)$$

As mentioned, the Viterbi algorithm is used to implement the MLSE equalizer (7.1). In (7.1), the term $(r(n) - \psi_i)^2$ is called the branch metric, and $\sum_{n=0}^l (r(n) - \psi_i)^2$ is called the path metric up to time l . In the Viterbi algorithm, ψ_i has to be estimated.

To determine ψ_i , we need to know the input-output mapping function of the channel. Once the function is identified, $\hat{\psi}_i$'s can be estimated from the mapping from \mathcal{X} to Ψ . If the channel is linear, we usually have the channel model as

$$\hat{r}(n) = \sum_{i=0}^{L_c-1} h_i x(n-i). \quad (7.2)$$

The coefficients h_i 's can be identified through the MMSE minimization.

$$J = \min_{\mathbf{h}} E[(r(n) - \hat{r}(n))^2], \quad (7.3)$$

where $\mathbf{h} = [h_0, h_1, \dots, h_{L_c-1}]$. The minimization can be achieved either by solving the Wiener equations or using an adaptive algorithm.

If the channel is nonlinear, there is no unique representation for its response. Here, we discuss the channel modeling using the Volterra series expansion. From Chapter 3.3, we have

$$\hat{r}(n) = \vec{\mathbf{h}}^T \vec{\mathbf{x}}(n), \quad (7.4)$$

where

$$\vec{\mathbf{x}}(n) = [1, \vec{\mathbf{x}}_1^T(n), \vec{\mathbf{x}}_2^T(n), \dots, \vec{\mathbf{x}}_P^T(n)]^T, \quad (7.5)$$

and $\vec{\mathbf{h}}$ contains the kernel coefficients $h_p(m_1, m_2, \dots, m_p)$, $1 \leq p \leq P$. The coefficient vector is given by

$$\vec{\mathbf{h}} = [h_0, \vec{\mathbf{h}}_1^T, \vec{\mathbf{h}}_2^T, \dots, \vec{\mathbf{h}}_P^T]^T. \quad (7.6)$$

Note that the above notations are defined in Section 3.3. Similar to the linear scenario, we can obtain the Volterra coefficients by solving the corresponding Wiener equations or using an adaptive algorithm. The later approach is preferred in general since its computational complexity is lower.

7.2 Branch Metric Calculation Using Signal State Mapping

Determination of the Volterra series order is not a simple task. If the order is not high enough, we will have large modeling error. On the other hand, if the order is high, we will suffer from the high computational complexity. Here, we proposed a method to find all ψ_i without any channel modeling. Our idea is based on the signal state mapping. Consider an equalizer with memory size one. It is simple to see that the signal state defined in Chapter 2 is exactly the same as the state in the Viterbi algorithm. Each ψ_i corresponds to a signal state in the receiver signal space. In other words, we only have to find these signal states in order to calculate branch metrics. In other words, we find out the mapping from \mathcal{X} to Ψ directly and skip any channel modeling. Examining the received signal $r(n)$, we have

$$r(n) = \psi(\mathbf{x}'(n)) + v(n), \quad (7.7)$$

where $\mathbf{x}'(n) = [x(n), x(n-1), \dots, x(n-L_c+1)]$. From (7.7), we can see that each signal state spreads to a cluster due to noise. Let \mathbf{x}_i denote an element in \mathcal{X} . We can have

$$r_i = E[\psi(\mathbf{x}_i) + v(n)] = \psi(\mathbf{x}_i) = \psi_i, \quad (7.8)$$

where r_i is the mean of $r(n)$ when $\mathbf{x}'(n) = \mathbf{x}_i$. In other words, r_i is the signal state and is ψ_i too. From (7.8), we see that the signal state can be found using the clustering method shown below:

If a received signal $r(n)$ corresponds to the training data $\mathbf{x}'(n) = \mathbf{x}_i$

$$\hat{\psi}_i(n+1) = \text{counter}_i \times \hat{\psi}_i(n) + r(n), \quad (7.9)$$

$$\text{counter}_i = \text{counter}_i + 1, \quad (7.10)$$

$$\hat{\psi}_i(n+1) = \hat{\psi}_i(n+1) / \text{counter}_i, \quad (7.11)$$

The advantage of this algorithm is apparent. It directly finds out the mapping from \mathcal{X} to Ψ and does not require any channel modeling. Also, the estimation algorithm is very simple. If the channel is time-varying, we can introduce a forgetting factor to limit the memory size of the clustering method.

In this paragraph, we discuss the computational complexity of the Viterbi algorithm with Volterra modeling and the proposed method. Since the track back operations are the same for both algorithms, we only focus on the branch matrix calculation ψ_i .

