DEVELOPMENT OF A NOVEL EQUALISER FOR COMMUNICATION CHANNELS USING TABU SEARCH TECHNIQUE IN NEURAL NETWORK PARADIGM

A thesis submitted for the degree of

Master of Technology (Research)

By

K.R.SUBHASHINI

Department of Electrical Engineering

National Institute of Technology

Rourkela

2008

DEVELOPMENT OF A NOVEL EQUALISER FOR COMMUNICATION CHANNELS USING TABU SEARCH TECHNIQUE IN NEURAL NETWORK PARADIGM

A thesis submitted for the degree of

Master of Technology (Research)

By

K.R.SUBHASHINI

Under the Guidance of

Prof. J. K. Satapathy

Department of Electrical Engineering

National Institute of Technology

Rourkela

2008

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**DEVELOPMENT OF A NOVEL EQUALISER FOR COMMUNICATION CHANNELS USING TABU SEARCH TECHNIQUE IN NEURAL NETWORK PARADIGM** submitted by Ms. **K.R.Subhashini** in partial fulfillment of the requirements for the award of MASTER of Technology(RESEARCH) Degree in **Electrical Engineering** with specialization in "**Electronics System and Communication**" at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him/her under my/our supervision and guidance.

To the best of my/our knowledge, the matter embodied in the thesis has not been submitted to any other University/ Institute for the award of any degree or diploma.

Date:

Prof. J. K. Satapathy
Dept.of Electrical Engg.
National Institute of Technology
Rourkela - 769008

DEDICATED TO MY PARENTS

ACKNOWLEDGEMENT

I express my sincere gratitude and appreciation to many people who helped keep me on track toward the completion of my thesis. Firstly, I owe the biggest thanks to my supervisor, **Prof. J. K. Satapathy**, whose advice, patience, and care boosted my morale.

I am very much thankful to our HOD, **Prof. S.Ghosh**, for providing me with best facilities in the department and his timely suggestions. I express special thanks to all my colleagues and seniors who have inspired me a lot during the course of research work.

I wish to express my gratitude to my parents and husband whose love and encouragement have supported me throughout my education.

K.R.Subhashini

ACRONYMS&ABBREVATIONS

ANN	artificial neural network
AWGN	additive white gaussian noise
BER	bit error rate
BP	back propagation
FNN	feed forward neural network
DCS	digital communication system
DFE	decision feedback equalizer
DSP	digital signal processing
FIR	finite impulse response
i.i.d	independent identically distributed
IIR	infinite impulse response
ISI	inter symbol interference
LMS	least mean squares
LTE	linear transversal equalizer
MAP	maximum a-posteriori probability
MLP	multi layer perceptron
MLPDFE	multilayer perceptron decision feedback equalizer
MLSE	maximum likelihood sequence estimation
MMSE	minimum mean square error
MSE	mean square error
PAM	pulse amplitude modulation
RLS	recurrent least squares
SNR	signal to noise ratio

TDL	tapped delay line
ZF	zero forcing
TS	tabu search
TBBP	tabu based back propagation
TL	tabu list
DS	deep search
SS	superficial search
GA	genetic algorithm
AC	aspiration criterion
PSO	particle swarm optimisation
TSP	traveling sales problem

LIST OF CHANNELS

S.No	Channel No	Channel Coeffecients
1.	$H_1(z)$	$1+0.5z^{-1}$
2.	$H_2(z)$	$0.3482+0.8704z^{-1}+0.3482z^{-2}$
3.	$H_3(z)$	$0.4084+0.8164z^{-1}+0.4084z^{-2}$
4.	$H_4(z)$	$0.2+0.8z^{-1}+1.0z^{-2}$
5.	$H_5(z)$	$0.5+1.0z^{-1},$
6.	$H_6(z)$	$0.2+0.3z^{-1}+1.0z^{-2}+0.3z^{-3}$
7.	$H_7(z)$	$0.407-0.815z^{-1}-0.407z^{-2}$
8.	$H_8(z)$	$0.35+0.8z^{-1}+1.0z^{-2}+0.8z^{-3}$
9.	$H_9(z)$	$0.9413+0.3841z^{-1}+0.5684z^{-2}+0.4201z^{-3}+1.0z^{-4}$

ABSTRACT

In recent years, a growing field of research in “Adaptive Systems” has resulted in a variety of adaptive automatons whose characteristics in limited ways resemble certain behaviors of living systems and biological adaptive processes. The essential and principal property of the adaptive systems is its time-varying, self-adjusting performance by using a process called “learning” from its environment. A channel equalizer is a very good example of an adaptive system, which has been considered in this work to assess its performance with reference to various novel learning algorithms developed.

The two main threats for the digital communication systems are Inter-symbol Interference (ISI) and the presence of noise in the channels which are both time varying. So, for rapidly varying channel characteristics, the equalizer too need to be adaptive. In order to combat with such problems various adaptive equalizers have been proposed. Particularly, when the decision boundary is highly nonlinear, the classical equalizers (so called linear ones) do not perform satisfactorily.

The use of Artificial Neural Networks (ANNs) provides the required nonlinear decision boundary. The Back Propagation (BP) algorithm revolutionized the use of ANNs in diverse fields of science and engineering. The main problem with this algorithm is its slow rate of convergence. But the high speed digital communication systems, in the presence of rapidly fading channels, demand for faster training. To overcome this problem a faster method of training the neural network using RLS algorithm is proposed in this thesis work.

But both the BP and RLS based BP algorithms belong to the family of Gradient-based algorithms, which have the inherent problem of getting trapped in local minima. Since obtaining a global solution is the main criterion for any adaptive system, an efficient search technique is highly desirable. Tabu Search serves this purpose.

The popularity of Tabu Search (TS) has grown significantly in the past few years as a global search technique. In this dissertation, it is proposed to find the so-called optimal values of the ANN parameters (slopes and weights) for channel equalization. Results show that the use of TS for adapting the weights and slopes for an ANN not only improves the performance of the equalizer but also reduces the structural complexity of the ANN.

LIST OF FIGURES

CONTENTS

Chapter 1

INTRODUCTION

Background
Motivation
Thesis Contribution
Thesis Layout

INTRODUCTION

The advent of cheap high speed global communication ranks as one of the most important developments of human civilization in the second half of twentieth century. This was only feasible with the introduction of digital communication systems. Today there is a need for high speed and efficient date transmission over the communication channels. This is really a challenging task for the engineers to provide a reliable communication service by utilizing the available resources effectively in spite of there being many factors, as explained in the later sections, which distorts the signal transmitted through the communication channels. The main objective of the digital communication system is to transmit symbols with minimum errors. The high speed digital communication requires large bandwidth, which is not possible with a practical channel.

1.1 BACKGROUND

In this section we will consider the case of a practical channel and explore the reasons for the distortion of signals in them. The two principal causes of distortion in the communication channel are ISI and Noise.

1.1.1 Intersymbol Interference (ISI)

Ideally, the impulse response of a linear transmission medium is defined by

$$h(t) = A\delta(t - \tau) \tag{1.1}$$

where t denotes continuous time, $h(t)$ designates the impulse response, A is an amplitude scaling factor, $\delta(t)$ is the Dirac delta function and τ denotes the propagation delay incurred in the course of transmitting the signal over the channel. Equivalently, in frequency domain the above equation can be written as

$$H(j\omega) = A\exp(-j\omega\tau) \tag{1.2}$$

Where $H(j\omega)$ is the frequency response of the transmission media. In practice, it is impossible for any physical channel to satisfy the stringent requirements embodied in equations (1.1) and (1.2). The best we can do is to approximate equation (1.2) over a band of frequencies representing the essential spectral content of the transmitted signal, which makes the channel '*dispersive'* [1]. This channel impairment gives rise to '*Inter-symbol*

Interference'. - A smearing of the successive pulses into one another with the result that they are no longer distinguishable.

1.1.2 Noise

Some form of noise is always present at the output of every communication channel. The noise can be internal to the system, as in case of thermal noise generated by the amplifier at the front end of the receiver or external to the system, due to interfering signal originated from other sources.

The net result of the two impairments is that the signal received at the channel output is a noisy and distorted version of the signal that is transmitted. The function of the receiver is to operate on the received signal and deliver a reliable estimate of the original message signal to a user at the output of the system.

Hence there is a need for adaptive equalization. By equalization we mean the process of correcting channel induced distortion. This process is said to be adaptive when it adjusts itself continuously during data transmission by operating on the input [2]. The digital communication scenario is illustrated in Fig.1.1.

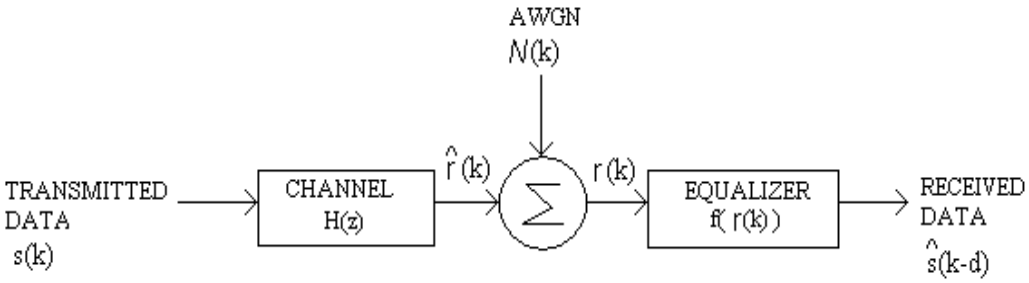

Fig..11 Schematic of Digital communication system

In the figure $s(k)$ is the transmitted data, the term $N(k)$ represents the Additive White Gaussian Noise (AWGN), $r(k)$ is the received signal and $\hat{s}(k-d)$ is an estimate of the transmitted data. Here the term d is called the decision delay, the importance of which is explained in chapter 3. Simply speaking the equaliser must perform the inverse operation of the channel [3, 4]. The channel transfer function $H(z)$ accounts for the ISI introduced by it.

1.1.3 Adaptive Equalisation

It is very difficult for estimating both the channel order and the distribution of energy among the taps and even it is very difficult to predict the effect of the environment on these taps. Hence it is a must for the equalization process to be adaptive. The equaliser need to be adapted very frequently with the changing environment. This includes two phases [5]. Firstly

the equaliser needs to be trained with some known samples in the presence of some desired response (Supervised Learning). After training the weights and various parameters associated with the equaliser structure is frozen to function as a detector. These two processes are frequently implemented to keep the equaliser adaptive. We call 'the Equaliser is Frozen' if we keep the adaptable parameters of the equaliser constant. Figure 1.2 depicts how the equalisation process is adaptive.

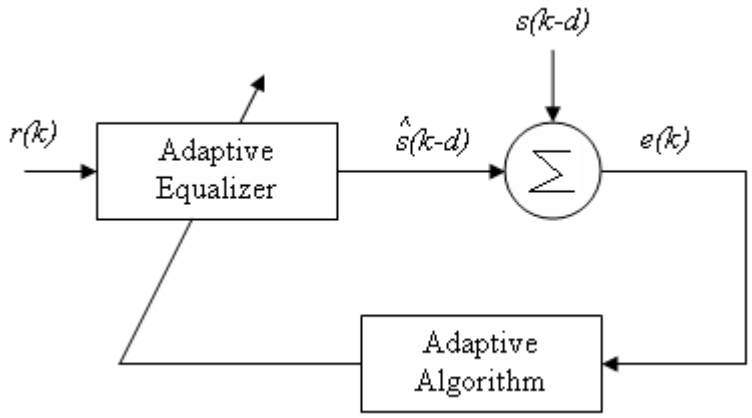

Fig. 1.2 Adaptive Equaliser

The received data is fed to an equaliser, the output of which is compared with some desired response, which here is the transmitted data with some decision delay, to get some error $e(k)$. This error is used to update the adaptable parameters of the equaliser using some adaptive algorithm. These steps constitute the training process of the equalisation. After the completion of training, the equaliser output is compared with some threshold and decision is made regarding the symbol received.

1.1.4 FIR model of a channel:

The channel can be characterised by a FIR filter (which accounts for ISI) with additive noise source [6].

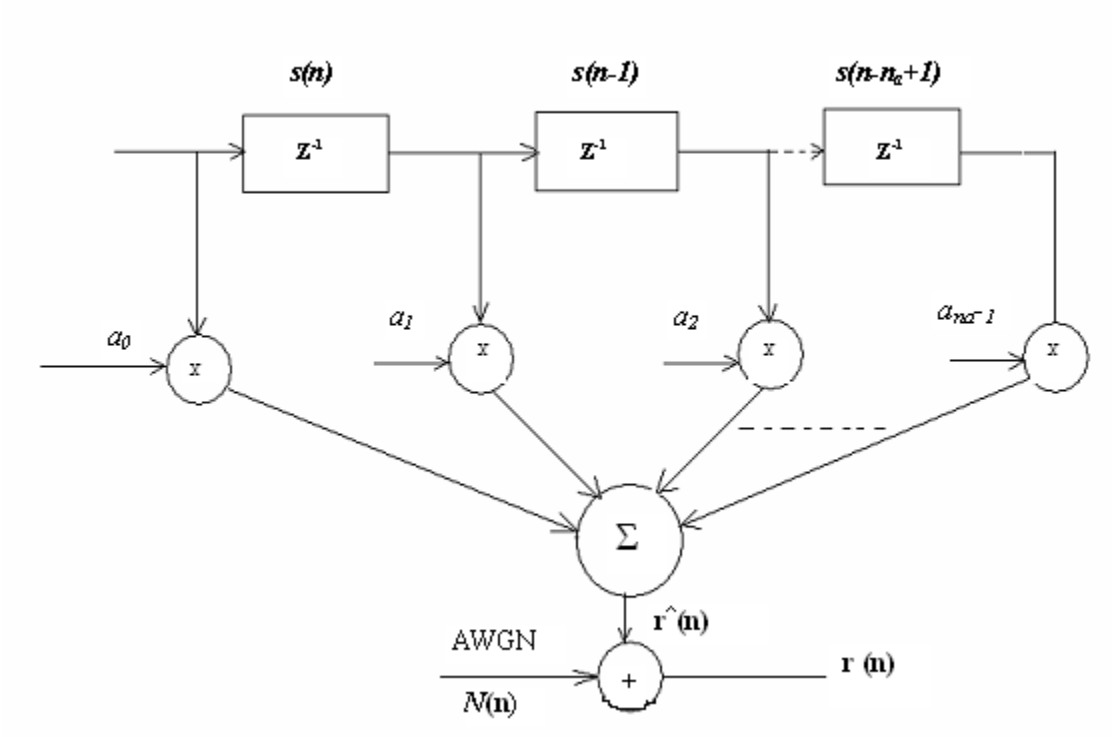

Fig. 1.3 FIR model of a channel

The channel impulse response in the *z*-domain can be represented by

$$H(z) = \sum_{i=0}^{n_a-1} a_i z^{-i} = a_0 + a_1 z^{-1} + a_2 z^{-2} + ... \qquad (1.3)$$

Where n_a represents the length of the channel impulse response (channel order) and the channel provides the dispersion upto n_a samples. The coefficients a_i represent the strength of the dispersion. The channel taps can be complex valued. The output from FIR modeled channel is described as

$$r(k) = \sum_{i=0}^{n_a-1} a_i s(k-i) + \mathrm{N}(k) \qquad (1.4)$$

Where $r(k)$ is the observed channel output (which is input to the equalizer) and $\mathrm{N}(k)$ represents (AWGN). For the case of computer simulations the taps of the FIR filter are chosen at the signal's sampling interval and coefficients chosen to accurately model the impulse response.

1.1.5 Classification of adaptive equalisers:

In general adaptive equalisers are either supervised or unsupervised. The equalizers with unsupervised training are called blind equalisers. The classification of the equalisers is shown in the figure1.4

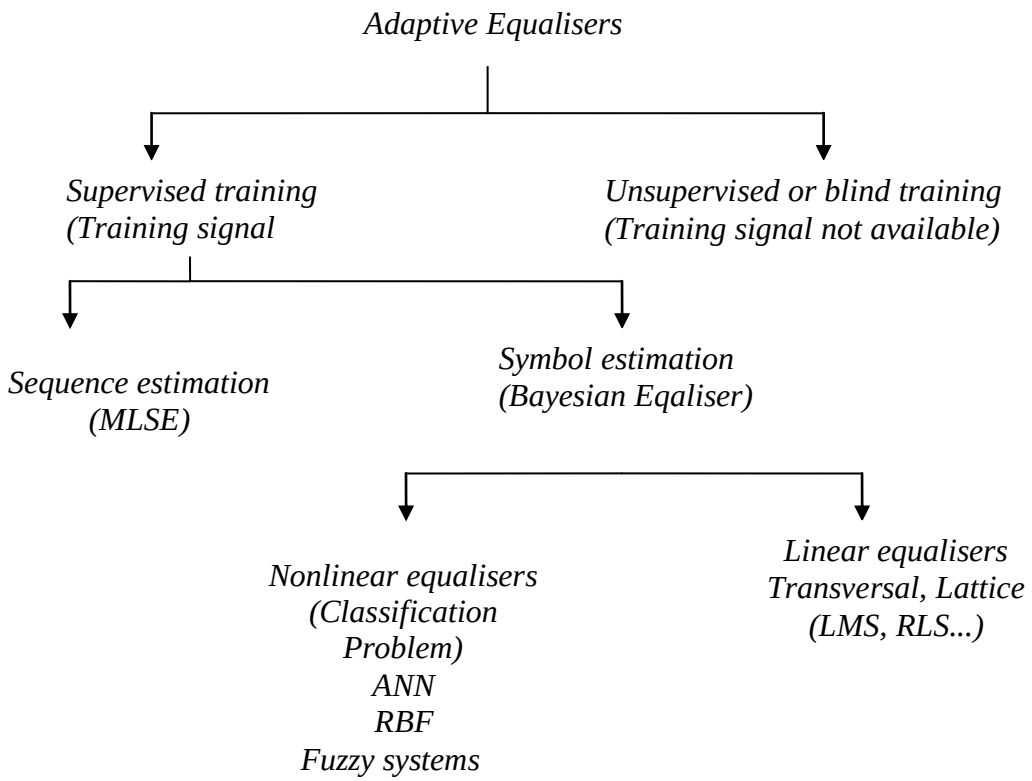

Fig. 1.4 Adaptive Equalisers classification

1.1.6 Optimal symbol-by-symbol equaiser: Bayseian equaliser

The optimal symbol-b-symbol equalizer is termed as Byesian equaliser. To derive the equaliser decision function the discrete time model of the base band communication system is presented in the figure1.5

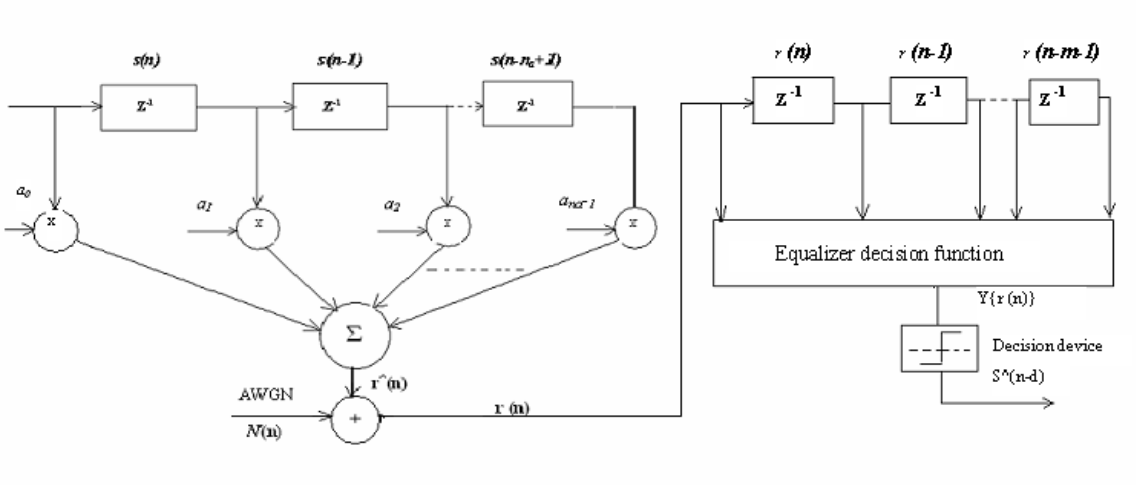

Fig. 1.5 Discrete time model of a digital communication system

The equaliser uses an input vector r(n) ϵR^m, the *m* dimensional space where the term *m* is the feed forward order of the equaliser. The equaliser provides a decision function $G\{r(n)\}$ based on the input vector which is passed through a decision device to provide the estimate of the transmitted signal ŝ(*n-d*) where *d* is a delay associated with equaliser decision. The communication system assumed to be a two level PAM system, where the transmitted sequence *s(n)* is drawn from a independent identically distributed sequence comprising $\{\pm\}$ symbols. The noise source is Additive White Gaussian Noise (AWGN) characterized by zero mean and a variance of σ_N^2

The eualiser performance is described by the probability of misclassification w.r.t Signal to Noise ratio(SNR).The SNR is defined as

$$SNR = \frac{\varepsilon[\hat{r}(n)^2]}{\varepsilon[N(n)^2]}$$

$$= \frac{\sigma_S^2 \sum_{i=0}^{n_a-1} a_i^2}{\sigma_N^2}$$

Where ε is the Exponential operator, σ_S^2 represents the transmitted signal power and is the channel power $\sum_{i=0}^{n_a-1} a_i^2$ and With assumption that the signal is draw from i.i.d. sequence of, $\{\pm\}$ the signal power σ_S^2 =1 becomes .Hence the SNR can be represented as

$$SNR = 20log_{10}^{\left(1/\sigma_N^2\right)} dB$$

The equaliser uses the received signal vector $r(n)=[r(n),r(n-1),\ldots\ldots r(n-m+1)]^T \in R^m$ to estimate the delayed transmitted symbols(n-d).The decision device at the equalizer output uses a *sgn(x)* function

$$sgn(x) = \begin{cases} +1 & if \ x \geq 0 \\ -1 & if \ x < 0 \end{cases}$$

Hence the estimate of the transmitted signal given by the equalizer is

$$\hat{s}(n-d) = sgn(G\{r(n)\}) = \begin{cases} +1 & if \ G\{r(n)\} \geq 0 \\ -1 & if \ G\{r(n)\} < 0 \end{cases}$$

The performance of an equaliser can be evaluated as follows. For bit error rate (BER) calculation if the equaliser is tested with statistically independent data of 10^7 channel samples then an error value e_i is generated in the following manner.

$$e_i = \begin{cases} 0 & if \ \hat{s}(n-d) = s(n-d) \\ 1 & if \ \hat{s}(n-d) \neq s(n-d) \end{cases}$$

Then BER is evaluated in decimal logarithm

$$BER = log_{10}^{\left(\sum_{i=0}^{10^7} \frac{e_i}{10^7}\right)}$$

The process of equalisation discussed here can be viewed as a classification process in which the equaliser partitions the input space in to two regions corresponding to each of the transmitted symbols {+1/-1}.The loci of the points which separate these two regions is termed as the *decision boundary* .If the received signal vector is perturbed sufficiently to cross the decision boundary due to the presence of AWGN, misclassifications result. To minimize the probability of misclassifications for a given received signal vector r(n),the transmitted symbol should be estimated based on having $s(n) \in \{\pm\}$ a maximum a posteriori

probability MAP[8,21]. The partition which provides the minimum probability of misclassification is termed as optimal (Bayseian) decision boundary.

1.1.7 Channel State Diagram

A channel state diagram is a plot drawn between the present received sample $r(k)$, on x-axis and the previous sample $r(k-1)$, on y-axis.

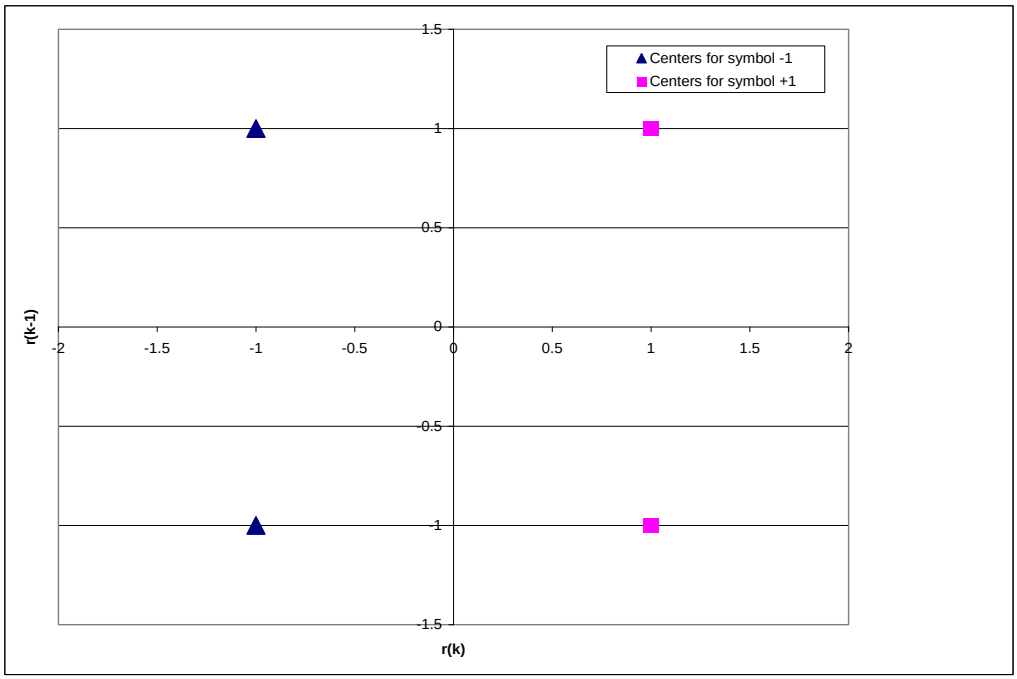

Fig 1.6 Channel State Diagram for an Ideal Channel

Fig. 1.6 is the channel state diagram of an ideal channel. It has four states, two belong to one class and the remaining two corresponds to other class. The two states, shown by triangles correspond to the centers for the transmitted bit being -1. The reaming two states shown by rectangles correspond to centers for the transmitted bit being +1.

As explained earlier, the practical channels are not ideal. They have the inherent problem of ISI and Noise. Due to the presence of ISI, the number of channel states increases from 4 to 2^{n_a+1}. And the presence of noise makes these states into clusters with the present state deviated from any of its original state by an amount which depends on the SNR at the receiver.

Since this channel state diagram is a plot between $r(k)$ and $r(k-1)$, this explains the classification feasibility for that channel at a specified SNR. The arrangement of the clusters determines the required equaliser order and channel delay (details are explained in chapter 3). And even the channel state diagram for some channels explains the need for equalization.

