NEW STRUCTURES AND ALGORITHMS FOR ADAPTIVE SYSTEM IDENTIFICATION AND CHANNEL EQUALIZATON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

NIHAR RANJAN PANDA

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007

NEW STRUCTURES AND ALGORITHMS FOR ADAPTIVE SYSTEM IDENTIFICATION AND CHANNEL EQUALIZATON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

NIHAR RANJAN PANDA

Under the Guidance of

Prof. G. Panda

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007

National Institute Of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **"New structures and Algorithms for Adaptive System Identification and Channel Equalization"** submitted by Sri **Nihar Ranjan panda** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & communication Engineering** with specialization in **"Telematics and Signal Processing"** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

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

Prof. G. Panda
Dept. of Electronics & Communication Engg.
National Institute of Technology
Rourkela-769008

Date:

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

I would like to extend my gratitude and my sincere thanks to my honorable, esteemed supervisor **Prof. G. Panda,** Head, Department of Electronics and Communication Engineering. He is not only a great lecturer with deep vision but also and most importantly a kind person. I sincerely thank for his exemplary guidance and encouragement. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

I want to thank all my teachers **Prof. G.S. Rath**, **Prof. K. K. Mahapatra, Prof. S.K. Patra** and **Prof. S.K. Meher** for providing a solid background for my studies and research thereafter. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

Last but not least I would like to thank my parents, who taught me the value of hard work by their own example. They rendered me enormous support during the whole tenure of my stay in NIT Rourkela.

Nihar Ranjan Panda

CONTENTS

Abstract

Over the last several years, adaptive digital filtering has been an active area of research and it has been considered for a variety of applications in signal processing and communications Systems.

An object formally referred to as a system is known through modeling and identification and can be understood by analysis. The main draw back of System Identification and Channel Equalization using ADF using LMS algorithm is that it takes a large number of iteration. BADF calculates a block or a finite set of filter outputs from a block of input values. From extensive computer simulation it is shown that BADF permits fast implementations while maintaining performance equivalent to that of the LMS. Furthermore efficient block algorithms such as the Fast Fourier Transform (FFT) can be used to advantage in terms of mathematical complexity and faster convergence when implementing block filters in frequency domain. But Adaptive filters perform poorly when it has to be operated in nonlinear dynamic systems.

In this thesis we propose an artificial neural network, which provides better and faster convergence performance when used for identifying nonlinear systems. The network employs chebyschev based nonlinear inputs updated with RLS algorithm. Through extensive computer simulation it is demonstrated that CFLANN updated with RLS is a better candidate compared to FLANN and CFANN updated with LMS. Along with these the proposed model requires less complex structure, less no of input samples and does accurate identification.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS USED

AF	Adaptive Filter
BAF	Block Adaptive Filter
LMS	Least Mean Square
RLS	Recursive Least Square
ANN	Artificial Neural Network
MLP	Multi layer perceptron
FLANN	Functional Link Artificial Neural Network
CFLANN	Chebyschev Functional Link Artificial Neural Network
CNN	Chebyschev Neural Network
DSP	Digital Signal Processing
FIR	Finite Impulse Response
MSE	Mean Square Error
BER	Bit Error Rate

Chapter 1

INTRODUCTION

1.1 Introduction

Over the last several years adaptive digital filtering is a major area of research and has been applied in many contexts such as non-linear system identification, forecasting of time-series, channel equalization, linear prediction, line enhancer and noise cancellation. Adaptive digital filter self adjusts its transfer function according to an optimizing algorithm to minimize the mean square between its output and that of an unknown system.

In recent years major advances are made in structures and optimizing algorithms to identify nonlinear systems with less mathematical complexity and with less no of input samples. Block adaptive digital filter calculates a block or finite set of filter outputs from a block of input values resulting in saving of lot of mathematical complexity. Block implementations allow efficient use of parallel processors, which results in speed gains while maintaining performance equivalent to that of the adaptive digital filter. Furthermore efficient block algorithms such as the Fast Fourier Transform (FFT) can be used to advantage when implementing block filters in frequency domain.

Recently, artificial neural networks (ANN) have emerged as a powerful learning technique to perform complex tasks in highly nonlinear dynamic environments. Some of the prime advantages of using ANN models are: their ability to learn based on optimization of an appropriate error function and their excellent performance for approximation of nonlinear function [1.1]-[1.2].

The functional link ANN (FLANN) proposed by Pao [1.3]-[1.4] has shown that this network can be used for function approximation and pattern classification with faster convergence rate and lesser computational complexity than a MLP network. The performance of the FLANN for the task of identification of nonlinear systems has been reported [1.5]. Using trigonometric functions as functional expansion, superior performance of the FLANN with respect to MLP network has been obtained. Here, we propose an alternate FLANN structure, which has been shown to provide effective identification of nonlinear dynamic systems. For functional expansion of the input pattern, we have chosen the Chebyschev polynomials [1.6] instead of trigonometric and the network is updated with recursive least mean square algorithm. Being a single layer neural network, its computational complexity is less intensive as compared to (MLP) and can be used for on-line learning. Pattern classification using CNN has been reported in [1.7]. It is shown that CNN based identification requires less computation as compared to MLP, less sample to converge.