For the Volterra model, the computational complexity in branch metric calculation depends on the number of the kernel coefficients $h_p(m_1, m_2, \dots, m_p)$, $1 \leq p \leq P$. Based on the symmetry property of the kernel, the number of independent coefficients in the p th-order kernel, N_p , is the number of distinct ways in which L_e can add up to p . Thus, N_p is given by

$$N_p = \binom{L_e + P - 1}{P} \quad (7.12)$$

Then, the coefficient number for the Volterra model with order P in (7.5) is given by

$$L_h = \sum_{p=0}^P N_p, \quad (7.13)$$

where $N_0 = 1$. For simplicity, we use the LMS algorithm to perform channel identification.

The update equations are given by

$$\hat{r}(n) = \vec{\mathbf{h}}^T(n) \vec{\mathbf{x}}(n), \quad (7.14)$$

$$e(n) = r(n) - \hat{r}(n), \quad (7.15)$$

$$\vec{\mathbf{h}}(n+1) = \vec{\mathbf{h}}(n) + \mu e(n) \vec{\mathbf{x}}(n). \quad (7.16)$$

The proposed algorithm only require the clustering method. Assuming that the channel is time-varying. We then modify the clustering method in (7.9)–(7.11) as

$$\psi_i(n+1) = (1 - \mu)\psi_i(n) + \mu(r(n) - \psi_i(n)). \quad (7.17)$$

where μ acts as a forgetting factor. Since the input signal is assumed to be BPSK, the computational requirement for the multiplication in (7.14) and (7.16) is not considered. The overall required computational complexity and storage requirement for both algorithms are summarized in Table 7.1.

From the table, we can find that the storage size for the proposed is not always less than the Volterra model. It depends on the channel length and the transmitted alphabet size. If the channel length is not particularly long, the proposed method may have less

Table 7.1: Computational complexity and storage comparison for the Volterra model and the proposed method

	Volterra	State mapping
Multiplications	1	1
Additions	$2^{L_c}(L_h - 1) + L_h + 1$	2
Storage	L_h	2^{L_c}

storage size. The main advantage of the proposed method is that it does not have to on-line compute all the required ψ_i 's and the computational complexity can be significantly reduced. This can be seen from Table 7.1.

7.3 Simulation Results

We use some simulation results to demonstrate the effectiveness of the proposed algorithm. Specifically, we consider a communication system with a high power amplifier. In this scenario, the channel is nonlinear. The input-output relationship of the nonlinear channel is given by

$$r(n) = A(q(n)) + v(n), \quad (7.18)$$

where

$$q(n) = -0.227x(n) + 0.460x(n-1) + 0.848x(n-2) + 0.460x(n-3) - 0.227x(n-4), \quad (7.19)$$

and

$$A(r) = \frac{\alpha_a r}{1 + \beta_a r^2}, \quad (7.20)$$

where $A(r)$ characterizes the AM/AM conversion of the high power amplifier, we set $\alpha_a = 1.9638$ and $\beta_a = 0.9945$ [68].

We compare the performance of the MLSE equalizer with a linear and a nonlinear channel model. For the linear model, we use 5 coefficients. The received signal can be expressed as:

$$\hat{r}_{lin}(n) = \sum_{i=0}^4 f_i x(n-i). \quad (7.21)$$

These 5 coefficients were trained by the conventional LMS algorithm with $\mu = 0.001$.

For the nonlinear model, we use the Volterra series up to 3 order. The total number of coefficients is 55. The received signal can be expressed as:

$$\hat{r}_{vol}(n) = \sum_{i=0}^4 h_i x(n-i) + \sum_{i=0}^4 \sum_{j=i}^4 h_{i,j} x(n-i)x(n-j) + \sum_{i=0}^4 \sum_{j=i}^4 \sum_{k=j}^4 h_{i,j,k} x(n-i)x(n-j)x(n-k). \quad (7.22)$$

We also train these 55 coefficients by the LMS algorithm with $\mu_1 = 0.001$, $\mu_2 = 0.0005$, and $\mu_3 = 0.0001$ for three types of coefficients in (7.22), respectively.

For the proposed method, we have 32 cluster means to estimate. Define the estimation error as

$$e(n) = \frac{1}{32} \sum_{i=0}^{31} (\psi - \hat{\psi}(n))^2. \quad (7.23)$$

We applied the clustering method shown in (7.9) and (7.11). Fig. 7.1 shows the learning curving of these three methods.