This thesis investigates supervised equalisers in general. The process of supervised learning can again be classified into either Sequence estimation or symbol detection. Sequence estimation equalisers depend on the principle of maximum likelihood detection. The optimal solution for this class of equalisers is the maximum likelihood Viterbi algorithm (MLVA) which determines the estimated symbol sequence [6]. This MLVA provides the lowest error rate attainable for any equaliser when the channel is known but computationally very expensive. As this needs the estimation of the channel this family of equalizers are not considered in this work.

The symbol detectors can be further classified as either linear or nonlinear depending on the structure and training algorithm used. The linear equalizers may be either transversal or lattice. The main advantage of lattice equaliser is its numerical stability and faster convergence [7].

1.1.8 Symbol –by-Symbol linear equaliser

The symbol detectors can be further classified as either linear or nonlinear depending on the structure and training algorithm used. The linear equalisers may be either transversal or lattice. The main advantage of lattice equalizer is its numerical stability and faster convergence [7].The structure of the linear equaliser is presented in the figure1.7

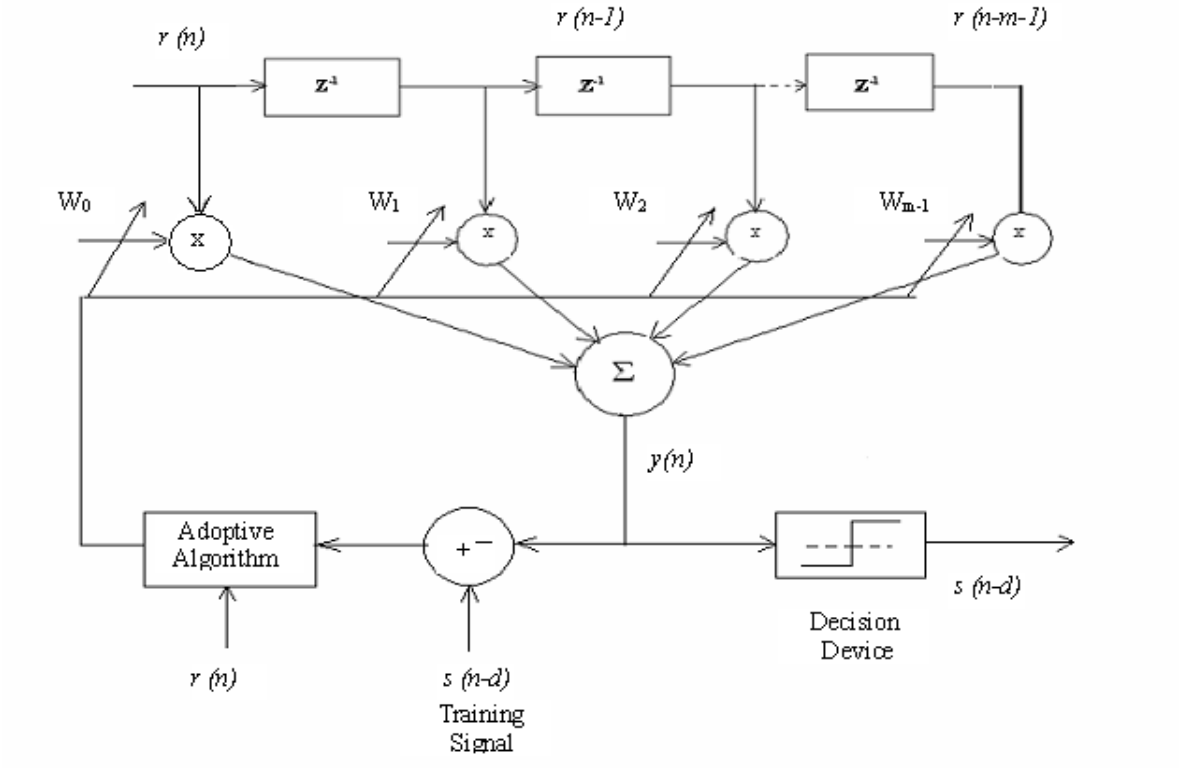

Fig 1.7 Structure of linear equaliser

The equaliser consists of a tapped delay line(TDL) which receives the receiver sampled input vector $r(n)=[r(n),r(n-1),.....r(n-m+1))]^T$ and provides an output $y(n)$ which is the convolution sum of the input vector $r(n)$ with the weight vector **w**. The out put is computed once per symbol and can be represented as

$$y(n) = \sum_{i=0}^{m-1} w_i r(n-i)$$

The decision device present at the output of the equalizer provides the transmitted signal constellation .The MMSE criteria provide equalizer tap coefficients to minimize the mean square error at the equalizer output before the decision device. This condition can be represented as

$$j(n) = \varepsilon|e(n)|^2$$

$$e(n) = s(n-d) - y(n)$$

Where e(n) is the error associated with the equalizer output y(n).Adaptive algorithms like Zero Forcing Algorithm(ZF),Least Mean Square Algorithm(LMS),Recursive Least squares Algorithm(RLS) can be used to update the equaliser weight vector during the training period. Linear equalisers with higher weight dimensions can improve accuracy. How ever higher dimensions leave the equalisers susceptible to noisy samples and such structures takes long time to converge. Thus LTE suffers performance degradation when the communication channel causes severe ISI distortion. When the channel has deep spectral null in its bandwidth , linear equalisations performs poorly since it places a high gain at the frequency of the null, thus enhancing the additive noise. Under such conditions decision feed back[4] can be employed to overcome these limitations.

1.1.9 Decision feed back equaliser

The basic structure of the Decision feed back equaliser is presented in the figure1.8

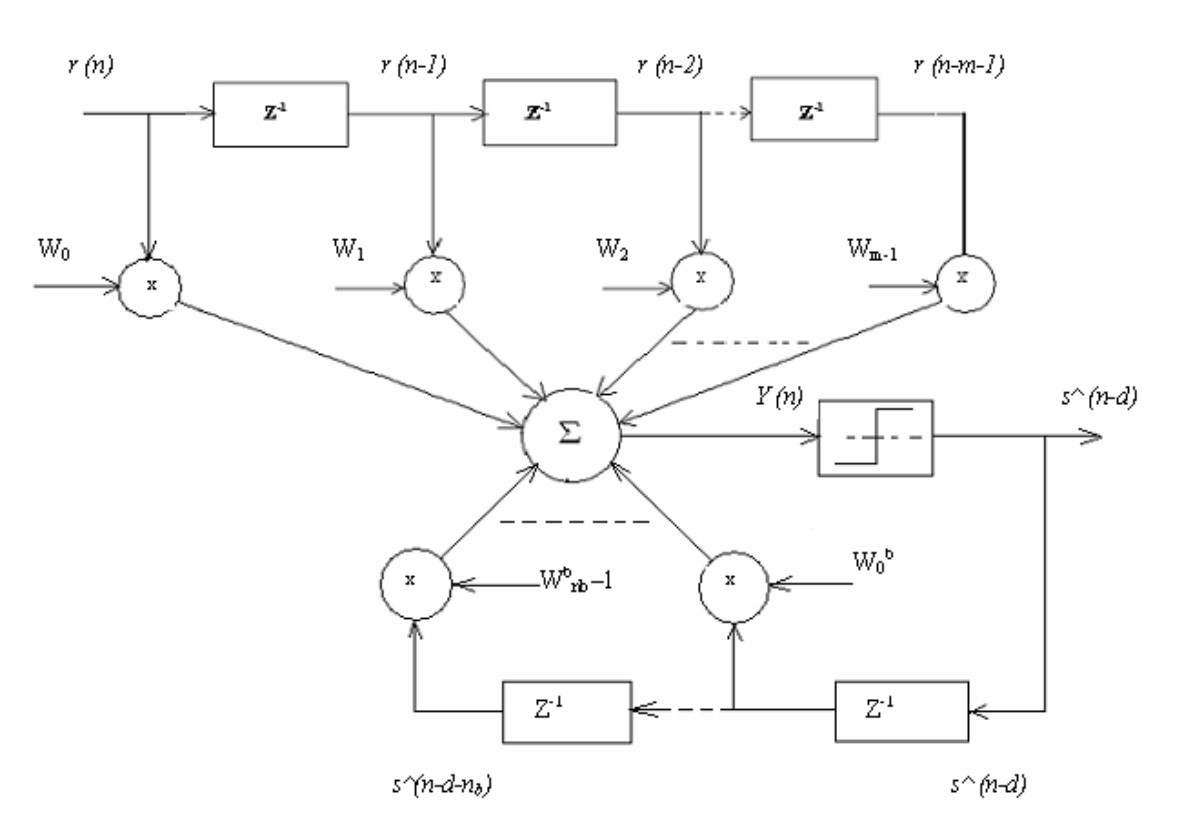

Fig 1.8 Structure of linear decision feedback equaliser

This equalizer is characterized by its feed forward order m and the feed back order n_b. The equalizer uses m feed forward samples and n_b feedback samples from the previously detected samples. The signal vector associated with the feedback weight vector $W_b = \left[W_0^b, W_1^b, \ldots\ldots\ldots\ldots W_{n_b-1}^b\right]^T$ is given by .Considering the DFE is updated with recursive LMS algorithm, the feedforward weights and the feedback filter weights can be jointly updated using the common error signal e(n).The feed back section in the filter helps to remove the ISI contribution from the estimated symbols and hence the DFE provides better performance than a conventional feed forward linear equalizer under severe ISI distortion.

The non-linear equalisers are used in applications where the channel distortion is too severe for a linear equalizer[22,23,24] to handle. There are few channels for which the decision boundary is very complicated and a simple linear filter equalizer cannot perform well in those cases. Further the process of equalization can be considered as the classification problem in which the equaliser needs to classify the input vector into a number of transmitted symbols. The optimal solution for the symbol detection equaliser is given by the Bayes Decision theory [6, 8] which inherently is a nonlinear classifier. Hence structures which can incorporate some amount of nonlinearity are required to obtain optimal or near optimal solution. Some of the examples of nonlinear equalisers include those using Artificial Neural Networks (ANNs), Fuzzy Systems etc. The thesis work mainly concentrates on the neural network equalisers. The well known algorithm for training the neural network equaliser is the Back Propagation (BP) Algorithm.

1.2 MOTIVATION

The adaptive equalizers have undergone many changes in the last two decades with the introduction of ANNs and many modifications in their training algorithms [9-12]. Almost much of the research was on modification of BP algorithm to improve its convergence [10, 11] or improving the performance of equaliser in some form. But the BP algorithm and its derivatives are all Gradient-based methods. Since these techniques converge locally, they often become trapped at sub-optimal solutions depending on the serendipity of the initial starting point. Since obtaining an optimal solution is the goal of ANN training, a global search technique seems more suitable for this difficult nonlinear optimization problem.

The popularity of Tabu search (TS) has grown significantly in the past few years as a global search technique [13, 14]. The main principle of TS is that it has some memory of the states that has already been investigated and it does not revisit those states. It considers the set of all possible neighbor states and takes the best one, but it will also take the best move in the neighborhood which might be worse than the current move. The TS focuses greatly on expanding the global search area and avoiding the search of the same area. It can always get much better global solutions. The TS uses a Tabu list (TL) to memorize the visited states and keep from recurrent search. Aspiration criterion (AC) is set to activate the ‘‘tabued state’’ in the TL around which some good global states may be found.

Most related researches of TS are focused on the combinatorial optimization problems, such as assignment problem, scheduling problem and TSP [13-15]. Few research works were done in the neural network learning. TS helps the NN learning process to the

gradient technique ‘‘jump out of’’ the local minima and get a great improvement on the gradient technique. The TS can also be combined with other improved learning algorithms.

1.3 THESIS CONTRIBUTION

The main aim of the thesis is to develop novel neural network training algorithms that overcome the problem of local minima and to reduce the structural complexity of the neural network equaliser so that it can facilitate real time implementation with much ease.

Firstly a faster RLS based neural network training algorithm is developed. Here we use the Recursive Least Squares (RLS) algorithm for updating the synaptic weights of the neural network. It can be noted that using this algorithm convergence and hence training is faster. But this RLS algorithm is also a Gradient based Algorithm [1]. Hence this algorithm also has the problem of converging to local minima and hence no much improvement in performance in terms of BER can be noted, with having the only advantage of reducing the training time.

Secondly, the Tabu search (TS), which is famous as a global search algorithm, is used in the training process of the equaliser. The work includes development of three neural network learning algorithms, all based on TS. They can be summarized into two classes as follows

(a) Tabu based BP algorithm: Here the synaptic weights of the neural network are adapted using Tabu search (TS). This algorithm includes two phases - Superficial Search (SS) and Deep Search (DS). BP algorithm is used in both the SS and DS.

(b) Slope Adaptation using Tabu search: The slope of the activation function, which is hyperbolic tangent in this case, is adapted using Tabu search. This can again be done in two ways.

  (i) First adapt the synaptic weights using BP algorithm and then use TS for adapting the slope of the activation function.

  (ii) In this case, both the weights and the slopes are adapted parallely in each iteration. BP algorithm is used to adapt weight and TS for adapting the slope.

Results show that the use of TS in both the cases, weight adaptation and slope adaptation, improves the performance of the equaliser in terms of Bit Error Rate (BER) and also reduces the structural complexity of the neural network.

1.4 THESIS LAYOUT

The transmitted symbols $s(k)$ and the channel taps a_i can be complex valued. Here they are restricted to real valued. This corresponds to the use of multilevel pulse amplitude modulation (*M*-ary PAM) with a symbol constellation defined by

$$s_i = 2i - M - 1, \qquad 1 \le i \le M \tag{1.5}$$

Concentration on the simpler real case allows us to highlight the basic principles and concepts. Hence binary symbols $(M = 2)$ have been considered in the thesis as it provides a very useful geometric visualization of equalisation process. For the equalizer to classify, the decision boundary must effectively partition the *m*-dimensional space into *M*-decision regions.

In chapter 2, the use of neural networks as a nonlinear equaliser is explained. Here the improvement in BER performance of the neural network equalizer over the linear equalizers trained using LMS algorithm is shown. Also the need for nonlinear equalisers and decision feedback is demonstrated in terms of the channel state diagrams. The decision feedback equalizers (DFE) are explained.

Chapter 3 is dedicated to the discussion on various parameters influencing the performance of the equaliser. The effect of changing these parameters is explained in this chapter. Also, in this chapter, the importance of decision delay and its effect on BER performance of the equaliser is considered.

In chapter 4 a new method of training the neural network using RLS is proposed. The basic idea behind the RLS is explained and the proposed algorithm is given.

In chapter 5, the concept of Tabu search (TS) is explained and its application to neural network is given. The algorithm for adapting the weights and slopes of the neural network is presented. Chapter 6 is dedicated for results and discussion.

Chapter 2

EQUALISATION USING NEURAL NETWORKS

Introduction

Neural Network Equaliser

Decision Feedback Equalisation

EQUALISATION USING NEURAL NETWORKS

2.1 INTRODUCTION

Numerous advances have been made in developing intelligent systems, inspired by biological neural networks. Researchers from many scientific disciplines are designing artificial neural networks (ANNs) to solve a variety of problems in pattern recognition, Function approximation, prediction, optimization, associative memory, and control.

Conventional approaches have been proposed for solving these problems. Although successful applications can be found in certain well-constrained environments, none is flexible enough to perform well outside its domain. ANNs provide exciting alternatives, and many applications could benefit from using them.

2.2.1 Definition of Neural network

A neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest. To achieve good performance, they employ a massive interconnection of simple computing cells referred to as 'Neurons' or 'processing units'. Hence a neural network viewed as an adaptive machine can be defined as [16]

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1. *Knowledge is acquired by the network from its environment through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

2.1.2 Importance

It is apparent that a neural network derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize. The use of neural networks offers the following useful properties and capabilities:

- Massive parallelism
- Distributed representation and computation
- Learning ability
- Generalization ability
- Input-output mapping

- Adaptivity
- Uniformity of Analysis and Design
- Fault tolerance
- Inherent contextual information processing
- VLSI implentability.

The advent of neural networks marked the modeling of nonlinear adaptive systems which could provide high degree of precision, fault tolerance and adaptability compared to other forms of mathematical modeling [9]. So the artificial neural networks are predominantly used for equalization. The Back Propagation (BP) algorithm is the best known and widely used learning algorithm for training ANNs since its proposal by Rumelhart and LeCun.

2.1.3 Need for nonlinear equalisers

The main reason nonlinear equalisers are preferred over their linear counterpart is that the linear equalizers do not perform well on channels which have deep spectral nulls in the passband. In an attempt to compensate for the distortion, the linear equaliser places too much gain in the vicinity of the spectral nulls, thereby enhancing the noise present in these frequencies.

Non-linear equalizers outperform the linear equalisers in terms of BER [17]. Also the linear equalizers view equalisation as inverse problem while non-linear equalisers view equalisation as a pattern classification problem.

Consider the following example of the channel states for the two channels.

$$H_1(z) = 1 + 0.5z^{-1} \quad \text{and}$$

$$H_2(z) = 0.3482 + 0.8704\, z^{-1} + 0.3482\, z^{-2}$$

The channel state diagram for the channel $H_1(z)$ is shown in figure 2.1 and that of channel $H_2(z)$ is shown in figure 2.2. For these two channels, the channel state diagram are plotted for $d = 0$ and at a $SNR = 20dB$. It can be noted that for channel $H_1(z)$, we can easily draw a linear decision boundary. And hence the positive and negative centers can be easily classified. Basically this channel $H_1(z)$ is a minimum phase channel and hence classification is not a big problem in this channel. Problem starts when equalising the non-minimum phase channels [17].

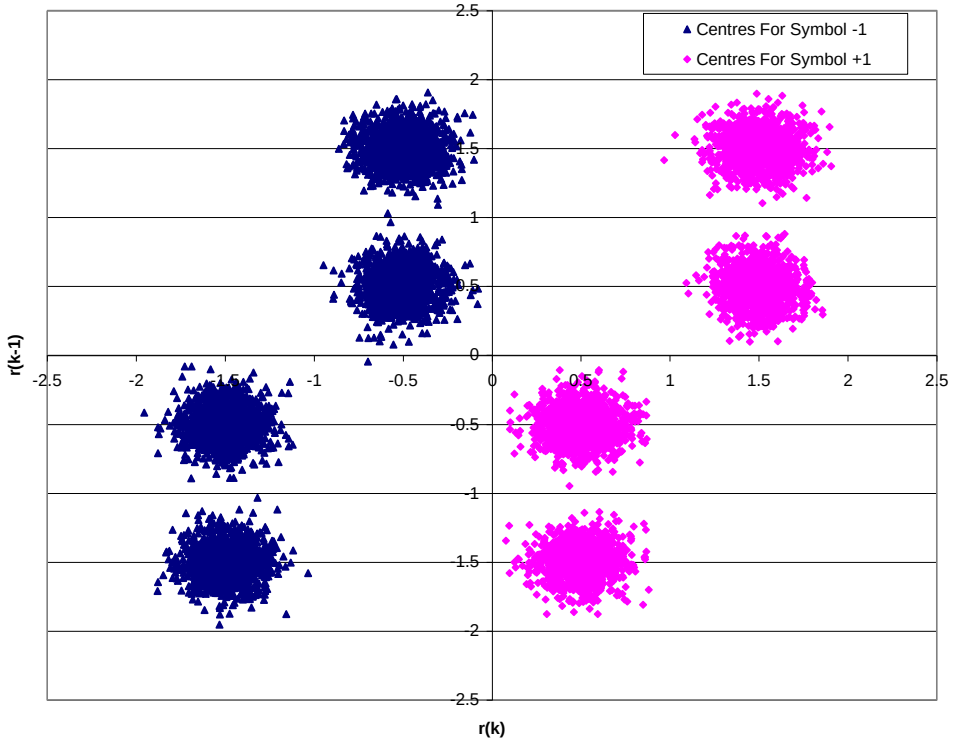

Fig. 2.1 Channel State diagram for $H_1(z)$

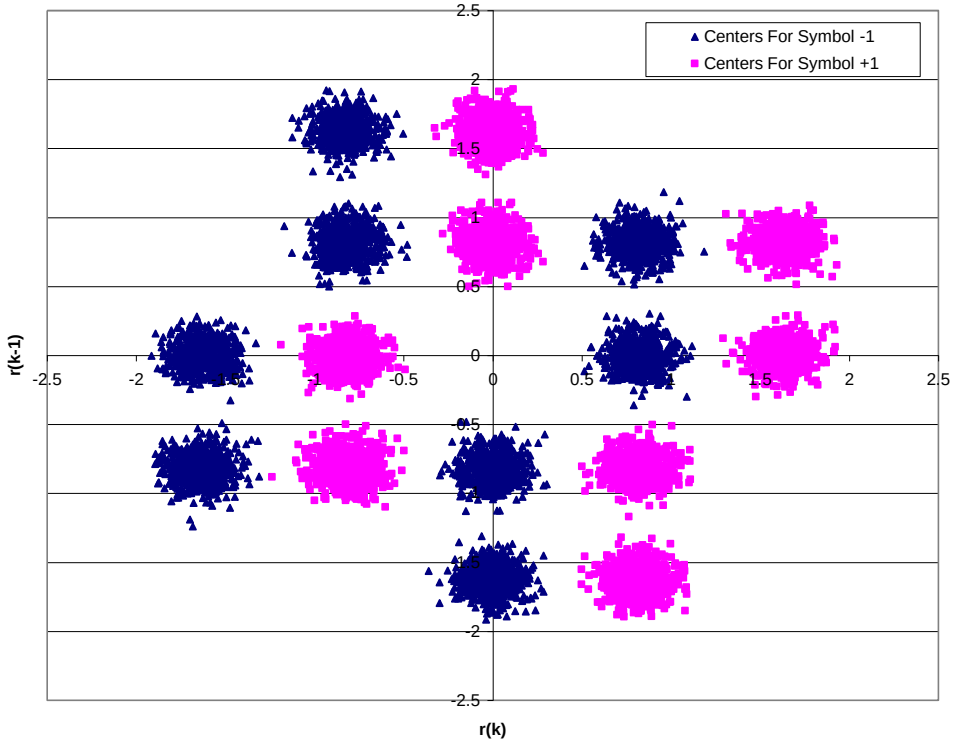

Fig. 2.2 Channel State diagram for $H_2(z)$

One of the popular channels belonging to this family is $H_2(z)$. It can be noted in figure 2.2, for this channel, a simple linear decision boundary cannot classify the symbols easily. It needs a nonlinear decision boundary or even a hyper-plane in multi-dimensional channel space. Such a decision boundary cannot be achieved using a linear filter. Hence we go for nonlinear structures which can adapt well to its environment are needed. Neural networks serve this purpose.

2.2 NEURAL NETWORK EQUALISER

These neural networks construct a functional relationship between input and output patterns through the learning process, and memorize that relationship in the form of weights for later applications [10]. The neural network equaliser outperforms the Linear Transversal filters in terms of bit error rate (BER) as most of the communication channels requires nonlinear decision boundary [17]. The structure of neural network equaliser is shown in the figure.

In the figure $r(k)$ represents received signal. The structure constitutes three significant parts- one input layer, a set of hidden layers, one output layer. All the nodes are interconnected by the weights w_{ij}^{l}, where i represents the destination node and j represents the source node. The superscript l gives the layer number.

An equalizer of order m implies that it has m input nodes in its input layer as shown in the figure. An equaliser will have a single node in its output layer. The signal received sequentially is allowed to propagate through the hidden layers up to the node in the output layer [9, 16].

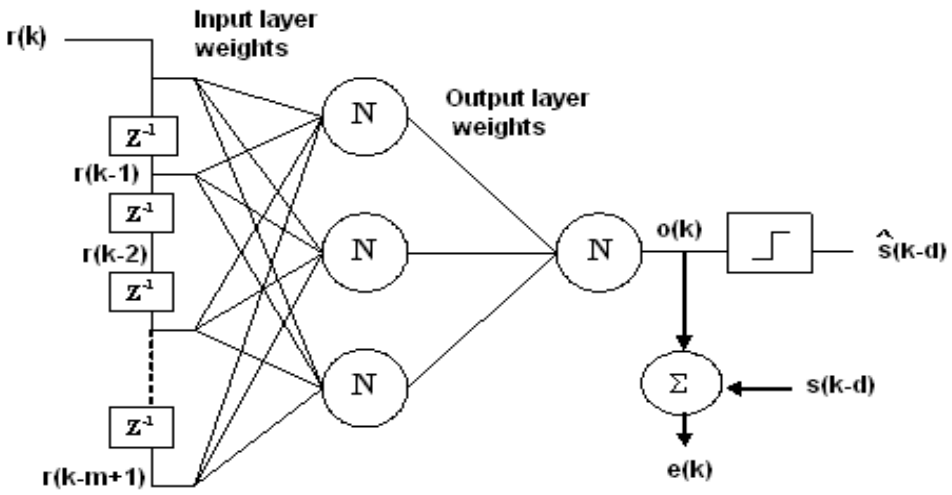

Fig.2.3 Neural Network Equaliser

The output of the each node y_i^l is the weighted sum of outputs of all the nodes in the previous layer and affected by the activation function, which here is the hyperbolic tangent function given by

$$\phi(x) = \frac{1-e^{-ax}}{1+e^{-ax}} \tag{2.1}$$

where a represents the slope of the activation function.

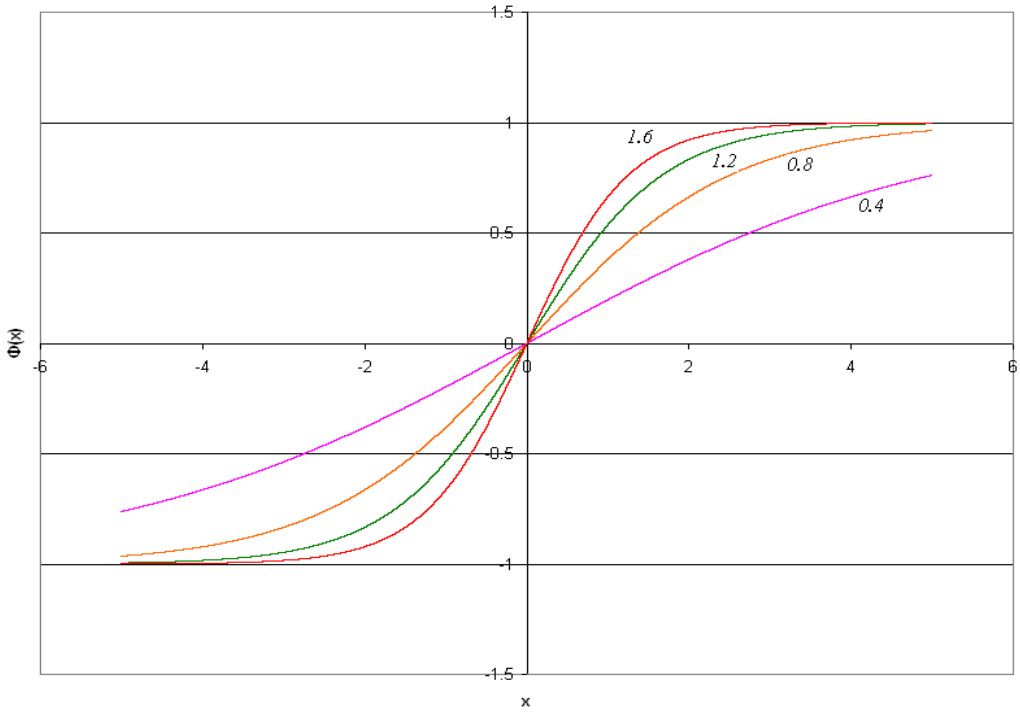

Fig. 2.4 Hyperbolic Tangent Function with varying slopes

The figure 2.4 shows the hyperbolic tangent activation function with slope a varying at 0.4, 0.8, 1.2 and 1.6. As slope increases the function tends towards threshold function. The function attains a maximum value of +1 at $x = \infty$ and a minimum value of -1 at $x = -\infty$. Hence the output of each node of the neural network will be in the range (-1, 1). Since we are using binary PAM signal which takes either of the two values -1 or +1, this function is very much suitable for equalization process.

Mathematically the forward propagation of the neural network is given by [16]

$$v_i^l = \sum_{j=1}^{N_{l-1}} (w_{ij}^{l-1} \cdot y_j^{l-1}) \tag{2.2}$$

$$y_i^l = \phi(v_i^l) \tag{2.3}$$

where v_i^l is called the induced local field or activation potential of node i in layer l and N_{l-1} is the number of neurons in the layer $(l-1)$.

2.2.1 Back Propagation Algorithm

The BP algorithm is a very good example for the Gradient-based algorithms. The BP algorithm consists of two passes through the different layers of the network: a forward pass and a backward pass [16]. In the forward pass, as mentioned earlier, an activity pattern (input vector) is applied to the sensory nodes of the network and its effect propagates through the network layer by layer. Finally an output is obtained at the output node as the actual response of the network. During the forward pass the synaptic weights are kept constant.

In the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with the error correction rule. The error signal, which is obtained by comparing the output of the node in the output layer with the desired response, is allowed to pass against the direction of synaptic weights (hence the name back propagation algorithm for this training process) and local gradients δ_i^l at each node is computed as given by Eqn.2.4.

$$\delta_j^l = \phi'(v_j^l) \cdot \sum_{i=1}^{N_{l+1}} (\delta_i^{l+1} \cdot w_{ji}^l) \tag{2.4}$$

The local gradient $\delta_j(n)$ depends on whether neuron j is an input node or a hidden node:

(1) If neuron j is an output node, $\delta_j(n)$ equals the product of the derivative $\varphi_j'(v_j(n))$ and the error signal $e_j(n)$, both of which are associated with neuron *j*.

(2) If neuron *j* is a hidden node, $\delta_j(n)$ equals the product of the associated derivative $\varphi_j'(v_j(n))$ and the weighted sum of the δs computed for the neurons in the next hidden or output layer that are connected to neuron *j*.

Due to the lack of availability of desired response at the hidden layers, it is not possible to compute the error at these nodes. Hence this local gradient is very important in providing the error measure at the hidden nodes. Using these local gradients the synaptic weights update is given by Eqn. 2.5

$$\begin{pmatrix} Weight \\ Correction \\ \Delta w_{ji}^l \end{pmatrix} = \begin{pmatrix} learningrate \\ parameter \\ \eta \end{pmatrix} \bullet \begin{pmatrix} local \\ gradient \\ \delta_j^{l+1} \end{pmatrix} \bullet \begin{pmatrix} inputsignal \\ ofneuron \\ y_i^l \end{pmatrix} \tag{2.5}$$

The weight correction Δw_{ji}^l is added to the present weight after each iteration. Once the weights are updated till the mean square error (MSE) is below some desired threshold, the

weights are kept fixed. Now the designed equalizer is tested for its performance with 10^6 samples. This is called the testing phase of the equalizer. This testing is done by calculating the bit error rate (BER) at different signal to noise ratio (SNR) values. This plot of SNR Vs BER is the performance plot for an equaliser.

Now consider the example of channel $H_2(z)$. If we observe the performance plots for the neural network equalizer trained using the BP algorithm and a linear transversal filter equalizer trained using the LMS algorithm, as shown in Fig 2.5, it can be noted that the neural network equalizer has a better BER performance.

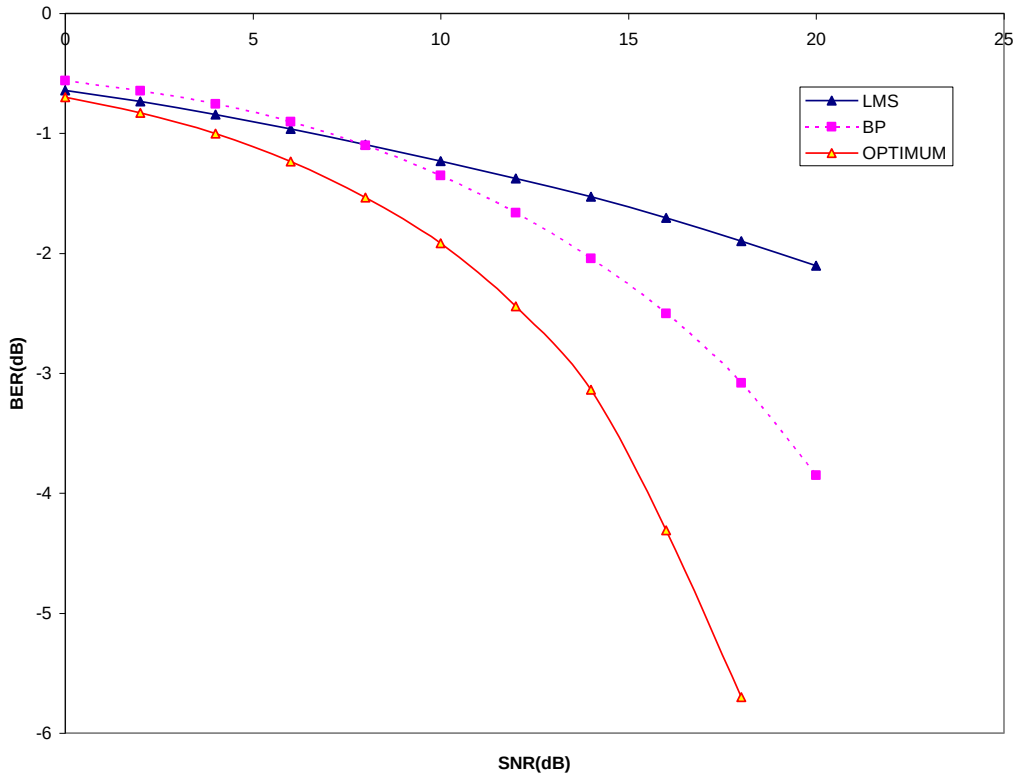

Fig. 2.5 BER comparison for FIR filter and Neural Network equaliser for $H_2(z)$

The plots in the above figure are for an equalizer order of $m = 4$, delay $d = 1$. For the equalizer we have chosen here 4-3-1 structure, which means the neural network has only one hidden layer with 3 nodes in it, 4 inputs and one output node. The main disadvantage of using neural networks is that their convergence is slower compared to linear filters.

2.3 DECISION FEEDBACK EQUALISATION

In this section the need for decision feedback, its structure and its importance is described. The advantage of using the decision feedback equaliser is that ISI is eliminated with out enhancement of noise by using past decisions to subtract out a portion of the ISI in addition to the feed forward filter [18]; a disadvantage is that decision error tend to propagate because they result in residual ISI and a reduced noise margin against noise at future decisions.

2.3.1 Need for decision feedback

There are some channels for which the negative centers and positive centers are very near and overlap in many cases due to the presence of additive noise. Such channels are called overlapping channels. One of the good examples of such a channel is

$$H_3(z) = 0.4084 + 0.8164\, z^{-1} + 0.4084\, z^{-2}.$$

The channel state diagram for this channel for $d = 1$ is shown in the Fig 2.6.

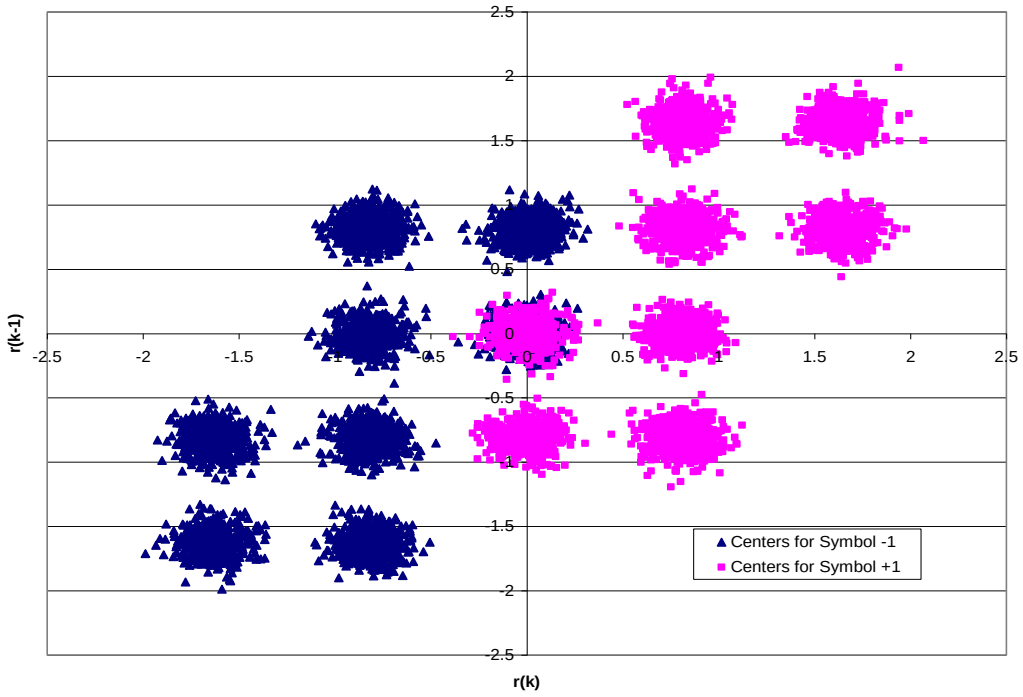

Fig.2.6 Channel state diagram for overlapping channel $H_3(z)$

Hence we need a decision feedback equaliser to classify these overlapping patterns. Earlier it is stated that equalisation process can be considered as a classification problem and the advantage of using the feedback concept is that it reduces the number of states for decision making and hence ease the classification.

2.3.2 Decision Feedback Neural Network Equaliser

The structure of the decision feedback neural network equaliser is shown in Fig. 2.7**.** It can be noted from figure that it is very much similar to the simple neural network equaliser except the inclusion of additional taps for feedback elements. This feedback is called decision feedback as we are feeding back the previous decisions. Number of feedbacks included is called feedback order n_b . Then the number of additional delay elements required is $n_b - 1$.

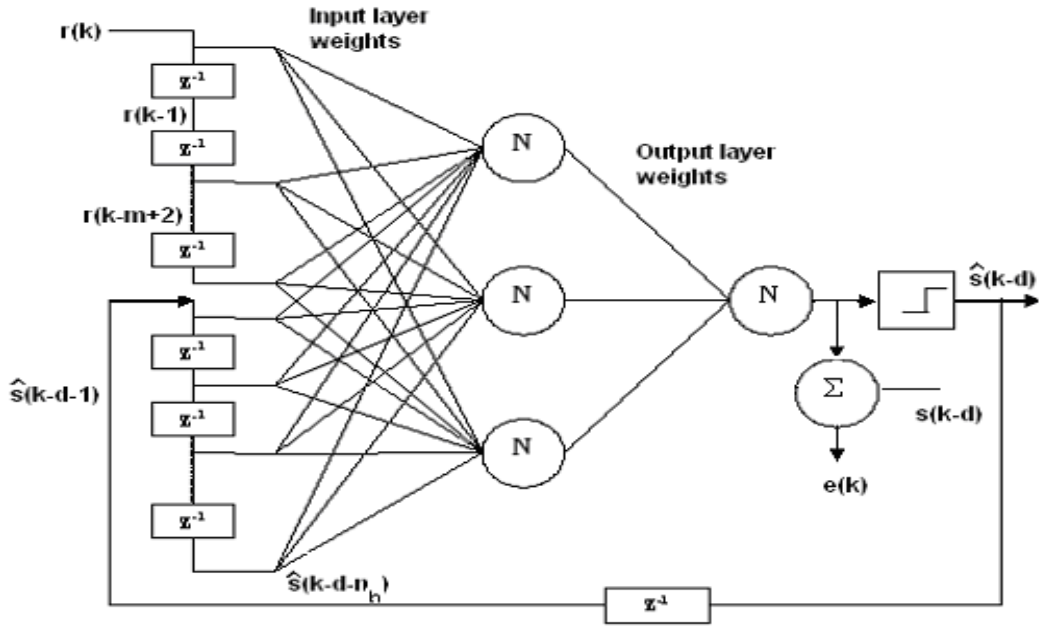

Fig. 2.7 Decision Feedback Neural Network Equaliser

Hence, now, the total number of nodes in the input layer becomes $m + n_b$ .in the absence of feedback number the transmitted symbols that influence the equaliser decision are

$$\{s(k), s(k-1),....., s(k-m-n_a+2\} \qquad (2.6)$$

Thus the channel input sequence has $n_s = 2^{m+n_a-1}$ combinations among which $n_s/2$ constitute centers belonging to symbol +1 and the remaining $n_s/2$ are for centers corresponding to symbol -1. Hence here all the n_s states are required for decision making.

If we include feedback then the number of transmitted symbols that influence the equaliser performance are

$$\{\langle s(k), s(k-1),...s(k-m-n_a+2)\rangle, \langle \hat{s}(k-d-1)...\hat{s}(k-d-n_a)\rangle\} \qquad (2.7)$$

Hence the feedback vector has $n_f = 2^{n_b}$ states. As a result of feedback, only a fractional number of these states, n_s / n_f are needed for decision making. It can be noted that it is sufficient to employ a feedback order n_b, given by [6]

$$n_b = n_a + m - 2 - d \tag{2.8}$$

In the next chapter the effect of changing the feedback order and its optimal values are explained. Fig. 2.8 shows the importance of decision feedback. It can be noted that for the channel $H_3(z)$, the performance of the neural network equaliser with out feedback is very poor. But if we use feedback for this channel, the BER performance will improve. In this example an equaliser order of $m = 3$, in both cases (with feedback and without feedback), feedback order is $n_b = 2$ and a decision delay of $d = 1$ has been considered.

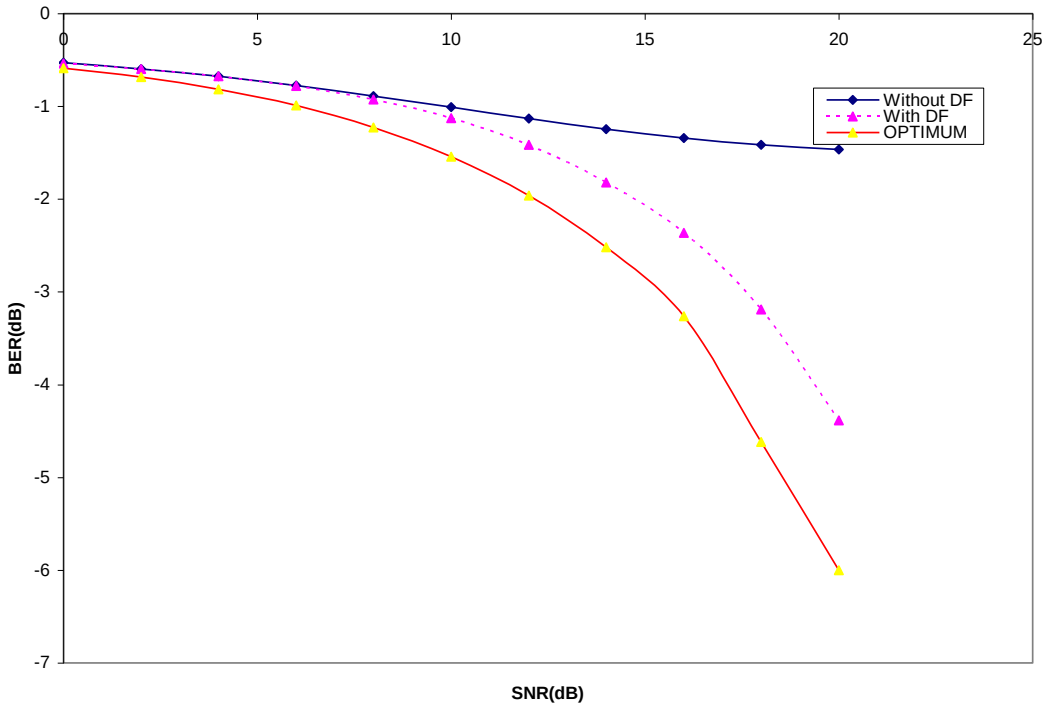

Fig.2.8 BER comparison for $H_3(z)$ with decision feedback and without decision feedback

The neural network structure chosen is a three layered i.e. $\langle 5,1 \rangle$ FNN, which means one hidden layer with 5 neurons in it and one output node. It can be noted that there is an improvement of almost 3dB in terms of BER at 20dB SNR. The feedback improves the performance of not only the overlapping channels but also other channels. Now consider the case of channel

$$H_4(z) = 0.2 + 0.8z^{-1} + 1.0z^{-2}$$

This channel does not have any of its states overlapping. It can be seen in Fig. 2.9 that the use of decision feedback improves the performance of equaliser. In this example an equalizer order of $m = 3$, feedback order of $n_b = 2$, decision delay of $d = 3$have been considered. And the $\langle 5,1 \rangle$ neural network structure has been used in both the cases. It can be noted that the use of feedback improves the BER performance of more than 2dB at 20dB SNR, the reason is that, as mentioned earlier, it eliminates ISI without enhancing noise by using past decisions to suppress the ISI. Also the number of states for decision making has been reduced. Hence decision boundary can be easily constructed between the centers belonging to symbol +1 and the centers belonging to symbol -1 and even the dimensionality of the decision boundary can be reduced as there are lesser number of states.

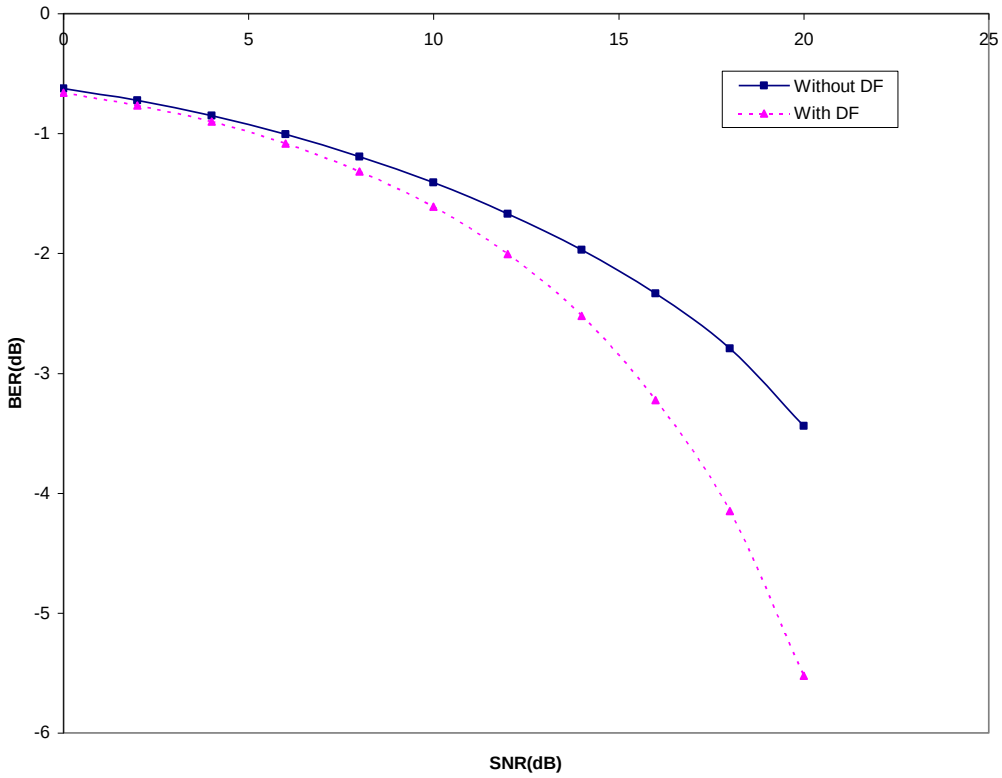

Fig. 2.9. BER Comparison for $H_4(z)$ with DFE and without DFE

Thus there is an additional advantage of using the decision feedback equalisers. They reduces the equaliser order and hence the structural complexity of the equaliser.

Chapter 3

PERFORMANCE OF EQUALISER UNDER THE INFLUENCE OF PARAMETER VARIATIONS

Effect of Additional noise level

Effect of Equaliser Order

Effect of Feedback Order

Effect of Decision Delay

PERFORMANCE OF EQUALISER UNDER THE INFLUENCE OF PARAMETER VARIATIONS

In this his chapter we will consider the influence of different parameters on equalizer performance. In the symbol-by-symbol detection procedure for equalization we apply the channel output which will be corrupted by both ISI and Noise, as described in the previous chapter, to the equaliser which will classify the symbols into their respective classes. The main goal of equalisation is to minimize the misclassification rate.

It has been noted that the BER performance of the equaliser are influenced by many factors which include the additional noise level, the equaliser order, the decision delay, number of samples used for training, and the neural network structure we have considered. It is noted in the earlier chapters that, for some channels classification is not possible without feedback as they may have overlapping states and also seen that the BER performance of the equalizer improves with the inclusion of feedback. Hence we get one more factor influencing the classification capability of the equaliser.

So, we can say that the principal parameters that affect the equaliser's BER performance are

(1) Additional noise level

(2) Equaliser order, m

(3) Feedback order, n_b

(4) Decision delay, d

In sections 3.1, 3.2, 3.3 and 3.4 we will compare the performance variations of the equalizers with the change in any of these principal parameters. However, as mentioned, there are others factors too, but their effect is not so appreciable. This will be discussed later in this chapter in section 3.4 and 3.5.

3.1 EFFECT OF ADDITIONAL NOISE LEVEL

As explained in chapter 1, for an ideal channel, which is not affected by ISI and Noise, the channel state diagram will have only four states. Due to the presence of ISI the number of channels increases. As the noise increases a point in the channel state diagram deviates from its state. As the noise level increases (SNR decreases), the spread of the clusters increases and hence the classification capability decreases for the equaliser.