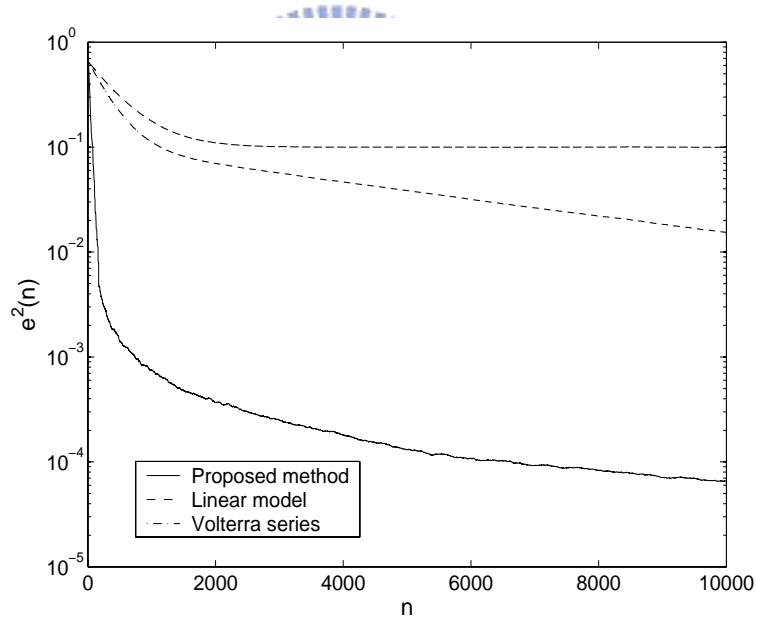


Figure 7.1: Learning curves for the linear model, the Volterra model, and the cluster means.

From the figure, we can know that the convergence for the clustering means is much faster than for Volterra coefficients. Also, the linear model is not adequate; the corresponding estimation error is large. The BER performance for these methods is shown in Fig. 7.2. The number of data used for simulations was 10^5 . After training, we can apply

the Viterbi algorithm to perform equalization. Using decisions, we can continuously train the coefficients.

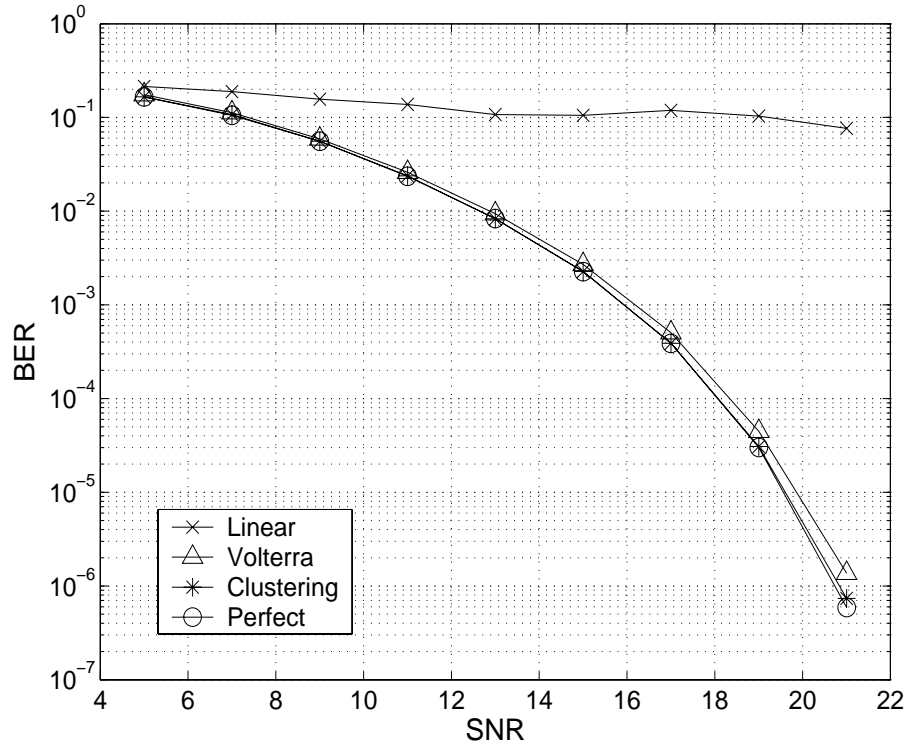


Figure 7.2: BER comparison for the Viterbi equalizer with the linear channel model, with the Volterra channel model, with the perfect channel model, and with the cluster means.