Figures 3.1, 3.2, 3.3 and 3.4 are the channel state diagrams for the channel

$$H_5(z) = 0.5 + 1.0z^{-1}, \qquad d = 1$$

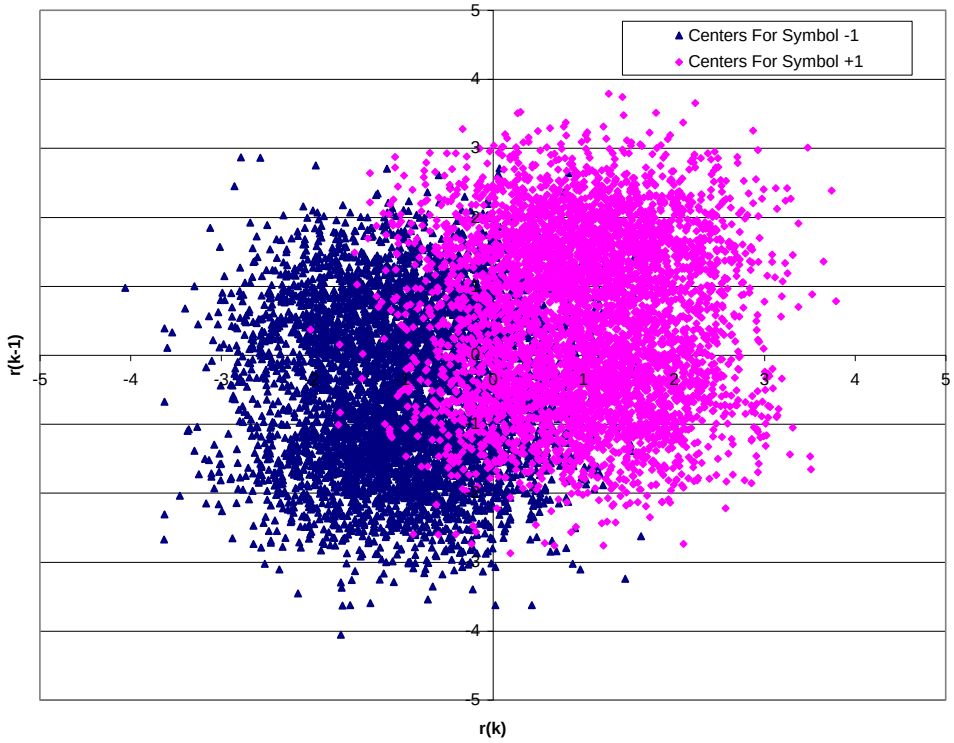

Fig. 3.1 Channel State Diagram for $H_5(z)$ at SNR=5dB

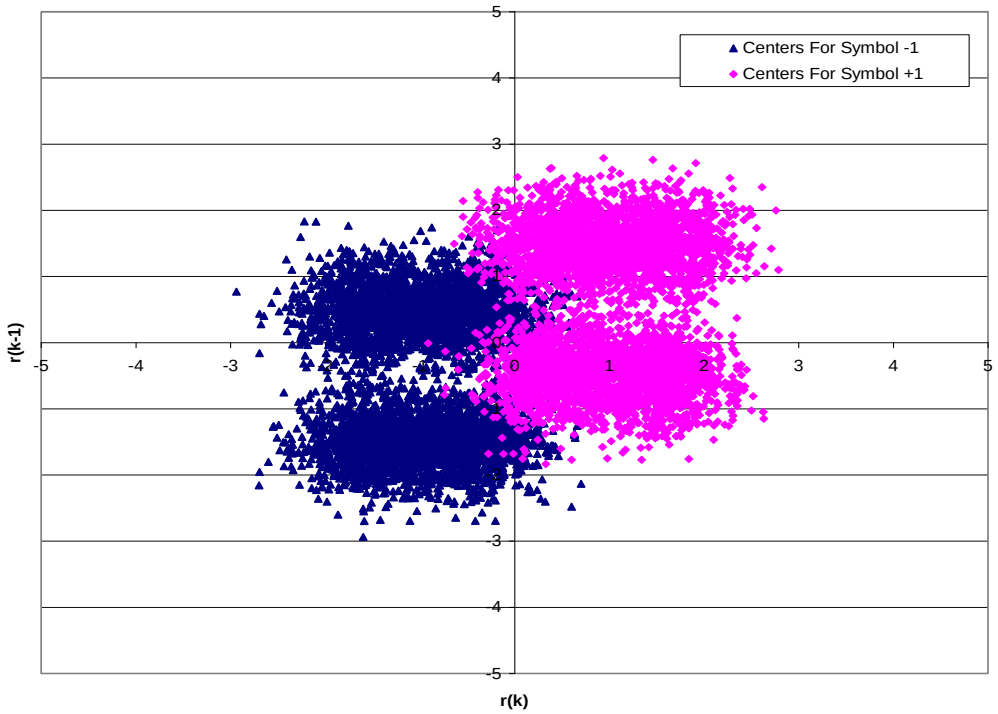

Fig. 3.2 Channel State Diagram for $H_5(z)$ at SNR=10dB

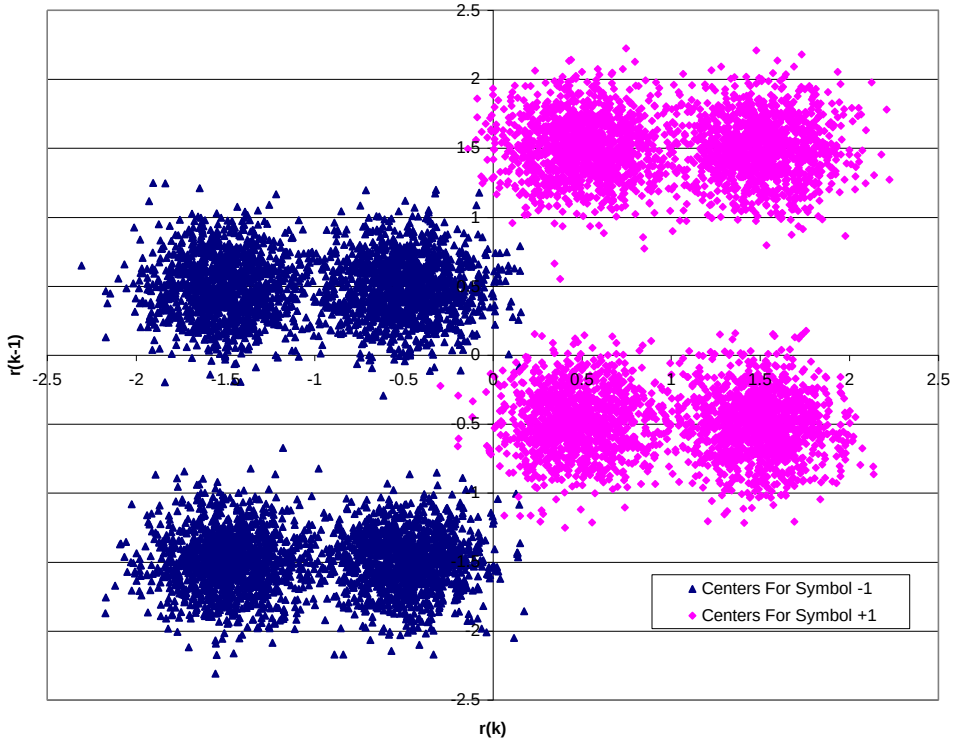

Fig. 3.3 Channel State Diagram for $H_5(z)$ ***at SNR=15dB***

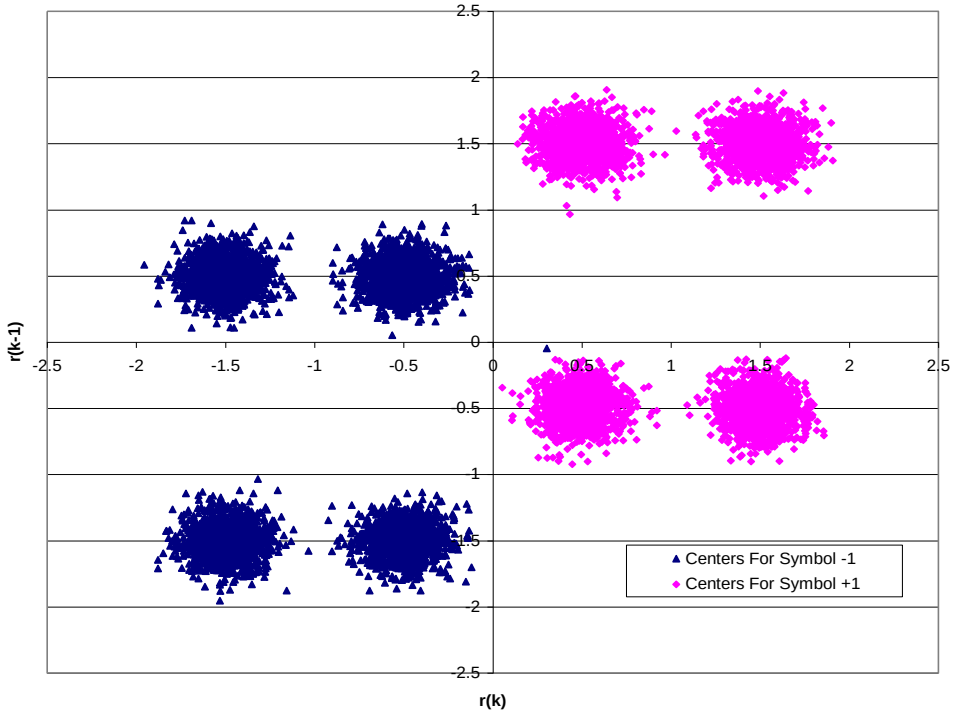

Fig. 3.4 Channel State Diagram for $H_5(z)$ ***at SNR=20dB***

Fig. 3.1 is plotted at an SNR=5dB. Here it can be noted that the point belonging to the cluster of a positive centre and those of negative centers overlap to a large extent, hence classification becomes difficult. Fig. 3.2, 3.3, 3.4 corresponds to the SNR values of 10dB, 15dB and 20dB respectively. As SNR increases, the problem of classification becomes easier.

3.2 EFFECT OF EQUALISER ORDER

The effect of equaliser order is directly related to Covers Theorem [16]. This theorem on the separability of patterns, which, in qualitative terms, may be stated as follows,

"*A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.*"

According to this theorem, as we move to higher dimensional space, classification becomes easier. For an equalizer, increase in the equaliser order, m, increases the dimensionality of the pattern space. And hence, an increase in the equaliser order eases the pattern classification.

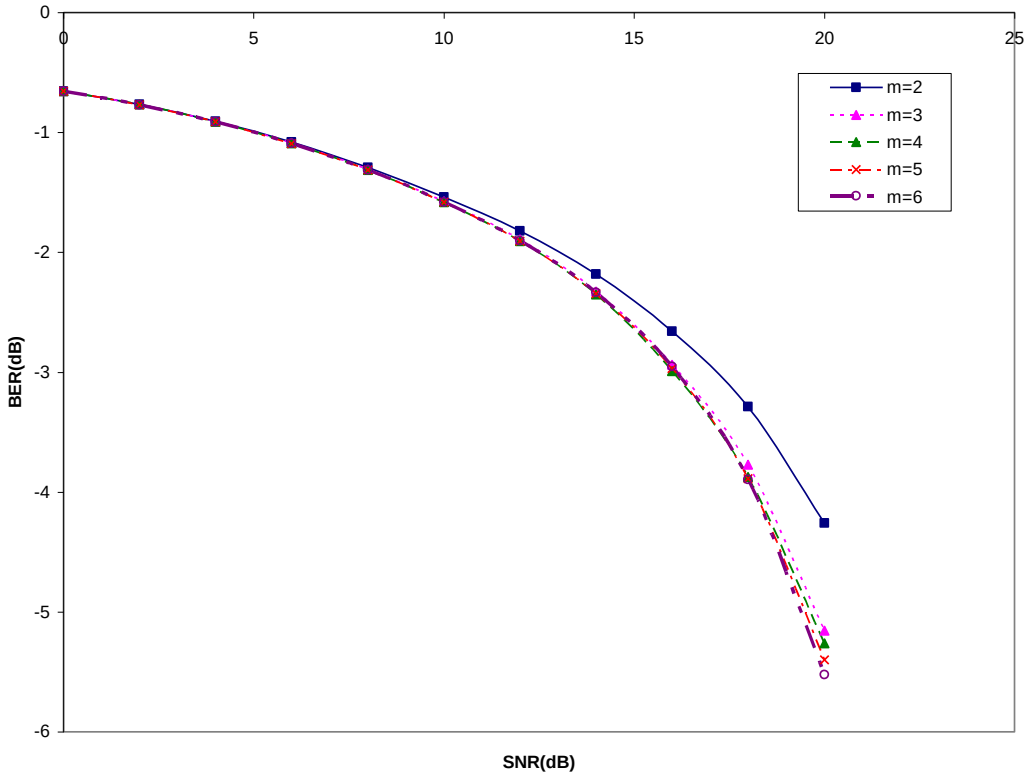

Fig. 3.5 BER comparison for $H_6(z)$ ***with varying*** m

Fig. 3.5 is the BER plot with varying equaliser order m. The plots are drawn for equalizers of order $m = 2,3,4,5,6$, for the channel

$$H_6(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$$

For this channel, the plots are obtained with a $\langle 5,1 \rangle$ FNN structure and for decision delay $d = 2$. It can be seen from the plots that with an increase in equaliser order from 2 to 3, there is an increase in equalizer performance of almost 1dB in terms of BER at the SNR value of 20dB. But if we increase the equaliser order further to 4, 5 or 6, there is no appreciable change in performance. This is because for an equaliser, for a fixed value of decision delay, d, an equalizer order of $m = d + 1$ is sufficient [6]. And for the equalizer order of $m > d + 1$ will almost have a similar performance and there will be no much appreciable change in BER performance. Even in the example of $H_6(z)$, this phenomenon can be noted. We have chosen an equaliser order of $d = 2$. And hence there will be no much improvement in BER performance for $m > 3$ compared to $m = 3$.

3.3 EFFECT OF FEEDBACK ORDER

As mentioned in the previous chapter, with the introduction of feedback, number of states present while decision making decreases, hence classification becomes easier. Also as feedback order increases, number of states decreases, hence performance increases as there is an increase in margin existing between states. Fig. 3.6 shows the plots for varying feedback order for the channel $H_3(z)$, with decision delay $d = 1$, equaliser order $m = 2$, and a neural network structure $\langle 5,1 \rangle$.

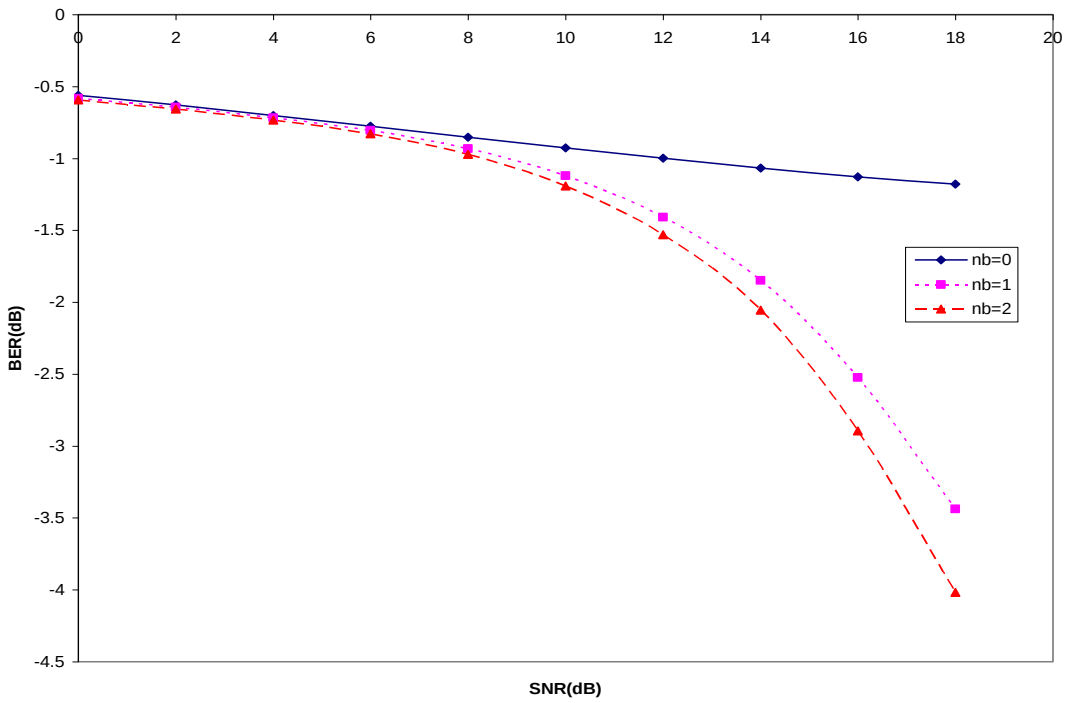

Fig. 3.6 BER performance comparison for varying feedback order for $H_3(z)$

3.4 EFFECT OF DECISION DELAY

The effect of decision delay d, can be easily understood by observing its effect on the channel state diagram.

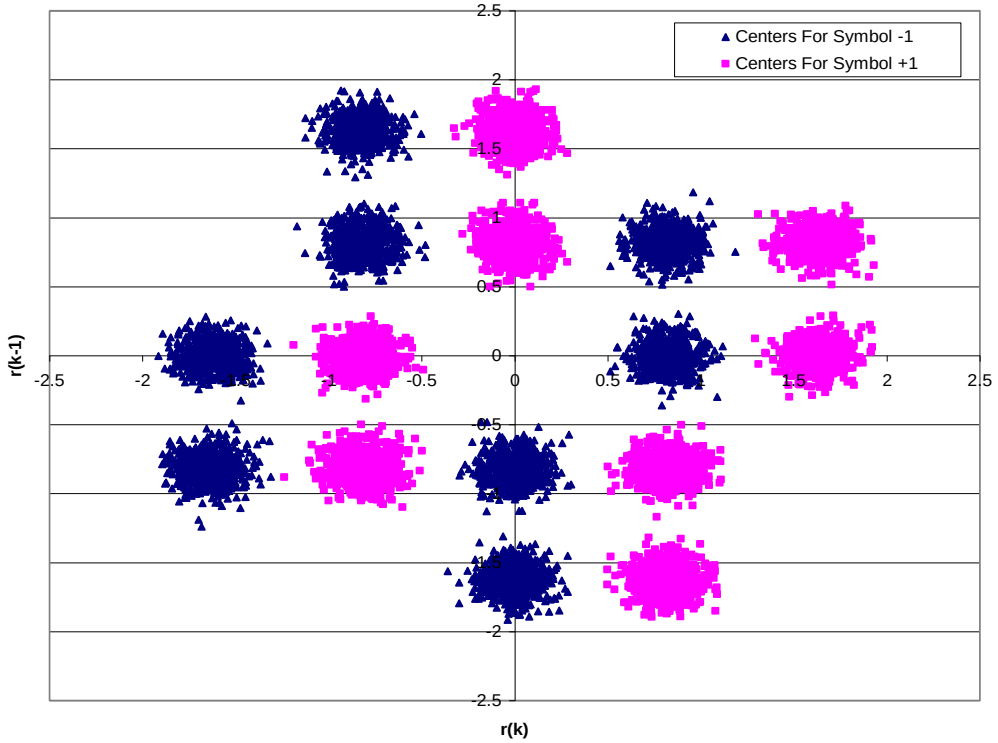

Fig. 3.7 Channel state diagram for $H_7(z)$ ***with*** $d = 0$ ***and*** $SNR = 20dB$

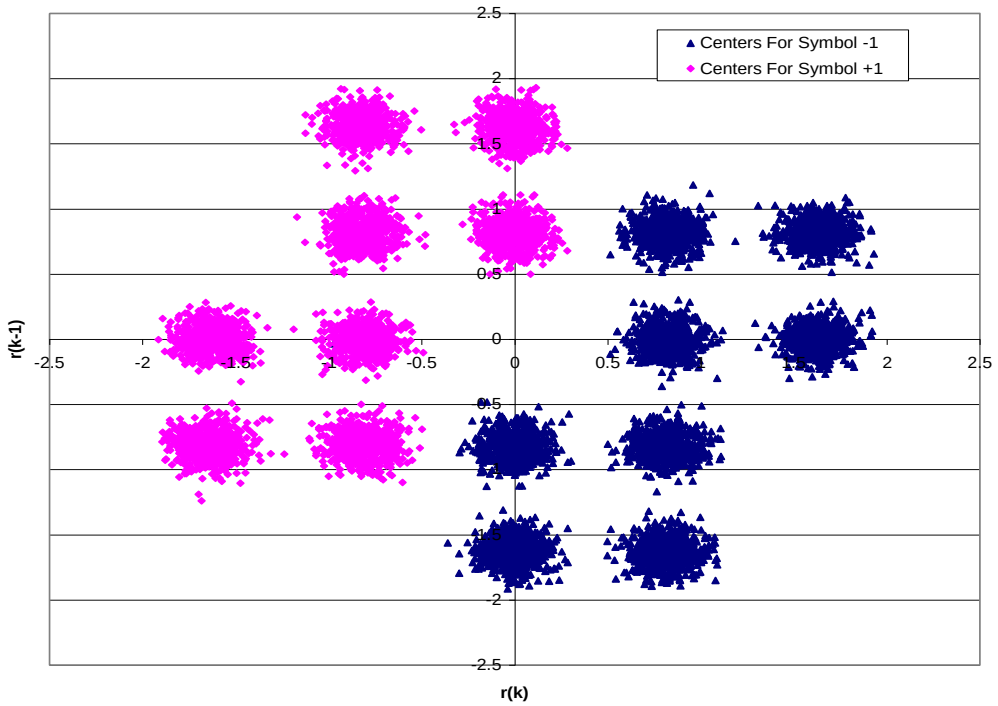

Fig. 3.8 Channel state diagram for $H_7(z)$ ***with*** $d = 1$ ***and*** $SNR = 20dB$

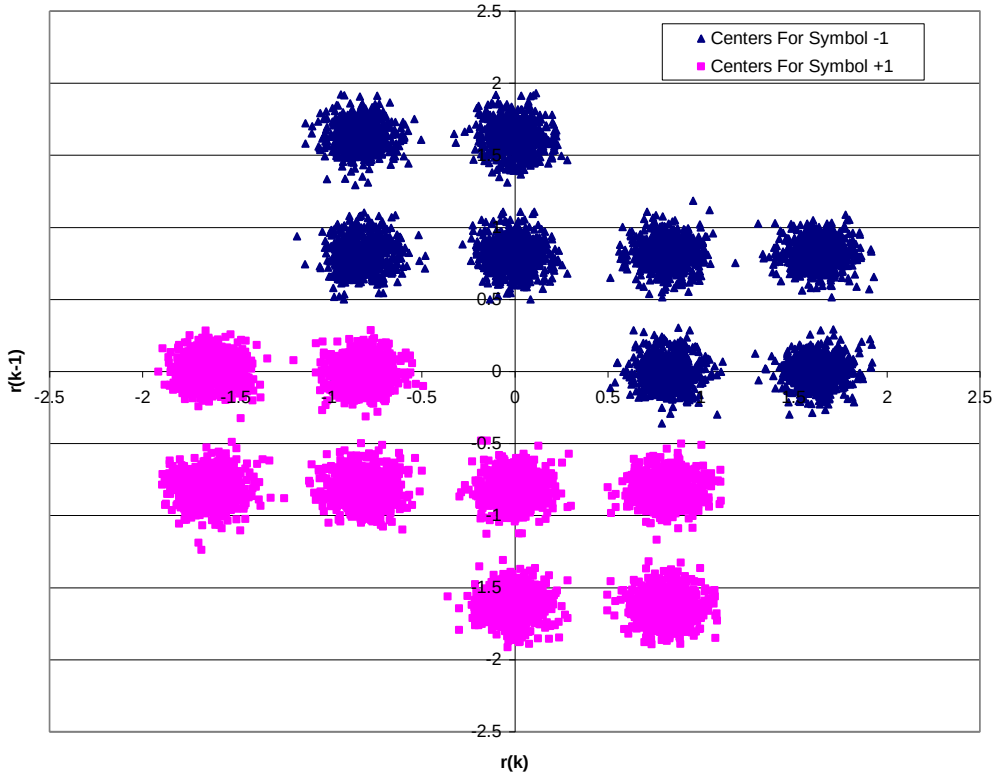

Fig. 3.9 Channel state diagram for $H_7(z)$ ***with*** $d = 2$ ***and*** $SNR = 20dB$

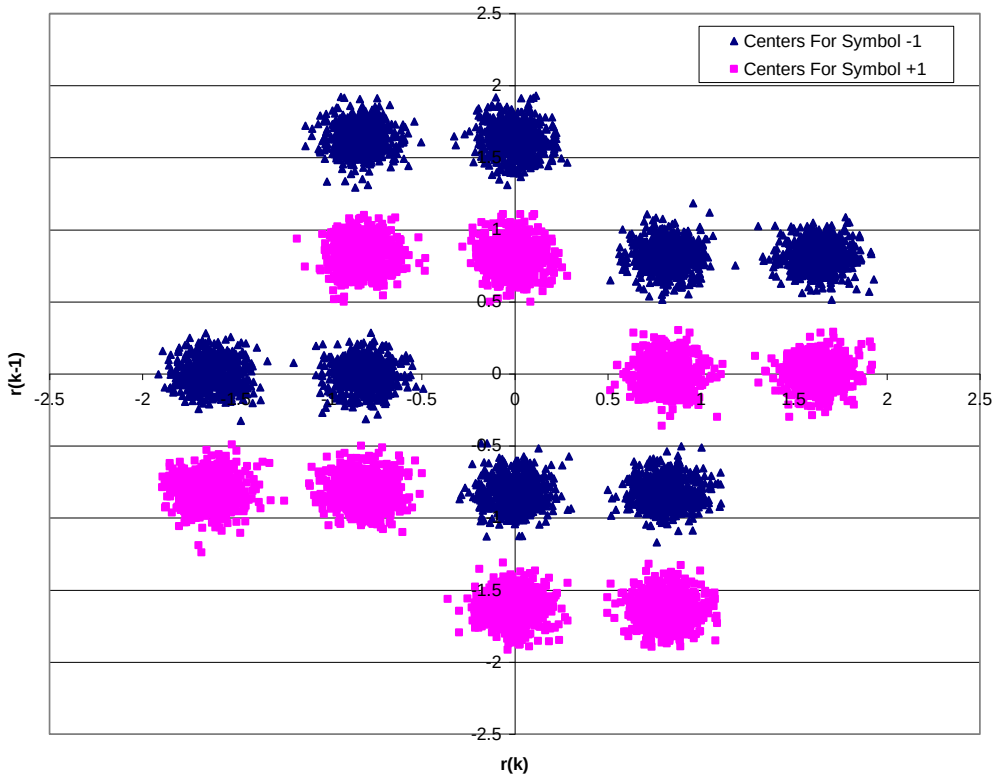

Fig. 3.10 Channel state diagram for $H_7(z)$ ***with*** $d = 3$ ***and*** $SNR = 20dB$

The channel which has been considered for plotting the above figures 3.7-3.10, has a transfer function given by

$$H_7(z) = 0.407 - 0.815\,z^{-1} - 0.407\,z^{-2}$$

The channel state diagrams for the channel $H_7(z)$ are drawn for the delay values of $d = 0,1,2,3$ in Fig.3.7, 3.6, 3.9, 3.10 respectively. It can be noted from the above figures that for the delay values of $d = 1,2$ classification will be easier compared to those for delay $d = 0,3$. Hence certainly a superior performance is expected with the former delay values. This can be seen in Fig. 3.11. there is a superior performance for delay values of $d = 1,2$.

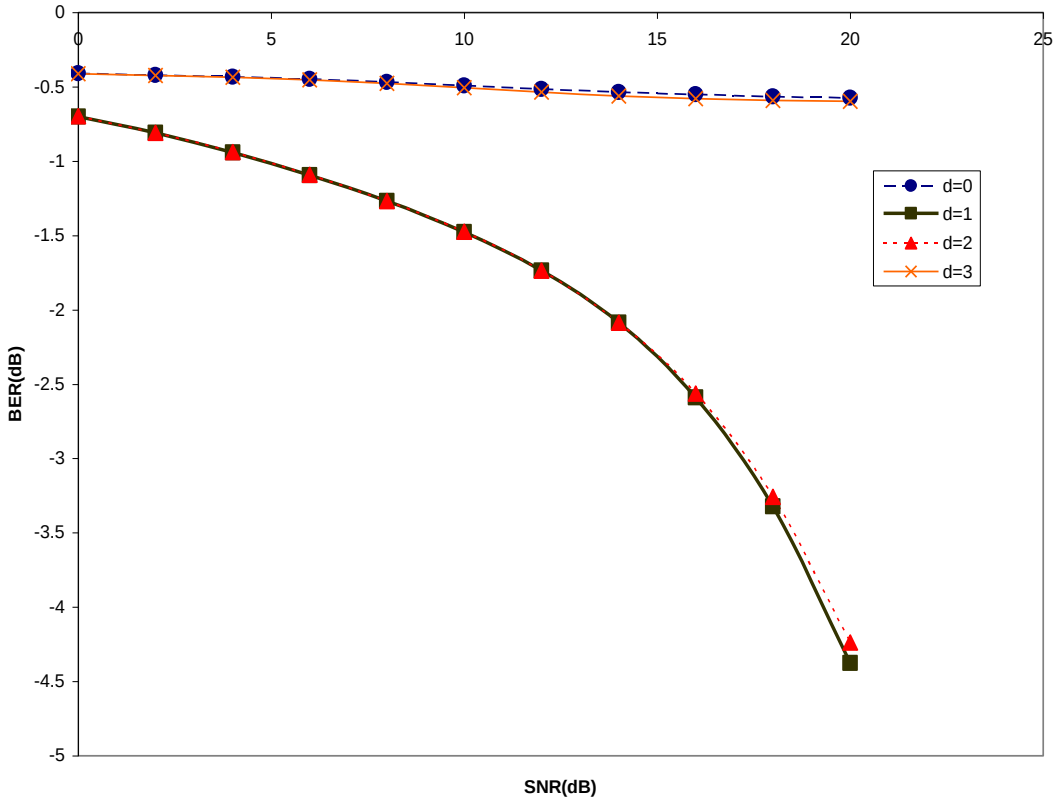

Fig. 3.11 BER performance comparison for varying decision delays for $H_7(z)$

Here a $\langle 5,1 \rangle$ FNN equaliser has been considered with out decision feedback with an equaliser order $m = 2$.

However, the parameters m, d, n_b are interdependent because, as explained earlier in section 3.2, an equaliser order of $m = d + 1$ is sufficient, any value more than this will not give an appreciable increase in performance. From Eqn. 2.8,

$$n_b = n_a + m - 2 - d \qquad (3.1)$$

Hence substituting $m = d + 1$ in above equation we get,

$$n_b = n_a - 1 \qquad (3.2)$$

i.e. a proper combination of these parameters, for a chosen channel order, is required to obtain a better performance (near optimal) of the equaliser.

Chapter 4

RLS BASED BP ALGORITHM

Introduction to RLS

Neural Network Training using RLS

RLS BASED BP ALGORITHM

In this chapter, the novel method of training the neural network using RLS algorithm is introduced. In section 4.1, a brief introduction to the RLS algorithm is given and in section 4.2 the proposed method of training neural networks using RLS is explained and the corresponding algorithm is given.

4.1 INTRODUCTION TO RLS

RLS algorithm was actually introduced to improve the rate of convergence of the linear adaptive filters. This RLS algorithm is an extension to the method of least squares [1]. This algorithm was developed on the basis of a relation in matrix algebra known as "*Matrix Inversion Lemma"*. An important feature of this algorithm is that its rate of convergence is typically an order of magnitude faster than the simple LMS filter, due to the fact that the RLS filter whitens input data by using the inverse correlation matrix of the data, assumed to be of zero mean.

The recursive implementations of the method of least squares starts with prescribed initial conditions and use the information contained in new data samples to update the old estimates. Hence here the length of the observable data is variable. Accordingly, the cost function [1, 4] to be minimized can be expressed as

$$\xi(k) = \sum_{i=1}^{k} \lambda^{k-i} |e(i)|^2 \qquad (4.1)$$

where k is the variable length of the observable data, λ is a positive constant close to but less than unity and is called the *"Forgetting Factor"*. The use of this forgetting factor is intended to ensure that data in the distant past are "forgotten" in order to afford the possibility of the statistical variations of the observable data when the filter operates in a non-stationary environment. The special case of $\lambda = 1$ corresponds to the ordinary method of least squares.

4.1.1 Regularization

Least-squares estimation, like the method of least-squares, is an ill-posed inverse problem, in that the input data consisting of tap-input vector $\mathbf{y}(k)$ and the corresponding desired response $d(k)$ for varying k are given, and the requirement is to estimate the unknown parameter vector of a multiple regression model that relates $d(k)$ to $\mathbf{y}(k)$.

The ill-posed nature of least squares estimation is due to the following reasons:

- There is insufficient information in the input data to reconstruct the input-output mapping uniquely.
- The unavoidable presence of noise or imprecision in the input data adds uncertainty to the reconstructed input-output mapping.

To make the estimation problem "well posed", some form of prior information about the input-output mapping is needed. This, in turn, means that the formulation of the cost function must be expanded to take the prior information into account. To satisfy this objective, the cost function to be minimized can be written as,

$$\xi(k) = \sum_{i=1}^{k} \lambda^{k-i} |e(i)|^2 + \theta\lambda^k \|\mathbf{w}(k)\|^2 \tag{4.2}$$

The second term in the summation is called the regularization term and θ is a positive real number called the "*Regularization Parameter"*. Except for the factor $\theta\lambda^k$, the regularization term depends on the tap weight vector $\mathbf{w}(n)$.

Hence the $M \times M$ time-average correlation matrix of the input vector $\mathbf{y}(i)$ can be given by

$$\Phi(k) = \sum_{i=1}^{k} \lambda^{k-i} \mathbf{y}(i)\mathbf{y}^H(i) + \theta\lambda^k \mathrm{I} \tag{4.3}$$

In the above equation, I is the $M \times M$ identity matrix. It can be noted that the addition of the regularization term has the effect of making the correlation matrix $\Phi(k)$ nonsingular at all stages of the computation, starting from $k = 0$.The weight update in the RLS algorithm is given by

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \mathbf{g}(k)\xi^*(k) \tag{4.4}$$

Where $\mathbf{g}(k)$ is called the gain vector, which is given by

$$\mathbf{g}(k) = \Phi^{-1}(k)\mathbf{y}(k) \tag{4.5}$$

Hence the weight update requires the calculation of the inverse of the correlation matrix $\Phi(k)$.

4.1.2 The Matrix Inversion Lemma

Let **A** and **B** be two positive-definite $M \times M$ matrices related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \tag{4.4}$$

Where **D** is a positive-definite $N \times M$ matrix and **C** is an $M \times N$ matrix. According to the matrix inversion lemma, the inverse of the matrix **A** can be expressed as

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B} \tag{4.5}$$

The main advantage of using this concept is that, it helps update weights with out actually determining the inverse of the correlation matrix. It instead updates the gain vector itself. Here we first declare some matrix **P** as the inverse of the correlation matrix. We initialize it with some small values and then update it in each iteration. Using this updated inverse correlation matrix we determine the gain vector $\mathbf{g}(k)$ in each iteration and use it in updating the weights.

4.1.3 Importance of RLS

a. In the LMS algorithm, the correction that is applied in updating the old estimate of the coefficient vector is based on the instantaneous sample value of the tap-input vector and the error signal. On the other hand, in the RLS algorithm the computation of this correction utilizes all the past available information.

b. In the LMS algorithms, the correction applied to the previous estimate consists of the product of three factors: the (scalar) step-size parameter η, the error signal $e(k-1)$, and the tap-input vector u(*n*-1). On the other hand, in the RLS algorithm this correction consists of the product of two factors: the true estimation error $\xi(k-1)$ and the gain vector $\mathbf{g}(k)$. The gain vector itself consists of $\Phi^{-1}(k)$, the inverse of the deterministic correlation matrix, multiplied by the tap-input vector $\mathbf{y}(k)$. The major difference between the LMS and RLS algorithms is therefore the presence of $\Phi^{-1}(k)$ in the correction term of the RLS algorithm that has the effect of de-correlating the successive tap inputs, thereby making the RLS algorithm *self-orthogonalizing.* Because of this property, we find that the RLS algorithm is essentially independent of the eigen value spread of the correlation matrix of the filter input.

c. The LMS algorithm requires approximately 20*M* iterations to converge in mean square, where *M* is the number of tap coefficients contained in the tapped-delay-line filter. On the other band, the RLS algorithm converges in mean square within less than 2*M* iterations. The rate of convergence of the RLS algorithm is therefore, in general, faster than that of the LMS algorithm by an order of magnitude.

d. Unlike the LMS algorithm, there are no approximations made in the derivation of the RLS algorithm. Accordingly, as the number of iterations approaches infinity, the least-squares estimate of the coefficient vector approaches the optimum Wiener value, and correspondingly, the mean-square error approaches the minimum value possible.

In other words, the RLS algorithm, in theory, exhibits zero mis-adjustment. On the other hand, the LMS algorithm always exhibits a nonzero mis-adjustment; however, this mis-adjustment may be made arbitrarily small by using a sufficiently small step-size parameter η.

4.2 NEURAL NETWORK TRAINING USING RLS

The method of training the neural network using the RLS algorithm is explained in this section. The RLS algorithm is used to update the synaptic weights of the neural networks. The proposed algorithm is almost similar to the Back Propagation (BP) algorithm except the step of weight update changes. This algorithm also has two passes, a forward pass and a backward pass. In the forward ass, the inputs are allowed to propagate through the network, till it reaches the output node. Then, at the output node, error is calculated and back propagated through the network. During the back propagation of error, the local gradient, δ_j^l, is computed at each node. The subscripts and superscripts used are already explained in chapter 2.

For using RLS to update the weights of the neural networks, each node in all the layers of the neural network should maintain a window of their past outputs. Also separate inverse correlation matrices and gain vectors need to be maintained at each node (including the input node and excluding the output node). The size of the window and the size of these matrices depend on the number of nodes in the next layer. Here the simple RLS algorithm is applied at each node.

4.2.1 Algorithm

The algorithm for the proposed method of training the neural network using the RLS algorithm is presented below:

a. Initialize the algorithm by setting small random values to all the synaptic weights.
b. Initialize the inverse correlation matrix $\mathbf{P}_i^l(0)$ at each node.

$$\mathbf{P}_i^l(0) = \theta^{-1} \cdot \mathbf{I} \tag{4.6}$$

where i represents the i^{th} node in layer l, this matrix is of the order $N_{l+1} \times N_{l+1}$ and N_{l+1} represents number of nodes in layer $(l+1)$, and θ is a small constant.

c. Initialize $n = 1$ and continue steps d-j for maximum number of iterations.
d. Forward propagation of the inputs, which is same as in BP algorithm.

e. Compute the error e at the output.

f. Back propagate the error to calculate the local gradient $\delta_i^l(n)$ at each node which is same as in BP algorithm.

g. Compute an intermediate matrix $\pi_i^l(n)$.

$$\pi_i^l(n) = \mathbf{P}_i^l(n-1) \cdot \mathbf{Y}_i^l(n) \tag{4.7}$$

Where $\mathbf{Y}_i^l(n)$ represents the windowed output at each node, the order of this matrix being $N_{l+1} \times 1$.

h. Compute the gain vector $\mathbf{K}_i^l(n)$, whose size is $N_{l+1} \times 1$.

$$\mathbf{K}_i^l(n) = \frac{\pi_i^l(n)}{(\lambda + \mathbf{Y}_i^{l^T}(n) \cdot \pi_i^l(n))} \tag{4.8}$$

Where the constant λ is called the forgetting factor, the superscript T represents the transpose of the matrix.

i. Update the weights.

$$w_{ji}^l(n) = w_{ji}^l(n-1) + K_{ji}^l(n) \cdot \delta_j^{l+1}(n) \tag{4.9}$$

Where $K_{ji}^l(n)$ is a scalar and gives the j^{th} value of the gain vector $\mathbf{K}_i^l(n)$

j. Update the inverse correlation matrix.

$$\mathbf{P}_i^l(n) = \lambda^{-1} \cdot \mathbf{P}_i^l(n-1) - \lambda^{-1} \cdot \mathbf{K}_i^l(n) \cdot \mathbf{Y}_i^{l^T}(n) \cdot \mathbf{P}_i^l(n-1) \tag{4.10}$$

Chapter 5

TABU BASED NEURAL NETWORK TRAINING

Tabu Search Algorithm

Weight Adaptation using TS

Slope Adaptation using TS

TABU BASED NEURAL NETWORK TRAINING

In this chapter, first an introduction to the TABU search (TS) algorithm is given and then the proposed algorithms for training the neural network using TS id presented.

5.1 TABU SEARCH ALGORITHM

The roots of tabu search go back to the 1970's; it was first presented in its present form by Glover [13]. Additional efforts of formalization are reported in [14]. Many computational experiments have shown that tabu search has now become an established optimization technique which can compete with almost all known techniques and which - by its flexibility - can beat many classical procedures. Up to now, there is no formal explanation of this good behavior.

A huge collection of optimization techniques have been suggested by a crowd of researchers of different fields; infinity of refinements has made these techniques work on specific types of applications. All these procedures are based on some common ideas and are furthermore characterized by a few additional specific features. Among the optimization procedures the iterative techniques play an important role: for most optimization problems no procedure is known in general to get directly an "optimal" solution.

The general step of an iterative procedure consists in constructing from a current solution *i* a next solution *j* and in checking whether one should stop there or perform another step. Neighborhood search methods are iterative procedures in which a neighborhood $N(i)$ is defined for each feasible solution *i*, and the next solution *j* is searched among the solutions in the neighborhood $N(i)$.

The most famous neighborhood search method which has been used for finding an approximation to the minimum value of a real-valued function *f* on a set ***S*** is the descent method. It is outlined below :

Descent method

1. Choose an initial solution *i* in ***S.***
2. Find a best *j* in $N(i)$ (i.e. such that $f(i) \leq f(k)$ for any *k* in $N(i)$).
3. If $f(i) \geq f(k)$ then stop. Else set $i = j$ and go to Step 2.

Such a method clearly may stop at a local but not global minimum of *f*. In general, $N(i)$ is not defined explicitly: *j* is searched by exploring some directions from *i* (for instance the coordinate axes).

Simulated annealing and tabu search can be considered as neighborhood search methods which are more elaborate than the descent method. The basic ingredients of tabu search are described in the next section.

5.1.1 Basic ideas of Tabu Search

In order to improve the efficiency of the exploration process, one needs to keep track not only of local information (like the current value of the objective function) but also of some information related to the exploration process. This systematic use of *memory* is an essential feature of tabu search (TS). Its role will be emphasized later on. While most exploration methods keep in memory essentially the value $f(i^*)$ of the best solution i^* visited so far, TS will also keep information on the itinerary through the last solutions visited [19]. Such information will be used to guide the move from *i* to the next solution *j* to be chosen in $N(i)$. The role of the memory will be to restrict the choice to some subset of $N(i)$ by forbidding for instance moves to some neighbor solutions.

More precisely, it can be noticed that the structure of the neighborhood $N(i)$ of a solution *i* will in fact be variable from iteration to iteration. It would therefore be more appropriate to include TS in a class of procedures called *dynamic neighborhood search techniques.*

Formally let us consider an optimization problem in the following way: given a set ***S*** of feasible solutions and a function $f : S \to \mathbf{R}$, find some solution i^* in ***S*** such that $f(i^*)$ is acceptable with respect to some criterion (or criteria). Generally a criterion of acceptability for a solution i^* would be to have $f(i^*) \leq f(i)$ for every *i* in ***S***. In such a situation TS would be an exact minimization algorithm provided the exploration process would guarantee that after a finite number of steps such an *i** would be reached.

In most contexts however no guarantee can be given that such an i^* will be obtained; therefore TS could simply be viewed as an extremely general heuristic procedure. Since TS will in fact include in its own operating rules some heuristic techniques, it would be more appropriate to characterize TS as a *metaheuristic*. Its role will most often be to guide and to orient the search of another (more local) search procedure.

As a first step towards the description of TS, we reformulate the classical descent method as follows:

a) Choose an initial solution *i* in ***S***.

b) Generate a subset V^* of solution in $N(i)$.

c) Find a best *j* in V^* (i.e. such that $f(j) \leq f(i)$ for any *k* in V^*) and set $i = j$.

d) If $f(j) \geq f(i)$ then stop. Else go to Step 2.

In a straightforward descent method generally $V^* = N(i)$ is take. However this may often be too time-consuming; an appropriate choice of *V** may often be a substantial improvement.

The opposite case would be to take $|V^*| = 1$; this would drop the phase of choice of a best *j*. A solution *j* would be accepted if $f(j) \leq f(i)$, otherwise it would be accepted with a certain probability depending upon the values of *f* at *i* and *j*..

The choice of V^* will be crucial; in order to define it at each step one will use systematically memory to exploit knowledge extending beyond the function *f* and the neighborhood $N(i)$.

Except for some special cases of convexity, the use of descent procedures is generally frustrating since it is likely to be trapped in a local minimum which may be far (with respect to the value of *f*) from a global minimum.

So any iterative exploration process should in some instances accept also non-improving moves from *i* to *j* in V^* (i.e. with $f(j) > f(i)$) if one would like to escape from a local minimum. Simulated annealing does this also, but it does not guide the choice of *j*, TS in contrast chooses a best *j* in V^*.

As soon as non-improving moves are possible, the risk of visiting again a solution and more generally of cycling is present. This is the point where the use of memory is helpful to forbid moves which might lead to recently visited solutions. If such a memory is introduced we may consider that the structure of $N(i)$ will depend upon the itinerary and hence upon the iteration *k*; so we may refer to $N(i,k)$ instead of $N(i)$. With these modifications in mind, an improvement of the descent algorithm is formalized in a way which will bring it closer to the general TS procedure. It could be stated as follows (i^* is the best solution found so far and *k* the iteration counter):

a. Choose an initial solution *i* in *S*. Set $i^* = i$ and $k = 0$.

b. Set $k = k + 1$ and generate a subset V^* of solution in $N(i,k)$

c. Choose a best *j* in V^* (with respect to *f* or to some modified function $\tilde{f}$) and set $i = j$.

d. If $f(i) < f(i^*)$ then set $i^* = i$.

e. If a stopping condition is met then stop. Else go to Step b.

It can be observed that the classical descent procedure is included in this formulation (the stopping rule would simply be $f(i) \geq f(i^*)$ and i^* would always be the last solution). Notice also that the use of a modified $\tilde{f}$ instead of *f* is consider in some circumstances.

In TS some immediate stopping conditions could be the following:

- $N(i, k+1) = \varnothing$
- k is larger than the maximum number of iterations allowed
- The number of iterations since the last improvement of i^* is larger than a specified number
- Evidence can be given than an optimum solution has been obtained.

While these stopping rules may have some influence on the search procedure and on its results, it is important to realize that the definition of $N(i, k)$ at each iteration k and the choice of V^* are crucial.

5.1.2 Tabu Search applied to Neural Networks

The popularity of TS as a global optimization technique has attracted the concentration of neural network engineers. Earlier, TS was just limited to the Operation Research community and was familiar for applications of combinatorial problems such as Traveling sales man problem, design optimization, quadratic assignment problem, etc. and now there have been attempts to use it for the continuous problems. In this section, the application of TS for the more difficult problem of optimization of neural networks [20] is discussed.

The TS algorithm explained in the previous section can be applied to neural networks to optimize two sets of parameters. First being the synaptic weights, and the second being the set of slopes of the activation functions of different nodes.

The principal reason for adapting the slopes is based on the fact that the key contributor for the introduction of nonlinearity in the neural networks is the activation function. But in the traditional BP algorithm, the slopes of activation functions of all the nodes of all the layers in the network are constant. Due to this fact, the degree of nonlinearity introduced is fixed for all nodes. But it's a well known fact that some of the nodes are key players in the performance of the ANNs compared to others. Thus we need to choose proper values for these parameters to obtain a near optimal solution. Further, once the weights are adapted such that the error at the output node cannot be further minimized, reason being that it has been trapped in local minima, the error can be minimized by correcting the slopes.

Hence using the TS algorithm the (near) optimal set of slopes we can chosen for training the neural network.

In this method of slope adaptation using TS, the synaptic weights are trained using the BP algorithm. This process can be done in two ways. First method is to update the weights of the NN using BP algorithm, then fix them and use the TS to adapt the slopes. In the second method, both the weights and slopes are updated in each iteration. Here, the weights are updated using the local gradient δ_j^l, then using the TS choose the best set of slopes for minimizing the error in that iteration. Then all those best set of slopes, obtained during each iteration, are tested for minimum mean square error (MSE) to obtain the best set of slopes.

The total three algorithms of adapting the weights and slopes of the neural network can be called together as Tabu Based Back Propagation (TBBP) algorithms. Use of TBBP not only helps us to obtain a superior solution but also reduces the structural complexity of the neural network by reducing the number of neurons (nodes) required, and hence reducing the number of adjustable parameters. The algorithms are explained in the next sections.

5.2 WEIGHT ADAPTATION USING TS

This method of updating the weight update can be divided into two steps - the Superficial Search (SS) and the Deep Search (DS). In the SS we search for the solution which has the higher probability of finding good global solutions. The DS trains these solutions, found in SS, to find the best solution in the neighborhood of the solution.

The original weight $\mathbf{W}_0$ is randomly generated, where $\mathbf{W}_0$ include all the weights of the neural network. The SS trains this original weight to a state, $\mathbf{W}_0^{'}$ a point in the local minima, but is not at the bottom of the concave [4]. If the point is in tabu list (TL), then this is considered to be in a searched concave and is not considered for DS.

There may be some $\mathbf{W}_i^{'}$s, where $i = 0,1,2,\ldots$ which may be in the searched concave but may satisfy

$$E(\mathbf{W}_i^{'}) < (1 - AC)E(\mathbf{W}_b) \text{ or}$$

$$E(\mathbf{W}_i^{'}) < (1 + AC)E(\mathbf{W}_b) \qquad (5.1)$$

where $E(\mathbf{W}_i^{'})$ is the sum of error evaluated at the superficial state $\mathbf{W}_i^{'}$ and $\mathbf{W}_b$ is the best solution found so far. The above equation is called the AC and is used to activate some of the tabued solutions. The solution obtained in the SS is further searched, called DS, in its neighborhood.

Algorithm

Here the basic steps involved in the algorithm are discussed. It mainly consists of 7 steps.

a. Generate an initial solution $\mathbf{W}_i$, $i = 0,1,2\ldots$

b. Superficial search:

   i. The initial weight is trained with BP algorithm to obtain a Superficial state $\mathbf{W}_i^{'}$ and $E(\mathbf{W}_i^{'})$.If this solution is in TL and does not satisfy AC then go to step (a) to generate new solution.

   ii. Else add the solution to the TL and go to step (c) for DS.

c. Deep Search:

   iii. Deeply search $\mathbf{W}_i^{'}$ and get the corresponding deep state $\mathbf{W}_i^{''}$ and get its corresponding $E(\mathbf{W}_i^{''})$.

   iv. If $E(\mathbf{W}_i^{''}) < E(\mathbf{W}_b)$ then set $\mathbf{W}_b = \mathbf{W}_i^{''}$ and $E(\mathbf{W}_b) = E(\mathbf{W}_i^{''})$.

d. Generate a new solution $\mathbf{W}_{ij}^{'}$ in the neighborhood of $\mathbf{W}_i^{'}$ and evaluate $E(\mathbf{W}_{ij}^{'})$.

e. If this new superficial solution $\mathbf{W}_{ij}^{'}$ is in TL and does not satisfy AC then go to step (d) to generate a new neighbor, else add $\mathbf{W}_{ij}^{'}$ to TL and go to step (f).

f. Deeply search $\mathbf{W}_{ij}^{'}$ similar to step (c) and update the best solution if the squared error obtained in this deep search is less than the best error square till now.

g. If j is less than maximum number of neighborhood searches go to step (d). Or else finalize $\mathbf{W}_b$ as the best solution.

The superior performance, in terms of BER, of this algorithm can be noted in next chapter, where the number of decision errors is compared with that of BP algorithm. The reduction in structural complexity is also explained.

5.3 SLOPE ADAPTATION USING TS

As mentioned earlier, the concept of adaptation of slopes of the activation functions can be implemented in two ways. Algorithm-1 describes the first method in which the weights are adapted first using BP and then fixing these we use TS for slope adaptation. Algorithm-2 describes the second method of adapting the slopes, where both the synaptic weights and the slopes are adapted in each iteration.

This method of slope adaptation also requires maintaining the Tabu List (TL), to keep track of the regions already searched. We start by choosing some random values to slopes. It is similar to weight adaptation which includes two parts- Deep Search (DS) and Superficial Search (SS).

In this process an initial set of weights and slopes are randomly generated. The initial set of slopes is given by $\Phi(0)$. In the SS we will obtain the best set of slopes $\Phi_b(0)$. Then this solution is further searched using the DS. In the DS we generate random solutions in its neighborhood and the squared error is calculated and the best solution is chosen.

Algorithm-1

The basic steps involved in the algorithm are

1. Generate an initial set of synaptic weights $\mathbf{W}(0)$ and initialize the slopes of all the activation functions to unity.
2. Update the slopes using BP algorithm as explained in section 2.2
3. Now freeze the weights. And start the Superficial Search:
    i. Initialize some variable, $x = 0$
    ii. Now, assign some random values, within the chosen range, to the set of slopes $\Phi(i)$, $i = 0,1,2\ldots$
    iii. Compute $E(\Phi(i))$, i.e. squared error.
    iv. If this solution is in TL and if $x \neq 0$ or if $E(\Phi(i)) > E(\Phi_b(i))$ then increment x go to step 3(ii).
    v. Else add $\Phi(i)$ to TL and set $\Phi_b(i) = \Phi(i)$ and $E(\Phi_b(i)) = E(\Phi(i))$.
    vi. If x is less than MAX number of iterations, increment x and go to step 3(ii), else start DS
4. Now Deep search:
    i. Initialize a variable $x = 0$
    ii. Generate the new solution $\Phi_n(i)$ in the neighborhood of $\Phi_b(i)$

iii. Compute $E(\Phi_n(i))$.

iv. If the solution is in TL or if $E(\Phi_n(i)) > E(\Phi_b(i))$ increment x and go to step 4(ii).

v. Else set $\Phi_b(i) = \Phi_n(i)$ and $E(\Phi_b(i)) = E(\Phi_n(i))$

vi. If x is less than MAX number of iterations, increment x and go to step 4(ii), else go to step 5.

5. If i is less than maximum neighborhood searches, go to step 3.
6. Else choose the best among all the best solution obtained.

The results obtained using this algorithm is explained in the next chapter. The algorithm has some practical limitations. It is not so suitable for real time applications as it needs to maintain a memory for the training data to compute the error square its DS. Hence a more useful algorithm is formulated which will be more useful for real time applications. In this algorithm starting from the time instant $k = 0$, the mean square error is computed over the data received till the present instant. The algorithm can be explained as follows.

Algorithm-2

The basic steps of this algorithm is explained below.

1. Generate an initial set of synaptic weights $\mathbf{W}(0)$ and slopes $\Phi(0)$.
2. Forward propagate the input sequence and compute the error at the output node.
3. Back propagate the error and compute the local gradient at each node to get the vector $\boldsymbol{\delta}(i)$, which is the set of local gradients of all the nodes at the iteration i.
4. Update the weight using weight update equation

$$\mathbf{W}(i) = \mathbf{W}(i-1) + \eta * \boldsymbol{\delta}(i) * \mathbf{y}(i) \qquad (5.2)$$

where $\mathbf{y}(i)$ is the set of output vector of all the nodes (including input layer)

5. Deep Search:
   i. Initialize some variable $x = 0$
   ii. Generate a new solution $\Phi_n(i)$ in the neighborhood of $\Phi(i)$
   iii. Compute the mean square error $E(\Phi_n(i))$ using the data received till now.
   iv. If $\Phi_n(i)$ is in the TL and if $x \neq 0$ or if $E(\Phi_n(i)) > E(\Phi_b(i))$, increment x and go to step 5(ii).

v. Else add $\Phi_n(i)$ to TL and set $\Phi_b(i) = \Phi_n(i)$ and $E(\Phi_b(i)) = E(\Phi_n(i))$

vi. If x is less than MAX number of neighborhood searches, increment x and go to step 5(ii), or else go to step 6.

6. If i is less than maximum number of iterations, increment i and go to step 2, else to step 7.
7. Choose the best solution among all the $\Phi_b(i)$'s.