The performance for the Viterbi algorithm with the linear channel model is very poor. That is not surprising since the channel is nonlinear. The performance for the Viterbi with the Volterra channel model is close to the that with the cluster means and that with the perfect channel model. This indicates that the Volterra can model the nonlinear channel very well. We will show the complexity comparison later to show the advantage of the proposed algorithm.

We consider another nonlinear channel as follows.

$$A(r) = r + 1.2r^2 - 0.35r^3 - 0.6r^4. \quad (7.24)$$

Note that the nonlinearity is higher in this scenario. Also, there is a fourth order term which cannot be modeled by the Volterra model in (7.22). The equalization results are shown in Fig. 7.3 and Fig. 7.4.

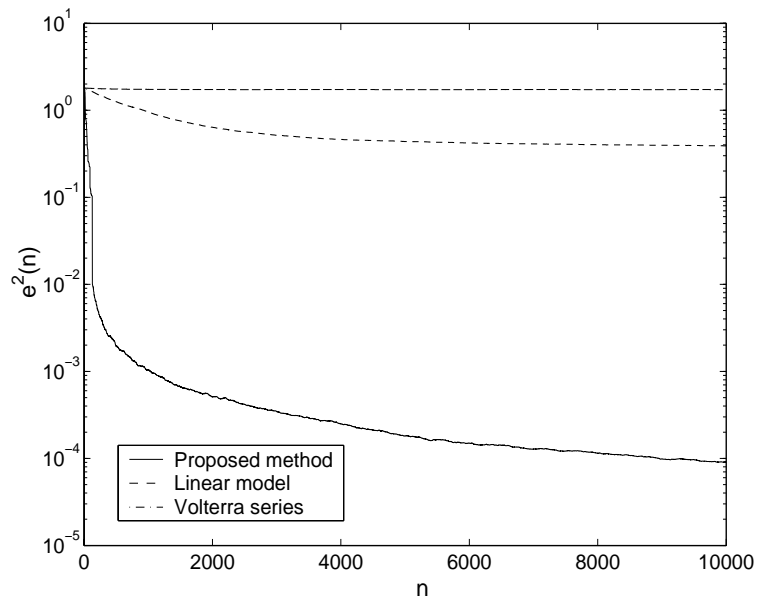


Figure 7.3: Learning curves for the linear channel model, the Volterra channel model, and the cluster means.

Finally, we show the computational complexity and storage comparison for the Viterbi equalizer with the Volterra channel model and the cluster means in Table 7.2. For the table, it is apparent that the computational requirement is much lower in the proposed approach.

Table 7.2: Computational complexity and storage comparison for the Volterra model and signal state mapping method

	Volterra	State mapping
Multiplications	1	1
Additions	1784	2
Storage	55	32

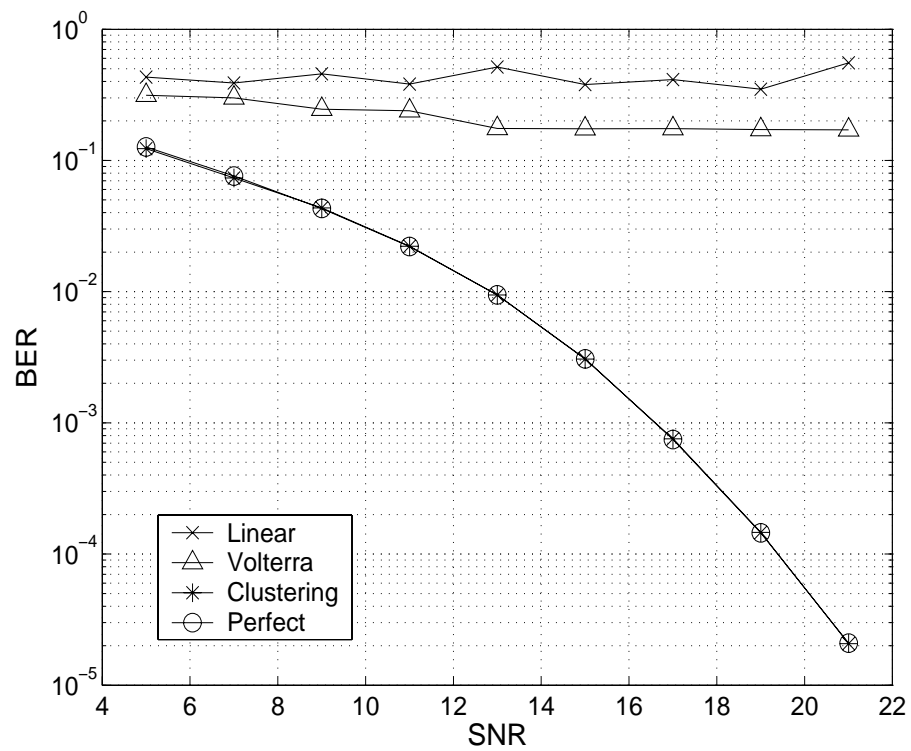


Figure 7.4: BER comparison of the Viterbi equalizer with the linear channel model, with the Volterra channel model, with the perfect channel model, and with the clustering means.