As mentioned earlier, this algorithm is more suitable for practical applications, as it uses the present data received to update the weights and also to compute the mean square error in the TS process to adapt the weights. In the next chapter, the results corresponding to the three algorithms explained in this chapter are discussed.

Chapter 6

RESULTS AND DISCUSSION

RLS based BP algorithm

Tabu based Weight Adaptation

Tabu based Slope Adaptation

RESULTS AND DISCUSSION

In this chapter, the results obtained using the proposed algorithms of using RLS and TS for training the neural network are shown and the improvement in performance either in terms of convergence or bit error rate are discussed. In section 6.1 the results using the RLS algorithm for training ANNs are presented. In section 6.2, the behavior of the tabu based weight updating is discussed and in the last section, 6.3, the advantage of using the TS for slope adaptation is explained. All the programs are written in C and compiled using Microsoft VC++ ver.6.0 and the plots are taken using Microsoft Excel-2003.

6.1 RLS BASED BP ALGORITHM

In this experiment, it is shown that when RLS algorithm is used to update the weights of a neural network, the rate of convergence is faster compared to training using simple BP algorithm. The channel, considered is having the transfer function

$$H_6(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$$

Here, the neural network structure used is $\langle 5,1 \rangle$ FNN with an equalizer order, $m = 4$ and the decision delay $d = 3$. No feedback is employed in this experiment. The convergence plots, which is the plot of mean square error against the iteration number, for the channel is shown in Fig. 6.1 for the two cases of RLS based training and BP algorithm. The plots are taken for 1000 iterations. It can be noted from these convergence plots that the proposed algorithm provides a rate of convergence faster than the simple BP algorithm. Hence the use of this algorithm reduces the training time and hence very much suitable for high speed digital communication systems.

When a neural network is trained for smaller number of iteration, then the RLS based training will have a superior performance, in terms of BER, compared to the case of training using BP algorithm. This is shown in Fig. 6.2, where the neural network is trained for 150 iterations and tested using 10^6 samples. From the figure it can be noted that there is an improvement of over 1.5dB in terms of BER at 20dB SNR.

But when we train the neural network for more number of iteration then there may not be much improvement in performance. The reason being that both the BP algorithm and the RLS based neural network training algorithms are derived from the Gradient-based learning algorithms. Hence it too have the problem of getting trapped in the local minima. Fig. 6.3 shows that the BP algorithm needs more iterations to achieve the same performance.

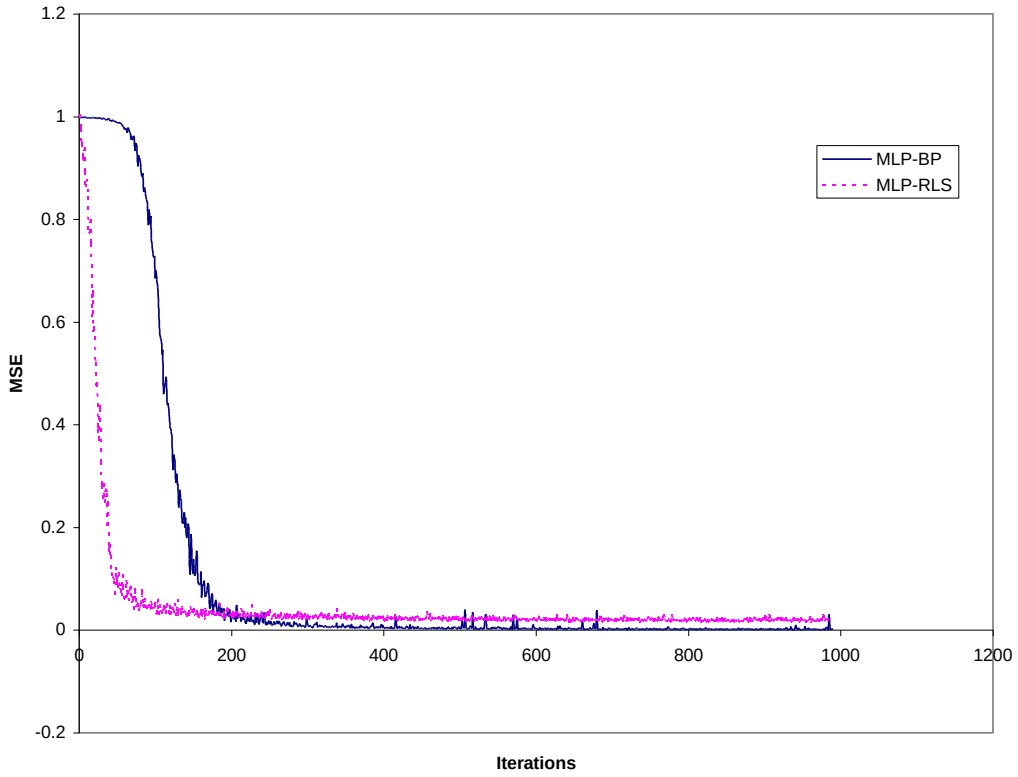

Fig. 6.1 Convergence plots of $H_6(z)$ for BP and RLS algorithms

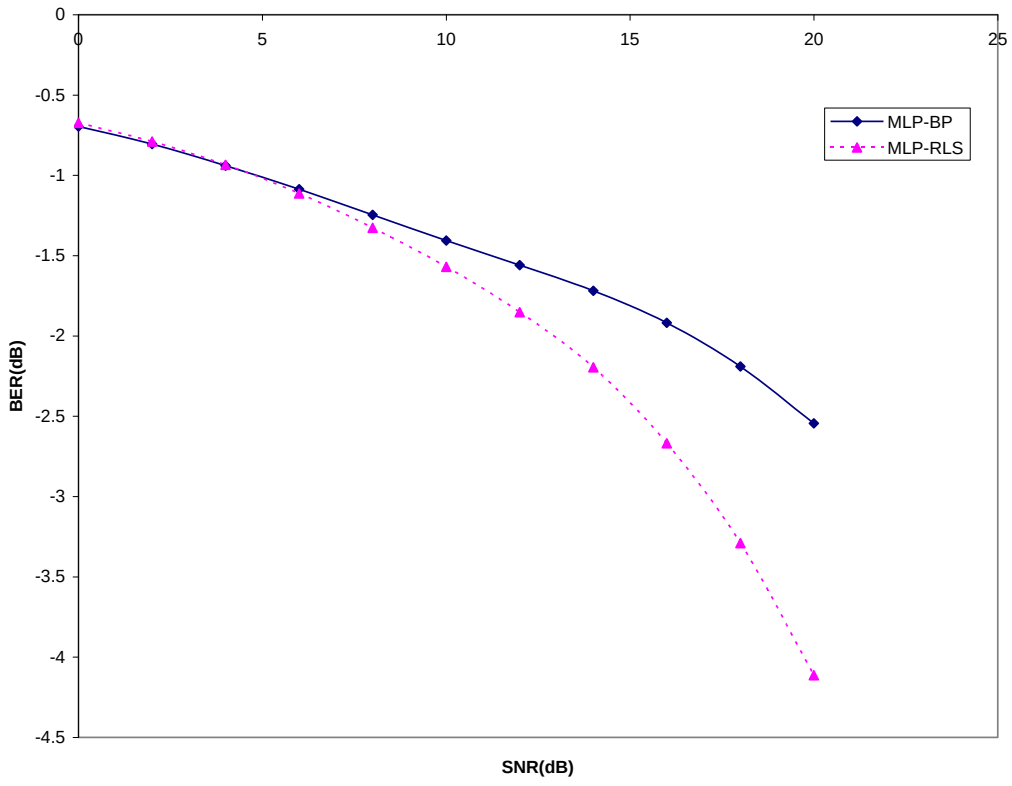

Fig. 6.2 SNR Vs BER plots of $H_6(z)$ for RLS and BP when trained for 150 iterations

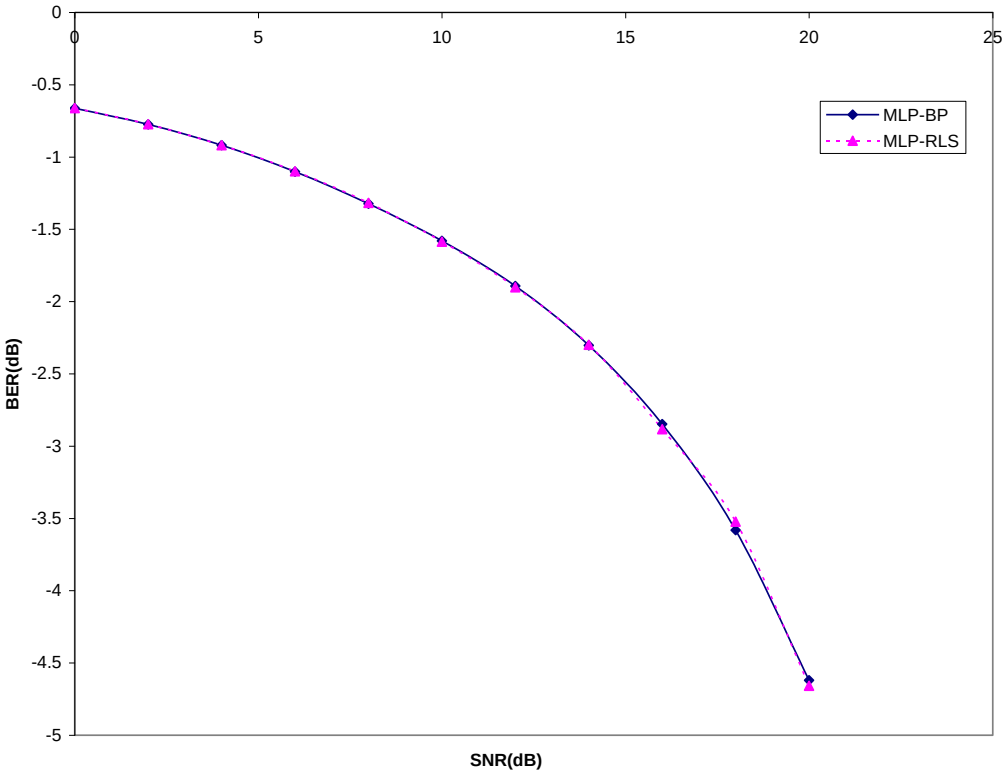

Fig. 6.3 SNR Vs BER of $H_6(z)$ ***for BP trained for 2000 iterations and RLS trained for 200 iterations***

6.2 TABU BASED WEIGHT ADAPTATION

As explained earlier, the Gradient-based training algorithm suffers from the problem of getting trapped in local minima. To overcome this problem, a global search technique is required to achieve the so called optimal solution. The use of TS for adapting the weighs of the neural network is considered. But the main concentration is to reduce the structural complexity of the neural network and reducing the number of parameters required to attain the similar performance. Here, in the experiment, the channel considered is having a transfer function given by

$$H_9(z) = 0.9413 + 0.3841\,z^{-1} + 0.5684\,z^{-2} + 0.4201\,z^{-3} + 1.0z^{-4}$$

A simple three layered neural network with decision feedback is considered. The equalizer order is chosen to have $m = 5$, feedback order $n_b = 4$, decision delay $d = 1$ and in the hidden layer only one node is considered to design an efficient and compact equalizer with an objective to appreciate the superior performance of TS even with this very small structure. In Fig. 6.4, the BER performance of the BP algorithm and that of Tabu based weight adaptation are compared. It can be noted that for the same structure of $\langle 1,1 \rangle$, the performance of BP algorithm is very poor compared to the proposed algorithm. There is an improvement of over 2.5dB in terms of BER at SNR of 20dB.

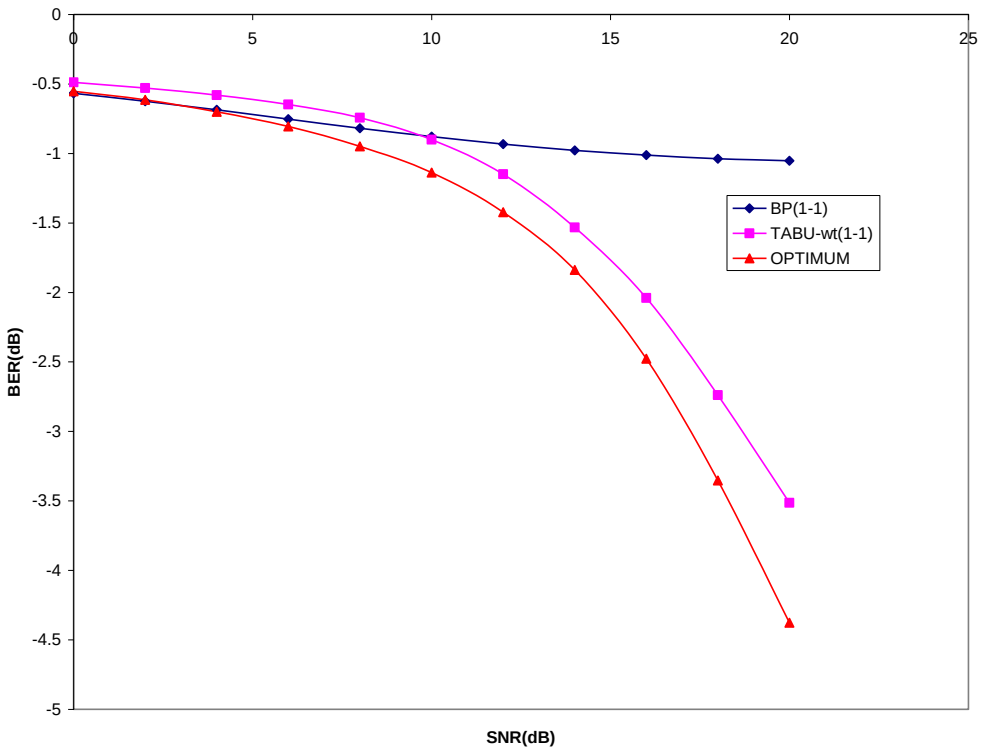

Fig. 6.4 SNR Vs BER for Tabu based weight adaptation and BP algorithms for same structure

To achieve the similar performance, the BP algorithm needs a more complex structure. This is shown in Fig. 6.5.

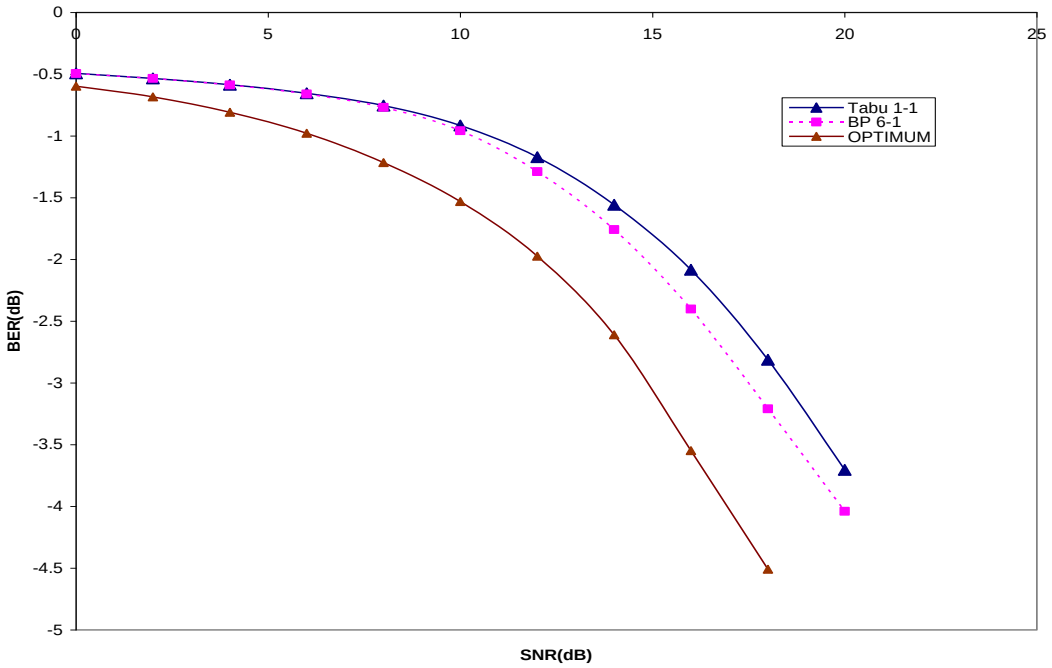

Fig. 6.5 SNR Vs BER for Tabu based Weight adaptation and BP algorithm for different structures.

A 1-1 structure for the neural network will have, for the chosen equalizer order of $m = 5$ and feedback order $n_b = 4$, $(5+4)\times 1+1=10$ synaptic weights. A neural network of structure 5-1, will have $(5+4)\times 6+6=60$ synaptic weights. Hence the proposed algorithm reduces the number of weights required from 60 to 10, to obtain the same performance.

6.3 TABU BASED SLOPE ADAPTATION

In this section, the results obtained for the novel method of training the neural network by adapting the slopes of the activation functions is discussed. As mentioned earlier slope can be adapted using TS in two ways as explained in previous chapter.

First, the results for algorithm-1 are described. The channel chosen for this experiment is the same one as used in previous section for adapting the weights of the neural networks. Here it is rewritten for convenience

$$H(z) = 0.9413 + 0.3841\, z^{-1} + 0.5684\, z^{-2} + 0.4201\, z^{-3} + 1.0 z^{-4}$$

The equalizer order, structure feedback order and other parameters are the same as explained in the previous section. Fig. 6.6 shows the improvement in performance by using the TS for slope adaptation.

It can be noted that there is an improvement of over 2dB in terms of BER at 20dB SNR. Now consider the case of algorithm-2 in which both the weights and slopes of the

neural network are adapted in each iteration. For this experiment also the same channel is considered with all the parameters as explained in section 6.1. Fig.6.7 shows the comparison of BER performance for this algorithm and the BP algorithm.

The main factors affecting the performance of the equalizer trained using both these algorithms of adapting the slope using TS, is the range of values chosen in which the optimum values of slopes are searched for using the TS approach. This is because, if the range is very less or the range in which the optimum solution may be present is missing, then there may not be any improvement in the performance. Even if the range is very large, there is a possibility of missing the optimal values within the given number of neighborhood searches or iterations. Hence the range of values must be properly chosen to obtain a superior performance. Here, in Fig. 6.6 and 6.7 the slopes are searched in the range, 0.5 to 3.0. Fig. 6.8 explains the effect of choosing different ranges for the slopes. The BER performance is compared for different slope ranges from 0.5 to 4.5. it can be noted that the plot for the range 1.5 to 3.0 has the better performance. But if we compare the performance with that of Fig. 6.7, where we have chosen the range from 0.5 to 3.0 it can be noted that the range of 1.5 to 3.0 is in fact not sufficient.

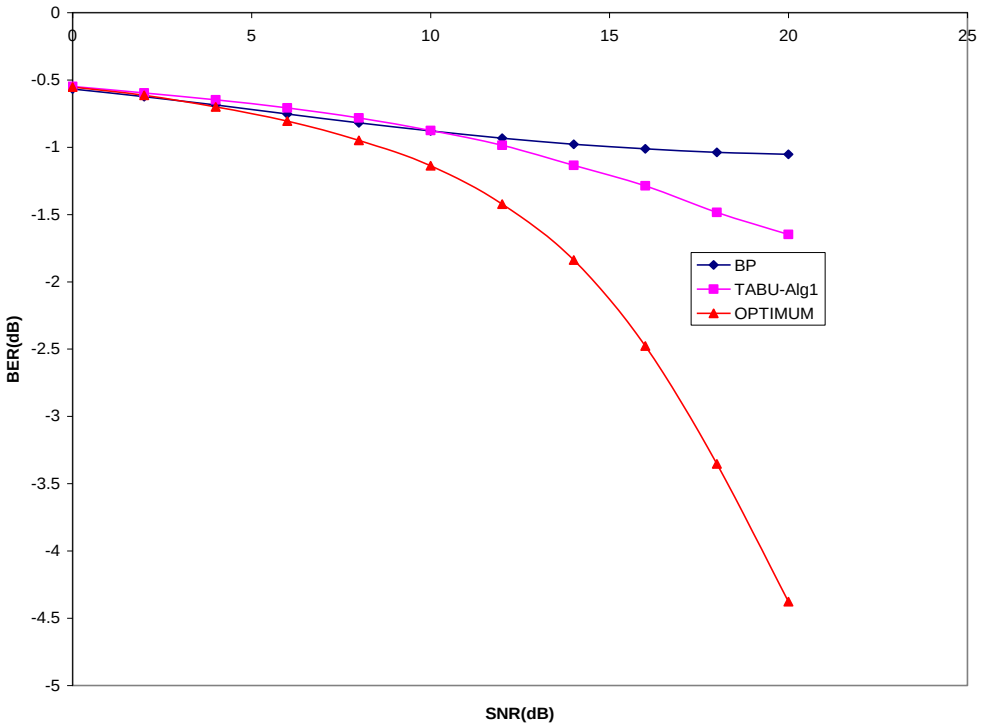

Fig.6.6 SNR Vs BER plot for Algorithm-1and BP algorithm.

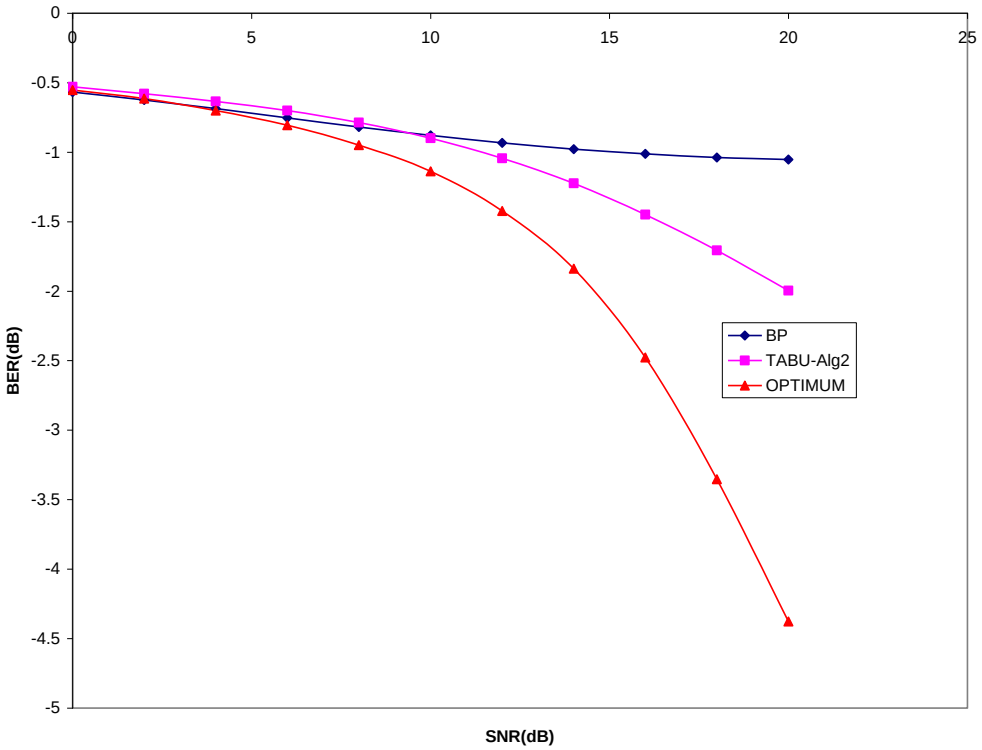

Fig. 6.7 SNR Vs BER plot for Algorithm-2 and BP algorithm

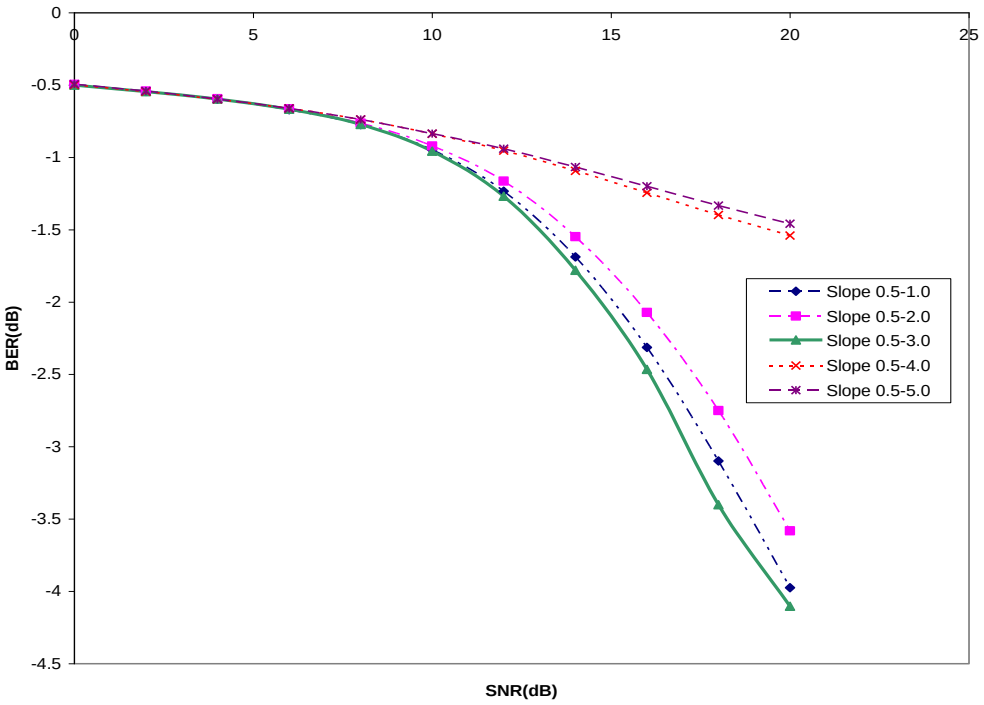

Fig. 6.8 SNR Vs BER plot for Algorithm-2 for varied slope ranges

The performance comparison for the three Tabu based neural network training is given in Fig.6.9.