Chapter 8

Conclusion

We have treated the equalization problem as a pattern classification problem. The optimal equalizer is known to be the Bayesian equalizer. However, since its computational complexity is usually very high, real-world applications are difficult. We have proposed new adaptive nonlinear algorithms to overcome this problem. Conventionally, observations for a possible transmit symbol are mapped to a class. The main novelty in this dissertation is that we let observations for a possible transmit symbol are mapped to multiple classes. We develop efficient nonlinear equalizers which are independent of channel responses and allows an easy trade-off between performance and computational complexity. We investigate the problem from two different perspectives and found that the results are similar. The proposed equalizer consists of a set of parallel linear discriminant functions followed by a maximum function. The number of functions can be arbitrarily chosen and the corresponding coefficients can be adaptively trained. We also extend the proposed equalizer to the antenna array communication systems and this results in new nonlinear spatio-temporal equalizers. We demonstrated that while the proposed equalizers can closely approximate the performance of optimal Bayesian equalizers, their computational complexity is significantly lower. Due to its adaptive nature, the proposed equalizer are applicable to time-varying channels. The adaptive algorithm is implemented with the SGD which is simple and robust and this will be a great advantage for real-world applications. Finally, we have proposed an efficient MLSE equalizer which does not require any channel modeling. This is particularly useful for nonlinear channels.

The idea of subset partition can be further extended. For example, if we want to

have eight subsets, we may choose $x(n - D)$ and any two other elements from $\mathbf{x}_c(n)$. Using these three elements, we can partition the signal space into eight subsets. This is different from the partitioning method described in this paper where all three elements of $\mathbf{x}_{c,2}(n)$ must be consecutive. The specific elements to choose may depend on the channel mapping function. An intuitive thought is to choose the ones contributing most energy in the received signal vector. To do that, we may need a channel identification filter. This extended method may make the signal subsets more linearly separable and facilitate decision making.

One common disadvantage of the proposed equalizers is that their convergence is usually slow. This is because we have used the MBER criterion for coefficient adaptation. This cost function has a property that when the decision is right, the cost function is close to zero. As a result, when the equalizer is close to its optimum, its adaptation become slow due to the low BER. How to accelerate the convergence deserves further investigation.

Application of the proposed equalizers is not just confined in the communication systems considered here. Any communication system that require equalization can be the candidate. For example, we may consider the multiple-input-multiple output (MIMO) communication systems. The MIMO system can have a high data transmission rate; however, its receiver is considerably more complex. In this type of systems, a block DFE is usually applied to perform iterative symbol detection. We can then use the proposed equalization algorithms to replace the block DFE. This application can serve a good topic for further research.

Bibliography

- [1] S. Benedetto, E. Biglieri and R. DAFFARA, “Modeling and performance evaluation of nonlinear satellite links — A Volterra series approach,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-15, no. 4, pp. 494–506, July 1979.
- [2] S. Benedetto and E. Biglieri, “Nonlinear equalization of digital satellite channels,” *IEEE J. Sel. Areas Comm.*, vol. SAC-1, no. 1, pp. 57–62, 1983.
- [3] C. M. Melas, P. C. Arnett, I. A. Beardsley, and D. Palmer, “Nonlinear superposition in saturation recoding of disk media,” *IEEE Trans. Magn.*, vol. 23, no. 5, pp. 2079–2081, Sept. 1987.
- [4] D. Palmer and P. Zipserovich, “Identification of nonlinear write effects using pseudo random sequences,” *IEEE Trans. Magn.*, vol. 23, no. 5, pp. 2377–2379, Sept. 1987.
- [5] Newby and R. Wood, “The effects of nonlinear distortion on class IV partial response,” *IEEE Trans. Magn.*, vol. 22, no. 5, pp. 1203–1205, Nov. 1986.
- [6] R. W. Chang, J. C. Hancock, “On receiver structures for channels having memory,” *IEEE Trans. Inform. Theory*, IT-12, pp. 463–468, Oct. 1966.
- [7] K. Abend, B. Fritchman, “Statistical detection for communication channels with intersymbol interference,” *Proc. IEEE*, 58 (5), pp. 779–786, May 1970.
- [8] G. D. Forney Jr., “Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference,” *IEEE Trans. Inform. Theory*, IT-18 (3), pp. 363–378, May 1972.
- [9] S. U. H. Qureshi, “Adaptive equalization,” *Proc. IEEE*, vol. 73, pp. 1349–1387, Sept. 1985.

- [10] S. Chen, S. McLaughlin and B. Mulgrew, “Complex valued radial basis function network, Part II: Application to digital communications channel equalisation,” *Signal Process*, vol. 35, no. 1, pp. 175-188, Jan. 1994.
- [11] S. Chen, S. Mclaughlin, B. Mulgrew, and P. M. Grant, “Adaptive Bayesian decision feedback equaliser for dispersive mobile radio channels,” *IEEE Trans. Commun.*, vol. 43, no. 5, pp. 1937-1946, May 1995.
- [12] G. Karam and H. Sari, “Analysis of predistortion, equalization, and ISI cancellation techniques in digital radio systems with nonlinear transmit amplifiers,” *IEEE Trans. Comm.*, vol. 37, no. 12, pp. 1245–1253, 1989.
- [13] A. P. Clark, L. H. Lee, and R. S. Marshall, “Developments of the conventional nonlinear equaliser,” *Proc. IEE*, vol. 129, Pt. F, no. 2, pp. 85–94, Apr. 1982.
- [14] S. Im. and E. J. Powers, “Adaptive equalization of nonlinear digital satellite channels using a frequency-domain Volterra filter,” MILCOM '96, Conference Proceedings, IEEE , vol. 3, 21–24 pp. 843–848, Oct. 1996.
- [15] A. Gutierrez, and W. E. Ryan, “Performance of Volterra and MLSD receivers for nonlinear band-limited satellite systems,” *IEEE Trans. Comm.*, vol. 48, Issue 7, pp. 1171–1177, July 2000.
- [16] S. Chen, G. J. Gibson and C. F. N. Cowan, “Adaptive channel equalisation using a polynomial-perceptron structure,” *Proc. IEE*, vol. 137, Pt. I, no. 5, pp. 257–264, Oct. 1990.
- [17] J. M. Mendel, *Lessons in Digital Estimation Theory*, Prentice-Hall, Englewood-Cliffs, New Jersey, 1995.
- [18] Z. Xiang, G. Bi and T. Le-Ngoc, “Polynomial perceptrons and their applications to fading channel equalization and co-channel interference suppression,” *IEEE Trans. Signal Processing*, vol. 42, no. 9, pp. 2470–2480, Sep. 1994.

- [19] G. K. Ma, J. Lee, and V. J. Mathews, "A fast RLS bilinear filter for channel equalization," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Adelaide, Australia, pp. III-257–III-260, Apr. 1994.
- [20] J. Lee, and V. J. Mathews, "On the extend RLS adaptive bilinear filters," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Minneapolis, Minnesota, pp. III 428–431, 1993.
- [21] S. Choi and D. Hong, "Equalization using the bilinear recursive polynomial perceptron with decision feedback," *IEEE IJCNN 2000*, vol. 5, pp. 366–371, Como, Italy, July 2000.
- [22] G. J. Gibson, S. Siu and C. F. N. Cowan, "Application of multilayer perceptrons as adaptive channel equalisers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, pp. 1183-1186, 1989.
- [23] G. J. Gibson, S. Siu and C. F. N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals," *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1877-1884, Aug. 1991.
- [24] S. Chen, G. J. Gibson, C. F. N. Cowan and P. M. Grant, "Adaptive equalization of finite non-linear channels using multilayer perceptrons," *Signal Processing*, vol. 20, pp. 107–119, 1990.
- [25] S. Chen, G. J. Gibson, C. F. N. Cowan and P. M. Grant, "Decision feedback equalization using neural network structure," *IEE Int. Conf. Neural Networks*, London, 1989.
- [26] C. Z. W. H. Sweatman, B. Mulgrew and G. J. Gibson, "Two algorithms for neural-network design and training with application to channel equalization," *IEEE Trans. Neural Network*, vol. 9, no. 3, pp. 533–543, May 1998.
- [27] T. Adali, X. Liu, and M. K. Sönmez, "Conditional distribution learning with neural networks and its application to channel equalization," *IEEE Trans. Signal Processing*, vol. 45, no. 4, pp. 1051–1064, Apr. 1997.