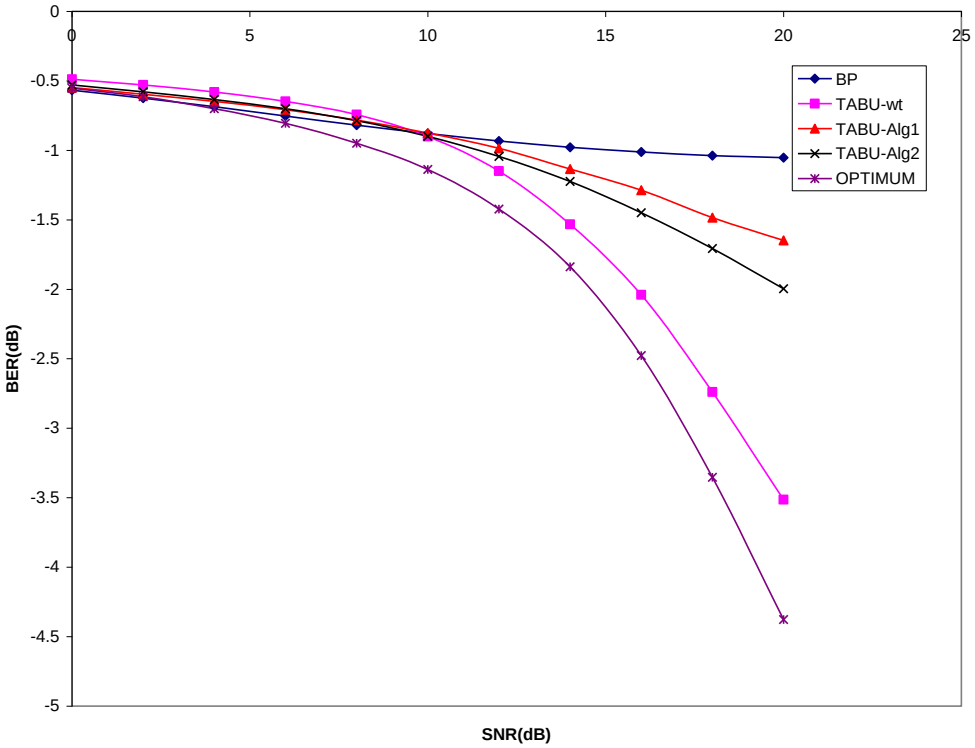

Fig. 6.9 SNR Vs BER plots for various Tabu based algorithms

In the Fig. 6.9, the curve represented using triangle is for the last algorithm, the curve shown in circular dots is for the case of weight updating using TS. The remaining curve, shown using rectangles is for algorithm-1 of adapting the slope using TS. It can be noted that among the three algorithms, the last one in which the slopes and weights are updated in each iteration in parallel, has a superior performance. And also as mentioned earlier the last algorithm is more suitable for real time applications. Hence among all the algorithms the last one is the best, provided the ranges for the slopes are properly chosen.

The main concentration is on the 5 tap channel, which was used to demonstrate the importance of Tabu based algorithms. The reason being that a simple Neural network structure trained using the BP algorithm is in many cases sufficient for 2 tap, 3 tap and 4 tap channels. This is shown in the following figures.

In Fig. 6.10, the performance of equalizer trained using the three proposed algorithms are compared with that of using BP algorithm for a simple 2 tap channel given by $H(z) = 1.0 + 0.5z^{-1}$ and the parameters chosen are $m = 2, n_b = 0, d = 0$.

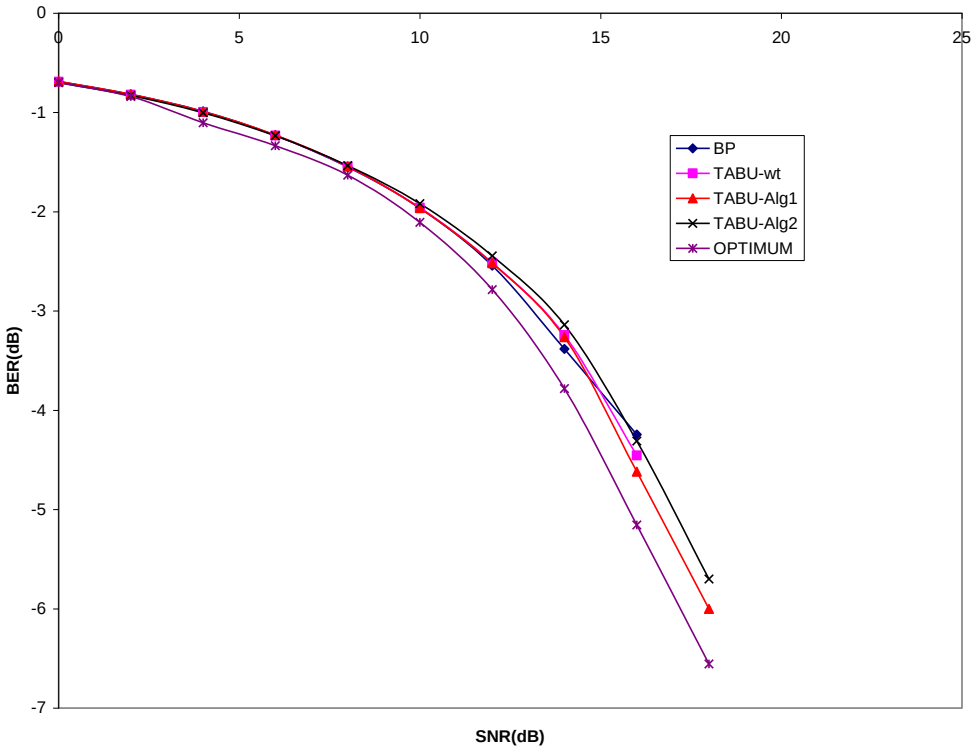

Fig. 6.10 SNR Vs BER for Tabu based algorithms and BP algorithm for 2-tap channel

In Fig. 6.11 the performance is compared in the cases of a 3-tap channel and in Fig. 6.12 the case of a 4-tap channel is considered. For the 3-tap channel having transfer function given by $H(z) = 0.4084 + 0.8164\, z^{-1} + 0.4084\, z^{-2}$ the parameters chosen are $m = 3, n_b = 2, d = 2$. And for the 4-tap channel having transfer function $H(z) = 0.35 + 0.8z^{-1} + 1.0z^{-2} + 0.8z^{-3}$, the parameters chosen are $m = 4, n_b = 3, d = 3$.

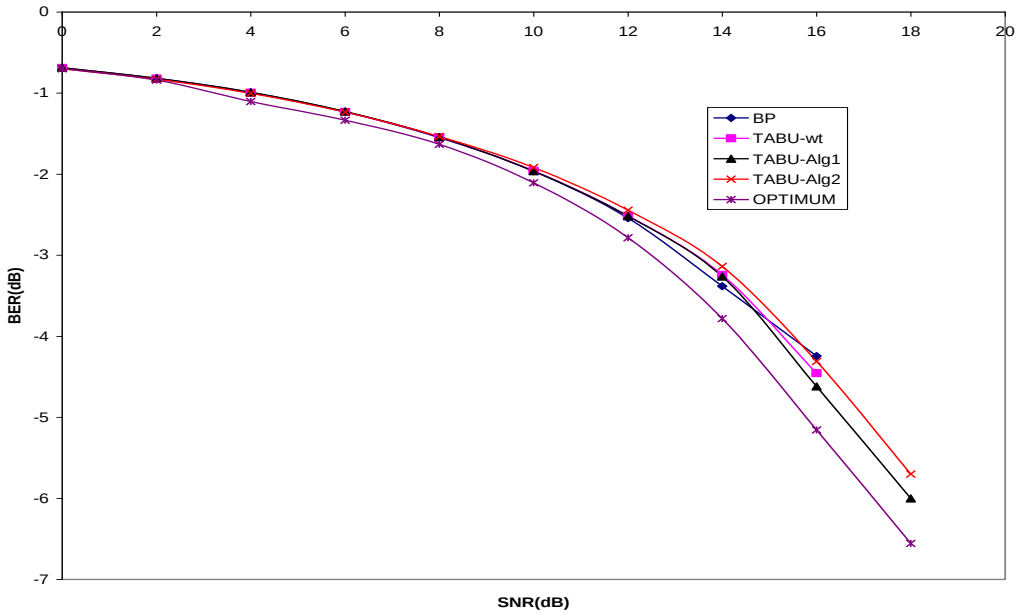

Fig. 6.11 SNR Vs BER comparison for Tabu based Algorithms and BP Algorithm for a 3-tap channel

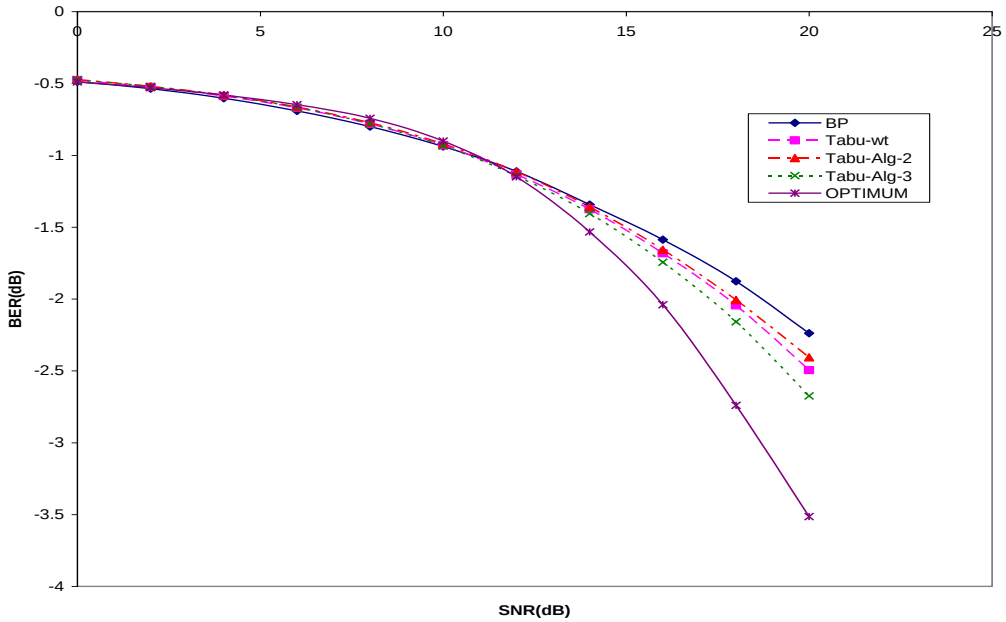

Fig. 6.12 SNR Vs BER comparison for Tabu based Algorithms and BP Algorithm for a 4-tap channel

The decision boundary for a two tap minimum phase channel $H_1(z)$ is plotted for the proposed algorithms for trained and untrained conditions. The figures from 6.13 through 6.18 shows for different iterations and for different searches fixing the delay and the equaliser order.

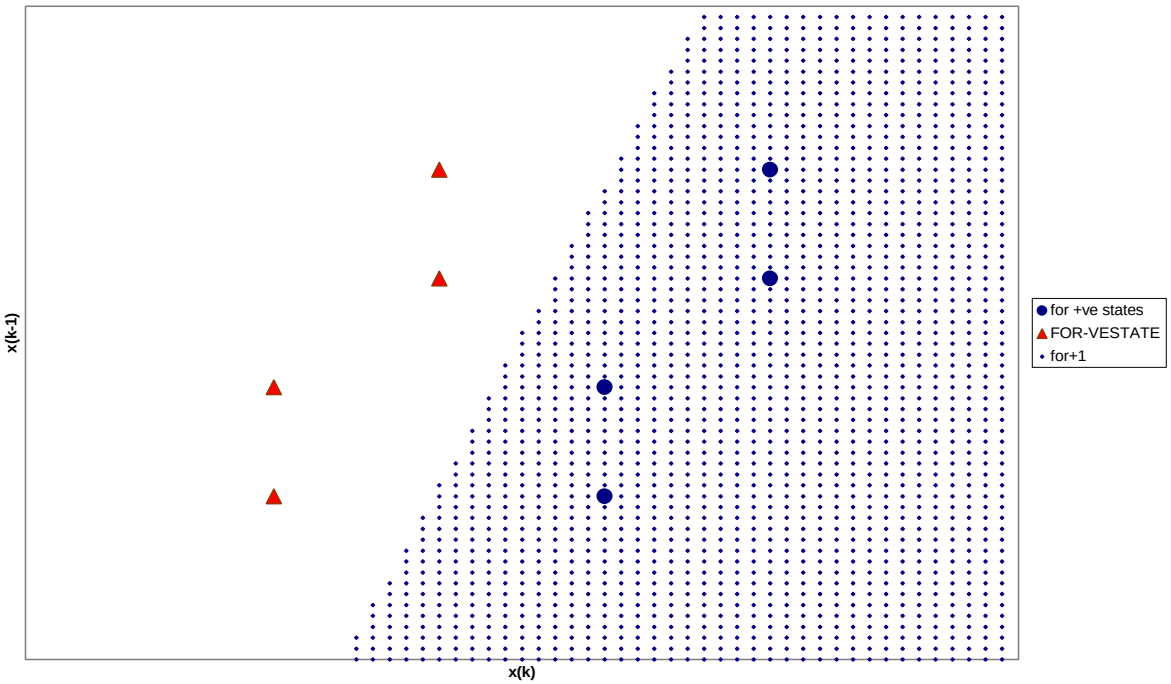

Fig 6.13 BP stopped training at 500 itr with m=2,d=0,struct=6:1

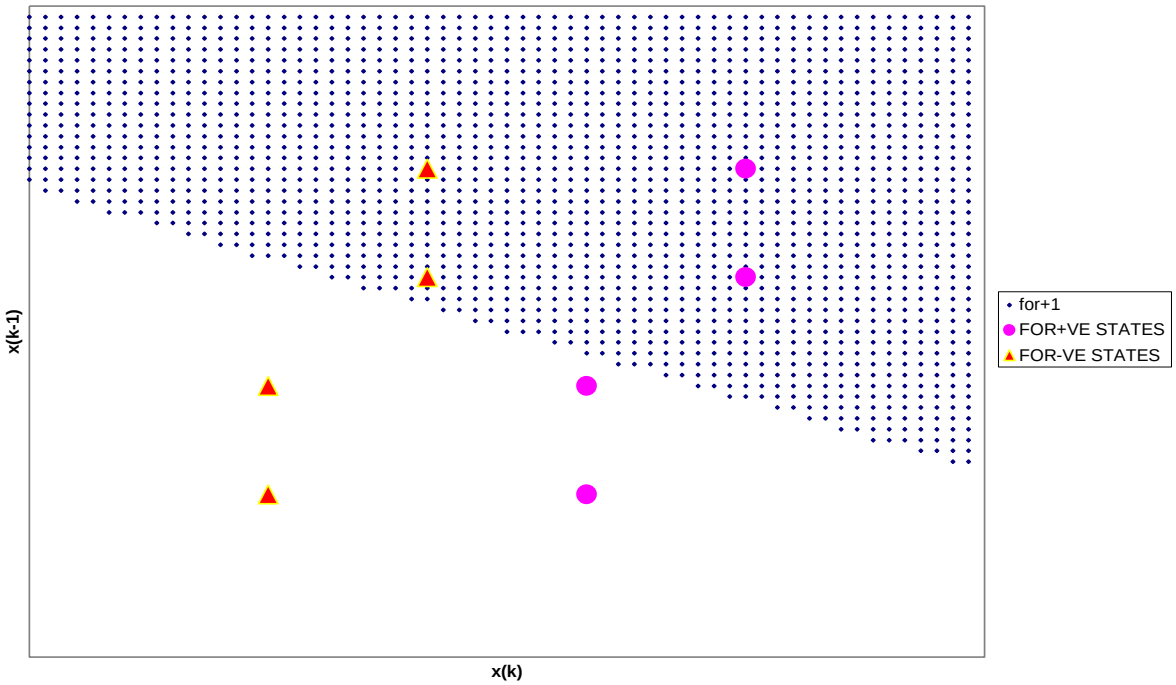

Fig 6.14 BP stopped training at 50 iterations structure 6:1,m=2,d=0

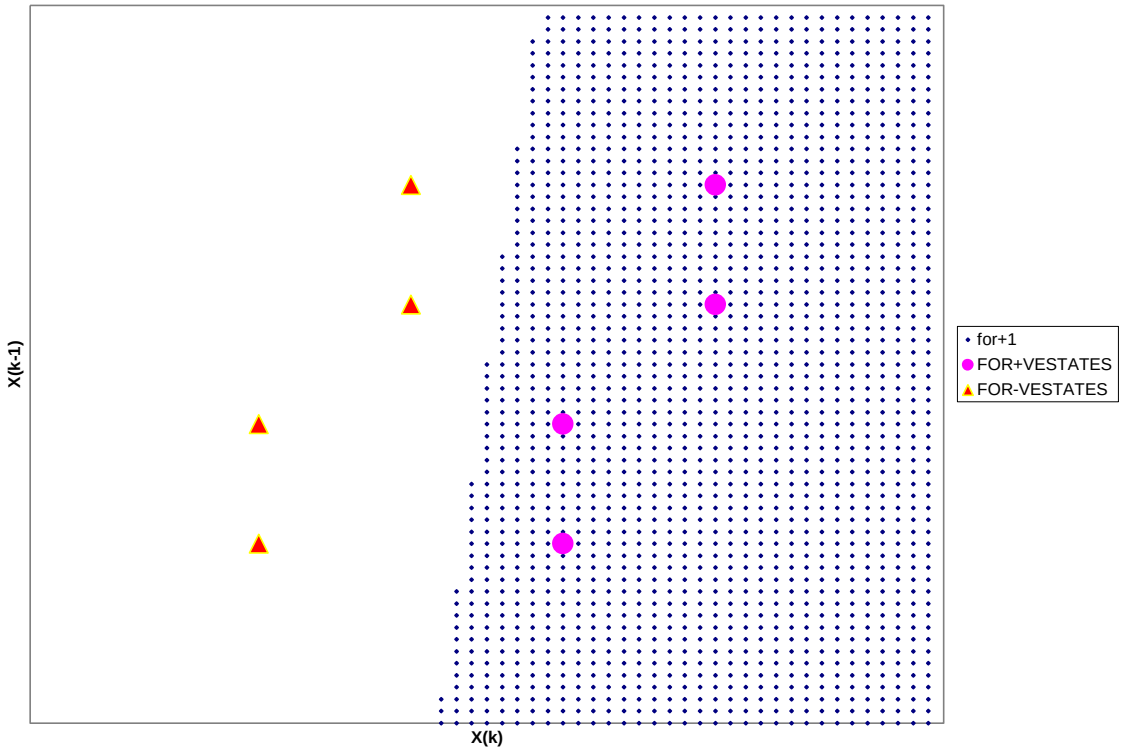

Fig 6.15TABU_wt stopped training for 100 iterations,10 searches,m=2,d=0 ,structure 1:1

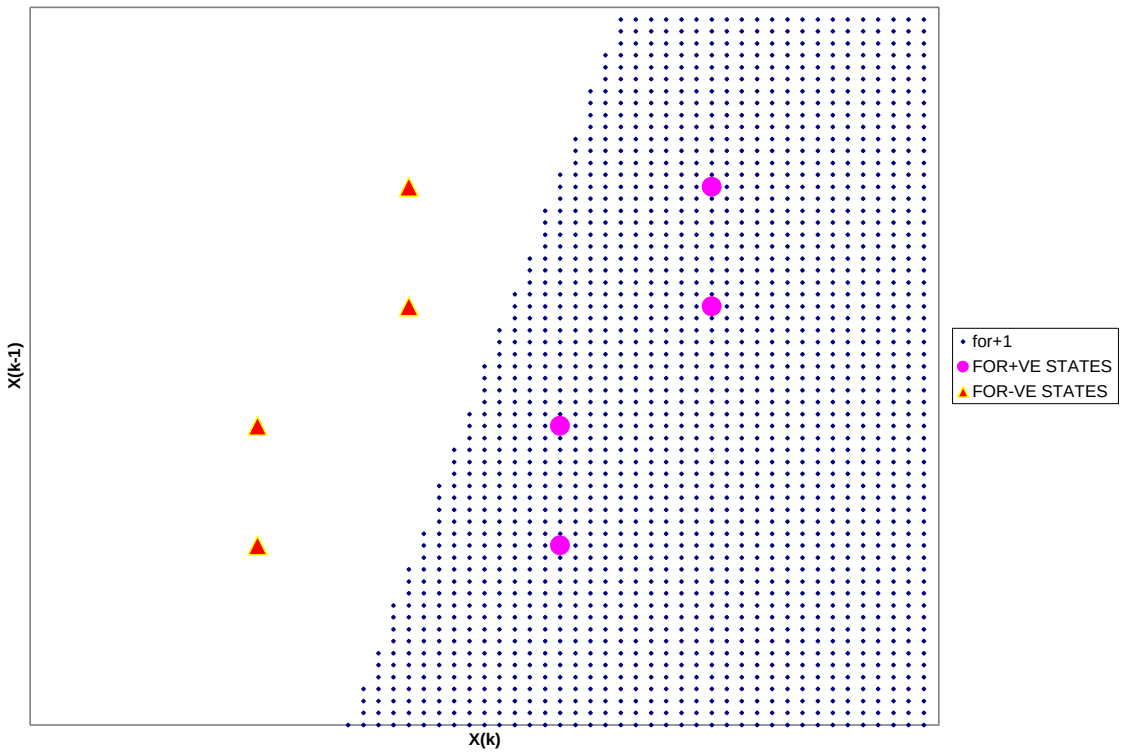

Fig 6.16TABU_wt stopped training for 50 iterations , 100 searches,m=2,d=0

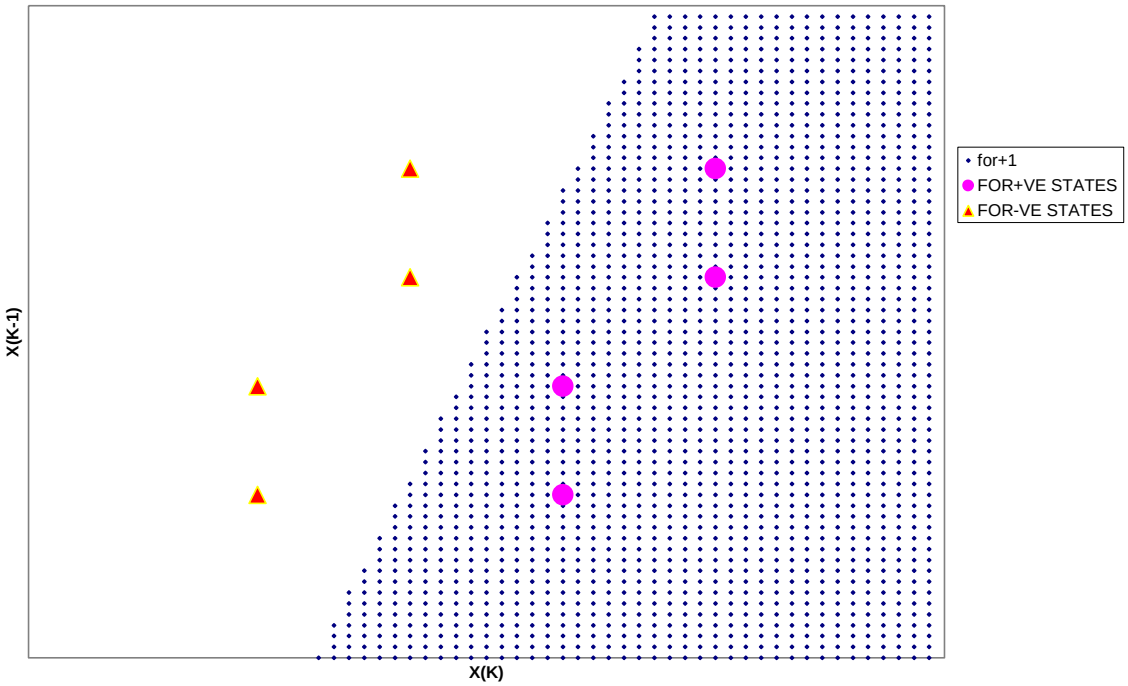

Fig 6.17TABU_Alg1,stopped training for 50 iterations100 searches,m=2,d=0,struct 1:1

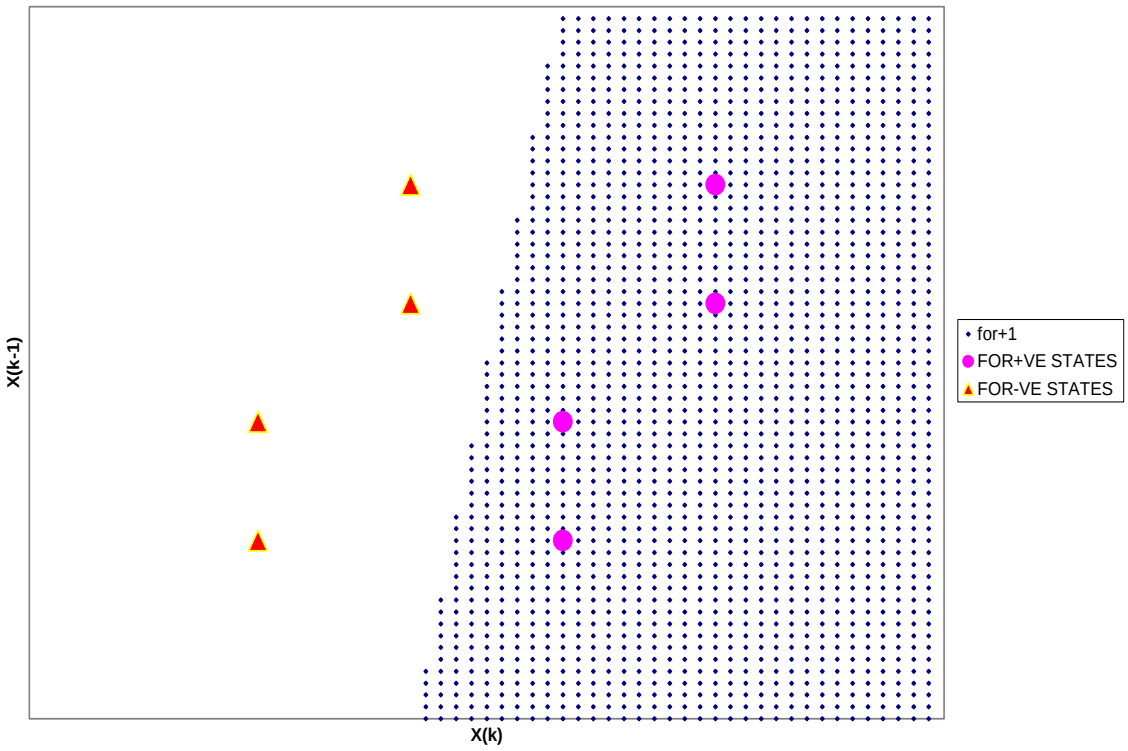

Fig 6.18 TABU_Alg2stopped training for 50 iterations , 25 searches m=2,d=0

An exclusive study of a 5 tap channel $H_9(z)$ is done for the proposed algorithm in terms of varying training samples, no of searches. The BER curves are shown in the fig 6.18 through 6.23. Attempt has been made to analyze that proposed scheme are possible for real time scenario. The time required for the searching process between *k* and *(k+1)* sample is more. Even exposing the structure to a sequence of 500,250 samples instead 1000,500 the BER performance is significant in the proposed algorithm when compared to the Back Propagation algorithm. The adaptation of the slope parameter will further increase the degree of freedom .

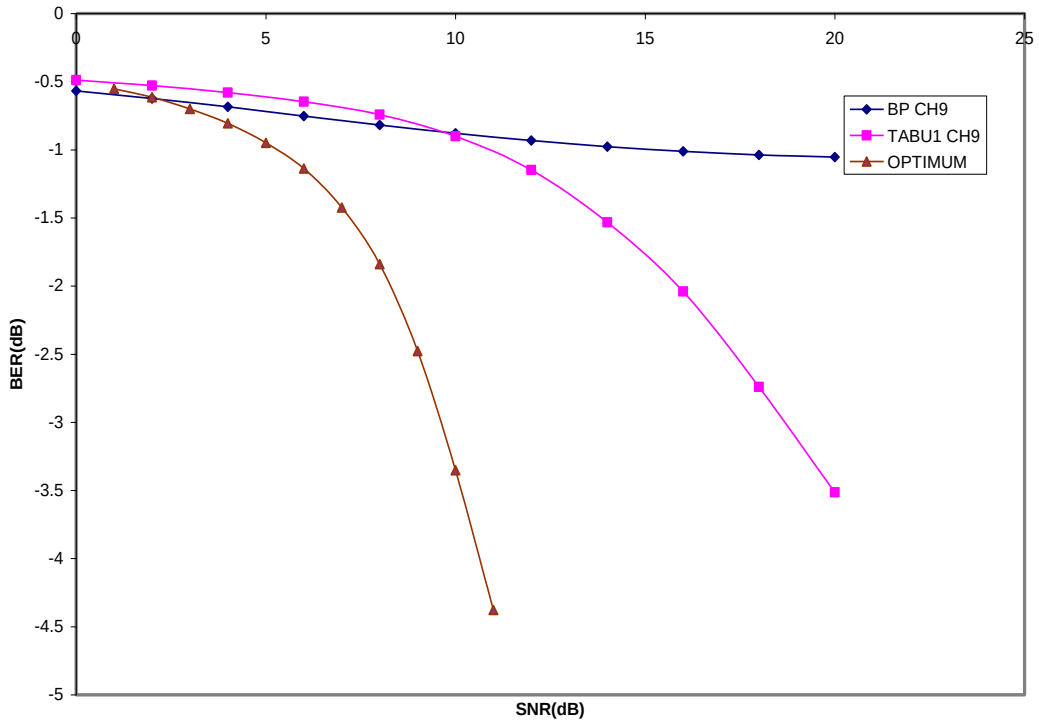

Fig 6.19 SNR Vs BER of BP(1:1) &TABU(1:1) trained for 1000 samples and 500 searches

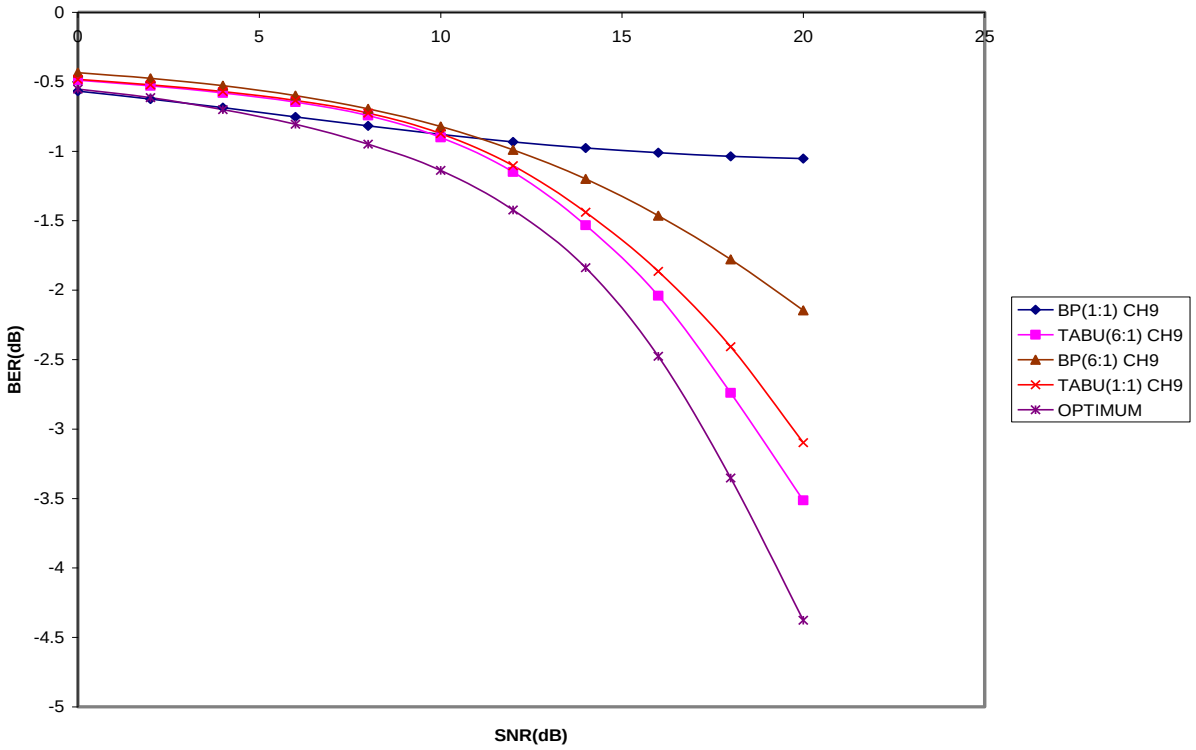

Fig6.20 SNR Vs BER BP&TABU for training stopped at 1000 iterations and 500searches

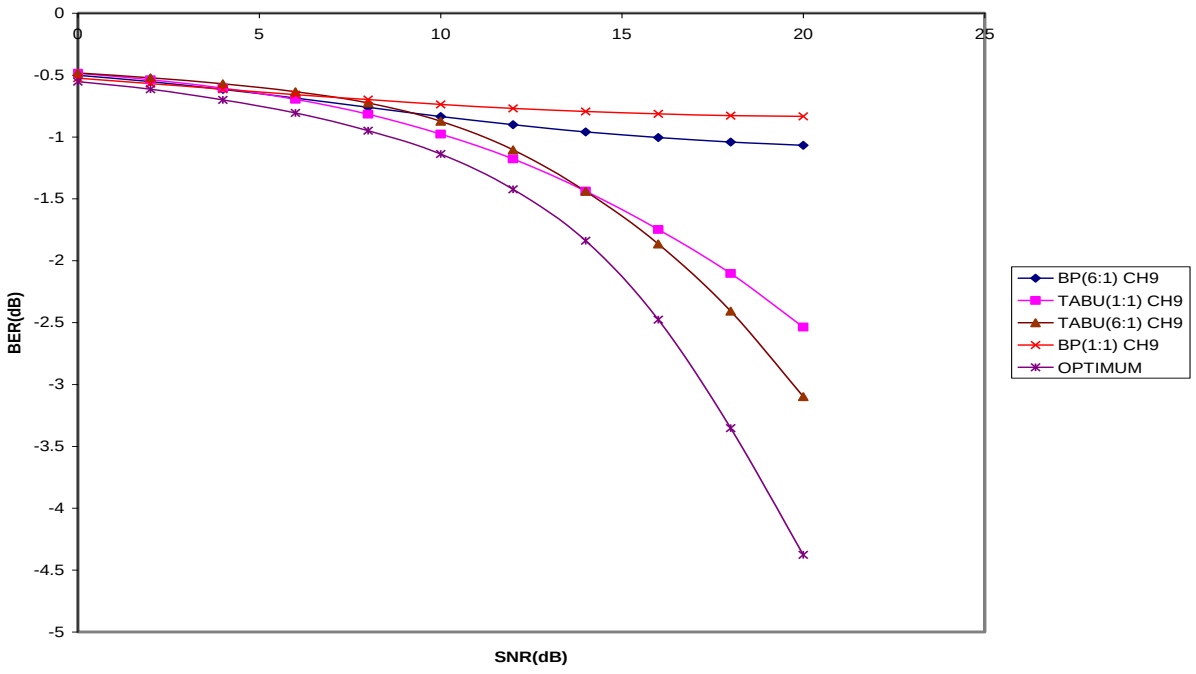

Fig 6.21SNR Vs BER BP&TABU for training stopped at 500 iterations and 500 searches

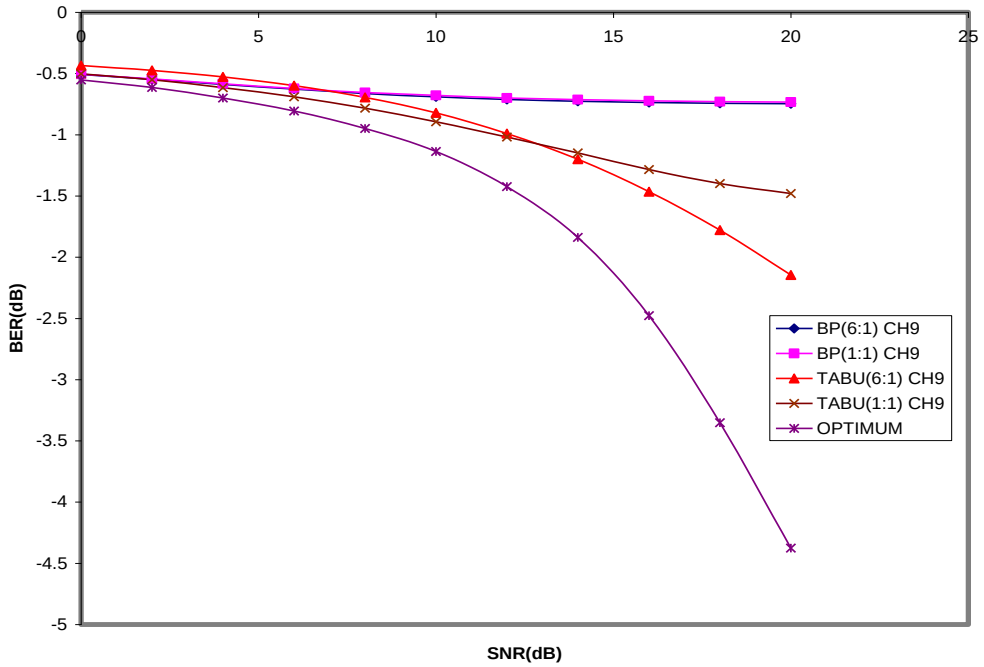

Fig 6.22SNR Vs BER BP&TABU for training stopped at 200 iterations and 500 searches

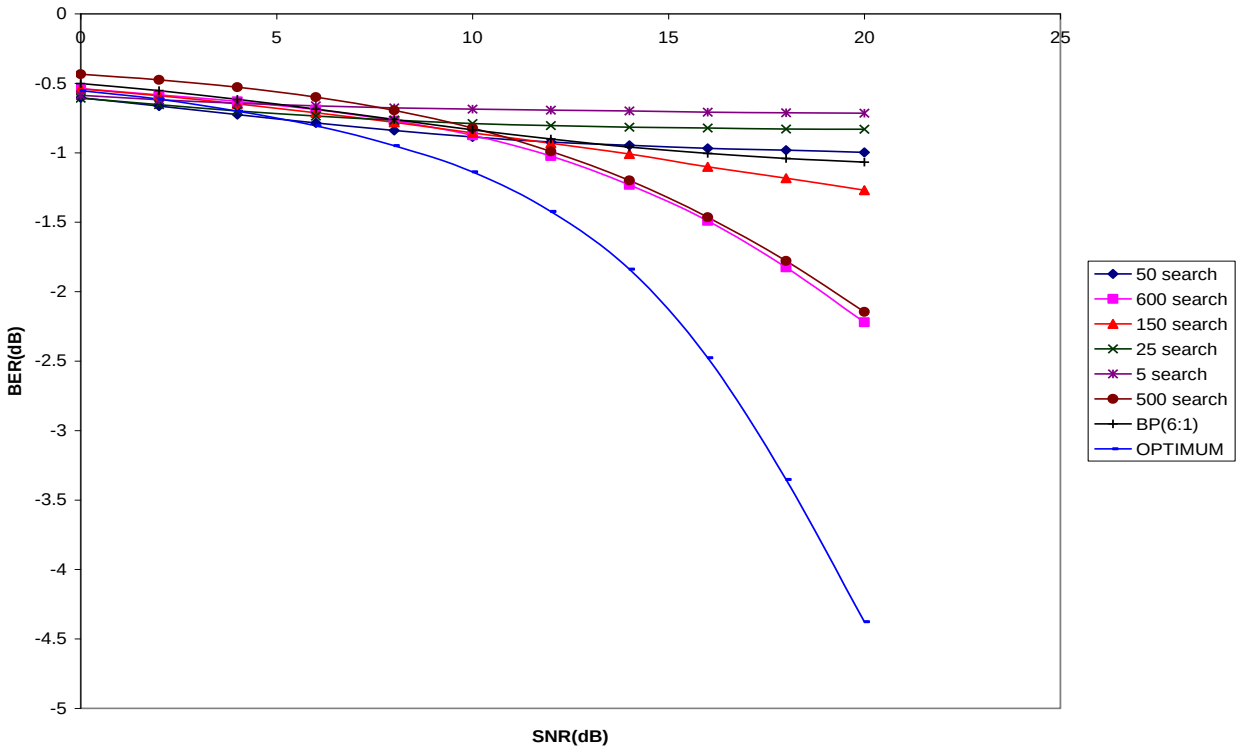

Fig 6.23 SNR Vs BER trained for 500 samples but only 250 samples are shown to TABU(6:1)

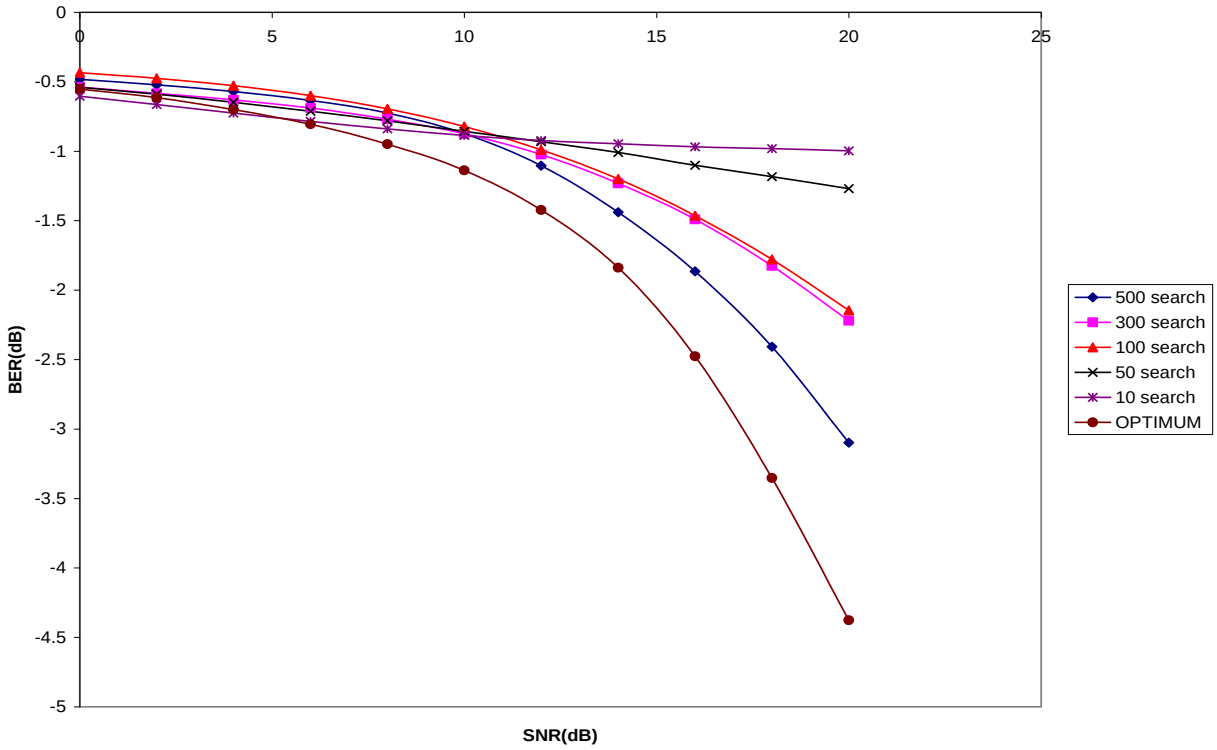

Fig 6.24 SNR Vs BER trained for 1000 samples but shown 500 samples for TABU(6:1)

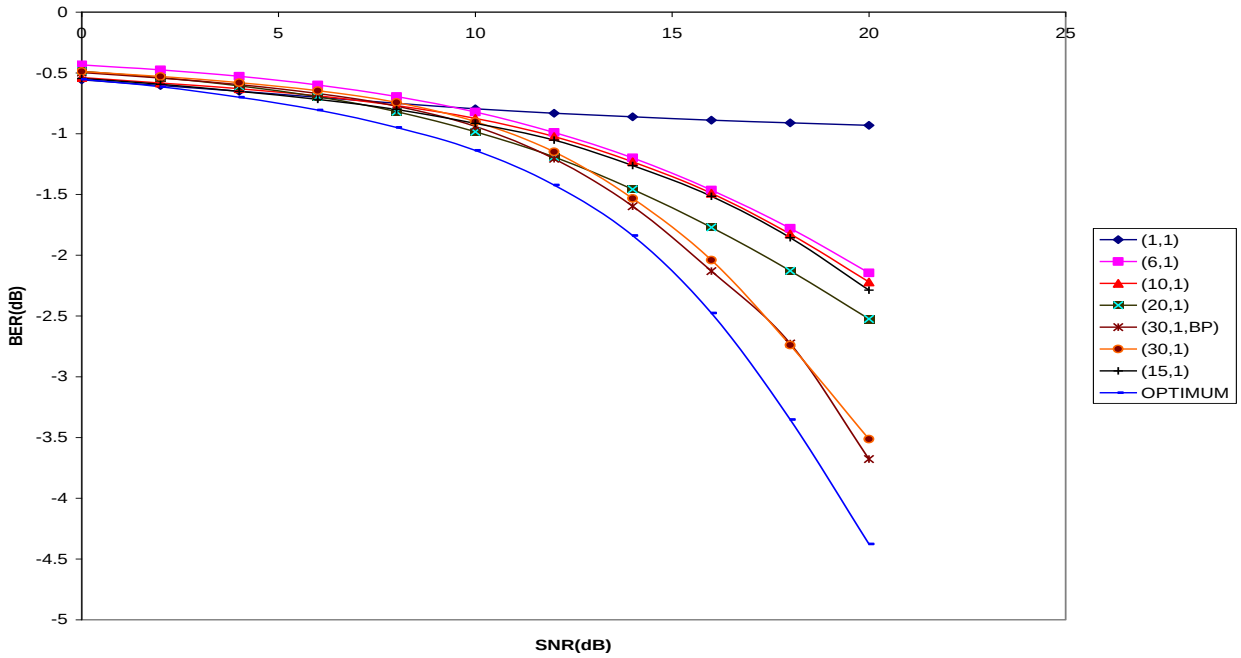

Fig 6.25 SNR Vs BER trained for 1000 samples but shown 500 samples for 100 fixed searches for TABU

Chapter 7

CONCLUSION

CONCLUSIONS

In this dissertation, the importance of artificial neural networks as nonlinear adaptive equalizers has been studied. The influence of different parameters on equalizers performance is observed. Various algorithms for training the neural network are presented. Among them the first one is the use of Recursive Least Square (RLS) algorithm for updating the weights of the neural network to improve the rate of convergence. It is noted that the use of this algorithm accelerates the training process and even provide better performance in terms of Bit Error Rate (BER), when trained for smaller number of iterations. But if they are trained with more number of iterations, there may not be appreciable increase in performance compared to that of training using BP algorithm, as both of them belong to the family of Gradient-based training algorithms, which are more likely to stop at some local minima.

Tabu Search (TS) has been used as a global search technique optimizing the synaptic weights and slopes of the activation functions of the neural network. Three algorithms are proposed, one for adapting the weights of the ANN using TS and other two for adapting the slopes. Slopes of the activation functions are adapted in two ways. In the first method first the weights are adapted using BP then the slopes are adapted using TS. In the second method, both weights and slopes are updated in each iteration.

Results show that the three proposed algorithms give superior performance compared to BP algorithm. But among the three, the last algorithm gives the best performance and is more suitable for real time applications. It is also shown that the proposed algorithms not only improves the equalizer performance but also reduces the structural complexity of the neural network. The sole objective of the proposed scheme is focused on the development of the reduced structure for a channel of higher order.

Every algorithm has its own pros and cons, TS is not an exception. It takes more time for searching the optimal solution and is computationally complicated, compared to BP algorithm. Also the TS is sensitive to the chosen search range. Further while the slopes are adapted using TS, the BP algorithm is chosen for adapting the weights. Any other superior optimization algorithms like Particle Swarm Optimization (PSO) may be used to adapt the weights. Moreover a simple TS has been used in this work, but over years, TS has undergone many modifications like the case of Adaptive Tabu Search [19]. The use of these algorithms may give a superior result.

Limitations of work

This section highlights some of the limitations of the proposed work reported in this thesis.

This thesis is generally concerned with the development of novel equaliser optimisation techniques in neural domain for communication systems. In all the experiments the equaliser feed forward order $\boldsymbol{m}$ has been restricted to channel order $\boldsymbol{n_a}$ .Even though that increasing $\boldsymbol{m}$ to a higher order can result in enhancement of BER performance, the selection of $\boldsymbol{m}$ is constrained to a specific value such that the objective of the proposed work is preserved. In this research work achieving the optimal Bayesian performance is not the major criterion, but development of optimization techniques with reduced structural complexity has been the main emphasis all along even though its results in some performance loss. However, minimal performance degradation is noticed and trade-off between structrural complexity and performance is maintained in the proposed algorithms.

The other limitation of work pertains to analysis of stationary channel models on time varying channels and multi channels has not been analysed in the present simulation study. Further, all the proposed training algorithms developed in this work is tested for applications using 2-PAM signaling only.

REFERENCES

[1] Haykin. S. Adaptive Filter Theory. Delhi: 4th Ed, Pearson Education, 2002

[2] Haykin. S. Digital Communication. Singapore: John Wiley & Sons Inc,1988.

[3] Proakis. J. G. Digital Communications. New York: McGraw-Hill, 1983.

[4] Qureshi. S. U. H, "Adaptive equalization," Proc. IEEE, vol. 73, no.9, pp.1349-1387, 1985.

[5] Widrow. B and Stearns. S. D. Adaptive Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1985.

[6] Chen. S, Mulgrew. B and Mclaughlin. S, "Adaptive Bayesian equalizer with decision feedback", IEEE Trans. Signal Processing, Vol. 41, No. 9, Sept 1993

[7] T. S. Rapport. Wireless Communications. Delhi: Pearson Education, 2000.

[8] Duda. R. O and Hart. P. E. Pattern Classification and Scene Analysis. New York: John Wiley & Sons Inc, 1973.

[9] Lippmann. R. P, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp.4-22, April 1987.

[10] Kim. M. C and Choi. C. H, "Square root learning in batch mode BP for classification problems," Proc. of ICNN, pp. 2769-2774, Nov. 1995.

[11] Zainuddin. Z, Mahat. N, Abu Hassan. Y, "Improving the Convergence of the Backpropagation Algorithm Using Local Adaptive Techniques", Proc. of ICCI, pp.173-176, Dec. 2004.

[12] Zhang. De-Xian, Liu. Can, Wang. Zi-Quiang, Liu. Nan-Bo, "A New Fast Learning Algorithm for Multilayer Feed Forward Neural Network", IEEE Proc. of ICMLC, (August 2006): pp- 2928-2934.

[13] Glover. F, "Tabu Search – Part I", ORSA Journal on Computing, vol.1, 1989, 190-206.

[14] Glover. F, "Tabu Search – Part II", ORSA Journal on Computing, vol.2, 1990, 4-32.

[15] Hertz A, Taillard E, de Werra D. "A Tutorial on Tabu Search". Proc. of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes), (1995), pp13-24.

[16] Haykin. S. Neural Networks: A Comprehensive Foundation. Delhi: 2nd Ed, Pearson Education, 2001

[17] Gibson. G. J, Siu. S, Cowan. C. F. N, "Multilayer Perceptron Structures Applied to Adaptive Equalizers for Data Communications", International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp.1183-1186, May 1989.

[18] Siu. S, Gibson. G. J and Cowan. C. F.N, "Decision feedback equalization using neural network structures and performance comparison with standard architecture", IEE Proceedings part I, vol. 137, no. 4, pp. 221-225, Aug 1990.

[19] Puangdownreong. D, Areerak. K. N, Srikaew. A, Sujitjorn. S, Totarong. P, "System Identification via Adaptive Tabu Search", Proc. of ICIT, Bangkok, (2002), pp: 915-920.

[20] Jian Ye, Junfei Qiao, Ming-ai Li, Xiaogang Ruan, "A Tabu based neural network learning algorithm", Neurocomputing, 70, 2007, 875-882.

[21] K.Abend and B.D.Fritchman,"Statistical Detection for Communication Channels with Inter sysmbol Interference",Proceedings of IEEE,Vol.58,pp.779-785,May 1970.

[22] Chen.S,Gibson. G. J, Cowan. C. F. N,"Adaptive Channel Equalisation using a Polynomial Perceptron Structure",IEE Proceedings-I on Communication,Speech and Vision, Vol.137,no.5,pp.257-264,October 1990

[23] Gibson. G. J, Siu. S, and Cowan. C. F.N, Grant P.M, "The application of non linear Architectures to Adaptive Channel Equalisation", IEEE International Conference on Communications,ICC'90 vol, no. 2, pp. 649-653, April 1990.

[24] Gibson. G. J, Siu. S, and Cowan. C. F.N, "The application of non linear Structures to the Reconstruction of Binary Signals", IEEE Transactions on Signal processing vol, no. 39,no 8, pp.1877-1884 , August 1991.

Contribution by the candidate

[P1] J. K. Satapathy "*A Faster RLS Based Learning Algorithm for Equalisation of Communication Channels"*, IEEE Conference on Computational Intelligence, Control and Computer Vision in Robotics &Automation, *CICCRA*, NIT ROURKELA 10-11 March 2008

[P2] J. K. Satapathy "*A Tabu based Neural Network Training Algorithm for Equalisation of Communication Channels"*, International conference on Artificial Intelligence,*ICAI'08(USA),*14-17 July 2008

[P3] J. K. Satapathy "*Tabu based Back propagation algorithm for performance Improvement in Communication Channels"*, accepted at IEEE -TENCON2008,18-21 November2008