- [28] S. Chen, B. Mulgrew, P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis function networks,” *IEEE Trans. Neural Network*, vol. 4, no. 4, pp. 571–579, July. 1993.
- [29] S. Chen, S. McLaughlin and B. Mulgrew, P. M. Grant, “Complex valued radial basis function network: application to digital communication channels equalization (part II),” *EURASIP Signal Processing Journal*, vol. 36, no. 2, pp. 175–188, Mar. 1994.
- [30] I. Cha and S. A. Kassam, “Channel equalization using adaptive complex radial basis function networks,” *IEEE J. Select. Areas Commun.*, vol. 13, no. 1, pp. 122–131, Jan. 1995.
- [31] S. Chen, G. J. Gibson, C. F. N. Cowan and P. M. Grant, “Reconstruction of binary signals using as adaptive radial basis function equaliser,” *EURASIP Signal Processing Journal*, vol. 22, no. 1, pp. 77–93, 1991.
- [32] B. Mulgrew, “Applying radial basis function,” *IEEE Signal Process. Mag.*, vol. 13, no. 2, pp. 50–65, Mar. 1996.
- [33] E. Chng, H. Yang and W. Skarbek, “Reduced complexity implementation of Bayesian equalizer using local RBF network for channel equalization problem,” *IEE Electron. Lett.*, vol. 32, no. 1, pp. 17–19, Jan. 1996.
- [34] J. Montalvão, J. C. Mota, B. Dorizzi and C. C. Cavalcante, “Reducing Bayes equalizer complexity: a new approach for clusters determination,” Proceedings of the International Telecommunication Symposium - ITS’98/IEEE, São Paulo, Brazil, pp. 428–433, Aug. 1998.
- [35] J. Montalvão, B. Dorizzi and J. C. Mota, “A family of nonlinear equalizers: sub-optimum Bayesian classifiers, Proceedings of IEEE–EURASIP Workshop on Nonlinear Signal and Image Processing — NSIP’99, Antalya, Turkey, pp. 263–267, June 1999.

- [36] D. G. Oh, J. Y. Choi and C. W. Lee, "Adaptive nonlinear equalizer with reduced computational complexity," *EURASIP Signal Processing Journal*, vol. 47, pp. 307–317, 1995.
- [37] S. Chen, S. Gunn, and C. J. Harris, "Decision feedback equalizer design using support vector machines" *Proc. Inst. Elect. Eng. Vision, Image, Signal Process.*, vol. 147, no. 3, pp. 213–219, 2000.
- [38] D. J. Sebald, and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Trans. Signal Processing.*, vol. 48, no. 11, pp. 3217–3226, Nov. 2000.
- [39] S. Chen, B. Mulgrew, and L. Hanzo, "Asymptotic Bayesian decision feedback equalizer using a set of hyperplanes," *IEEE Trans. Signal Processing*, vol. 48, no. 12, pp. 3493–3500, Dec. 2000.
- [40] Y. Kim and J. Moon, "Delay-constrained asymptotically optimal detection using signal-space partitioning," *IEE Proc. ICC'98*, Atlanta, GA, 1998.
- [41] Y. Kim and J. Moon, "Multidimensional signal space partitioning using a minimal set of hyperplanes for detecting ISI-corrupted symbols," *IEEE Trans. Commun.*, vol. 48, pp. 637–647, Apr. 2000.
- [42] S. Chen, L. Hanzo and B. Mulgrew, "Multiple hyperplane detector for implementing the asymptotic Bayesian detection feedback equalizer," *IEEE International Conference on Communications*, vol. 2, pp. 361–365, 2001.
- [43] S. Katagiri, C. H. Lee, and B. H. Juang, "Discriminative Multi-Layer Feed-Forward Networks," *IEEE-SP Workshop on NN for SP*, Princeton, Sept. 1991.
- [44] Di Claudio, E. D. Parisi, R. Orlandi, G. Orlandi, "Discriminative learning strategy for efficient neural decision feedback equalizers," *The 2000 IEEE International Symposium*, vol. 4, pp. 521–524, 2000.
- [45] B. H. Juang, and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 3043–3054, 1992.

- [46] R. Kohno, "Spatial and temporal communication theory using adaptive antenna array," *IEEE Personal Commun.*, pp. 28–35, Feb. 1998.
- [47] A. J. Paulraj and B. C. Ng, "Space-time modems for wireless personal communications," *IEEE Personal Commun.*, pp. 36–49, Feb. 1998.
- [48] K. Hayashi and S. Hara, "A new spatio-temporal equalization method based on estimated channel response," *IEEE Trans. Veh. Technol.*, vol. 50, no. 5, pp. 1250–1258, Sep. 2001.
- [49] S. N. Diggavi and A. Paulraj, "Performance of multisensor adaptive MLSE in fading channels," in *Proc. Vehicular Technology Conf.*, vol. 3, Phoenix, AZ, pp. 2148–2152, May 1997.
- [50] B. C. Ng, J. T. Chen and A. Paulraj, "Space-time processing for fast fading channels with co-channel interference," in *Proc. Vehicular Technology Conf.*, Atlanta, GA, pp. 1491–1495, Apr. 1996.
- [51] J. Montalvão, B. Dorizzi and J. C. Mota, "Why use Bayesian equalization based on finite data blocks?," *Signal Processing*, vol. 81, pp. 137–147, 2001.
- [52] M. R. Aaron and D. W. Tufts, "Intersymbol interference and error probability," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 26–34, Jan. 1966.
- [53] S. Chen, E. Chng, B. Mulgrew, and G. Gibson, "Minimum-BER linear-combiner DFE," in *Proc. Int. Conf. on Communications*, pp. 1173–1177, 1996.
- [54] C. C. Yeh and J. R. Barry, "Adaptive minimum bit-error rate equalization for binary signaling," *IEEE Trans. Commun.*, vol. 48, no. 7, pp. 1226–1235, July 2000.
- [55] V. J. Mathews and G. L. Sicuranza, *Polynomial signal processing*, Wiley.
- [56] R. R. Mohler, and W. J. Kolodziej, "An overview of bilinear system theory and applications," *IEEE Trans. Syst., Man Cybernet.*, vol. SMC-10, no. 9, pp. 683–688, Oct. 1980.

- [57] R. W. Brockett, "Volterra series and geometric control theory," *Automatica*, vol. 12, pp. 167–176, 1976.
- [58] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. E. and McClelland, J. L. MIT Press, Cambridge, MA.
- [59] V. Vapnik, "The Nature of Statistical Learning Theory." New York: Springer-Verlag, 1995.
- [60] S. Haykin, "Neural Network: A Comprehensive Foundation," 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.
- [61] S. Haykin, "*Adaptive Filter Theory*", Third Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1996.
- [62] G. E. Bottomley and K. Jamal, "Adaptive arrays and MLSE equalization", *Proc. VTC*, pp. 50–54, July 1995.
- [63] Y. Doi, T. Ohgane and E. Ogawa, "ISI and CCI canceller combining the adaptive array antennas and the Viterbi equalizer in a digital mobile radio", *Proc. VTC*, pp. 81–85, Apr. 1996.
- [64] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Mag.*, pp. 4–12, Apr. 1988.
- [65] R. T. Compton, Jr., "Adaptive Antennas: Concepts and Performance," Prentice Hall, 1998.
- [66] M. F. Mesriya, and P. J. M. Lane and L. L. Campbell, "Maximum likelihood sequence estimation of binary sequences transmitted over bandlimited nonlinear channels," *IEEE Trans. on Commun.*, vol. 25, no. 7, pp. 2286–2294, July 2002.
- [67] Y. J. Chen, "Nonlinear CDMA satellite receiver: system design and analysis." M.Sc. thesis, National Chiao Tung University, R. O. C., July 1999.

- [68] A. Gutierrez, W. E. Ryan, "Performance of adaptive Volterra equalizers on nonlinear satellite channels," *IEEE International Conference on Communications*, vol. 1, 18–22, pp. 488–492, 1995.
- [69] D. J. Sebald, and J. A. Bucklew, "A binary adaptive decision-selection equalizer for channels with nonlinear intersymbol interference," *IEEE Trans. Signal Processing*, vol. 50, no. 9, pp. 2286–2294, 2002.
- [70] H. Vincent Poor, *An Introduction to Signal Detection and Estimation*, 2nd. Springer.
- [71] R. J. Chen and W. R. Wu, "Adaptive asymptotic Bayesian equalization using a signal space partitioning technique," *IEEE Trans. Signal Processing*, vol. 52, no. 5, pp. 1376–1386, May 2004.
- [72] R. J. Chen and W. R. Wu, "Adaptive channel equalization using approximate Bayesian criterion," *IEEE, GLOBECOM '02*, vol. 1, pp. 292–296, Nov. 2002.
- [73] R. J. Chen and W. R. Wu, "Efficient neural equalization using multiple nonlinear discrimination functions," *International Conference on Communications, Internet, and Information Technology (CIIT), IASTED*, St. Thomas, US Virgin Islands, pp. 407–411, 2002.
- [74] R. J. Chen and W. R. Wu, "Efficient approach decision feedback Bayesian equalizer by using multiple linear discriminant functions," (Journal paper in preparation).