The primary purpose of this chapter is to develop a computationally efficient and accurate algorithm for on-line system identification that is applicable to a variety of problems.

The Chebyschev neural network models to identify time series problem as well as discrete time plants. The identification scheme exhibits a learning-while-functioning feature instead of learning-then-functioning, so that the identification is on-line without any need of off-line learning phase. The training scheme is based on recursive least squares algorithm which guarantees convergence of the Chebyschev neural network weights. The proposed scheme also ensures good performance in the sense that the identification error is small and bounded. The convergence issue is shown through Lyapunov stability theory. The results are compared with certain existing identification algorithm.

1.2 Motivation

In the field of signal processing and communication Adaptive Filtering has a tremendous application such as non-linear system identification, forecasting of time-series, channel equalization, linear prediction, and noise cancellation. Adaptive digital filtering self adjusts its transfer function to get an optimal model for the unknown system based on some function of error based on the output of the adaptive filter and the unknown system. To get an optimal model of the unknown system it depends on the structure, adaptive algorithm and nature of input signal. System Identification estimates models of dynamic systems by observing their input output response when it is difficult obtain the mathematical model of the system.

DSP-based equalizer systems have become ubiquitous in many diverse applications including voice, data, and video communications via various transmission media. Typical applications range from acoustic echo cancellers for full-duplex speakerphones to video deghosting systems for terrestrial television broadcasts to signal conditioners for wire line modems and wireless telephony. The effect of an equalization system is to compensate for transmission-channel impairments such as frequency-dependent phase and amplitude distortion. Besides correcting for channel frequency-response anomalies, cancel the effects of Multipath signal and to reduce the intersymbol interference. So, designing of Equalizer to work for the above specifications is always a challenge and an active field of research.

On-line system identification or identification of complex systems is a major area of research from last several years. To give new solution to some long standing necessities of automatic control and to work with more and more complex system to satisfy stricter design criteria and to fulfill previous points with less and less a priori knowledge of the unknown system. In this context a great effort is being made within the system identification towards the development of

nonlinear models of real processes with less no of mathematical complexity, less no of input sample, faster matching and better convergence.

1.3 Thesis Layout

In Chapter2, the Adaptive Filter and System Identification problem are discussed in brief and an Adaptive Model for System Identification problem is given. Further more the nonlinear issues in the System Identification problems are discussed.

In Chaper3, the problem of Channel Equalization is discussed in detail. A basic model for Channel Equalization is given with LMS Equalizer and the concept of bit error rate was given.

In Chapter4, the System Identification and Channel equalization problem was solved with LMS and BLMS algorithm in time domain and it was shown that BLMS algorithm works faster than the conventional LMS algorithm.

In Chapter5, the BLMS algorithm in frequency domain for both overlap add and save sectioning was discussed. It was shown with computer simulation that BLMS algorithm converges much faster and gives better bit error rate with less number of computational complexity.

In Chapter6, the concept of Neuron, MLP, FLANN and CFLANN was discussed and the RLS algorithm has been derived. Comparison of the above structures for nonlinear system identification and channel equalization problem was given through extensive computer simulation and it was seen that in almost all cases CFLANN is always a better candidate in terms of faster convergence less mathematical complexity. Further more CFLANN has a simple structure gives a better Bit Error Rate for all non-linear channels.

In Chapter6, the proposed structure is used for on-line identification problem. Some standard problems such as box and Jenkins identification problem, SISO Plant, MIMO Plant identification was solved through the proposed structure and its value is compared with other values with other structures.

Chapter 2

ADAPTIVE SYSTEM IDENTIFICATION

2.1. Introduction

System identification [2.1, 2.2] is the experimental approach to process modeling. System identification includes the following steps

(a) Experiment design: Its purpose is to obtain good experimental data and it includes the choice of the measured variables and of the character of the input signals.

(b) Selection of model structure: A suitable model structure is chosen using prior knowledge and trial and error.

(c) Choice of the criterion to fit: A suitable cost function is chosen, which reflects how well the model fits the experimental data.

(d) Parameter estimation: An optimization problem is solved to obtain the numerical values of the model parameters.

(e) Model validation: The model is tested in order to reveal any inadequacies.

The key problem in system identification is to find a suitable model structure, within which a good model is to be found. Fitting a model within a given structure (parameter estimation) is in most cases a common problem. A basic rule in estimation is not to estimate what you already know. In other words, one should utilize prior knowledge and physical insight about the system when selecting the model structure. It is customary to distinguish between three levels of prior knowledge, which have been color-coded as follows.

White Box models: This is the case when a model is perfectly known; it has been possible to construct it entirely from prior knowledge and physical insight.

Grey Box models: This is the case when some physical insight is available, but several parameters remain to be determined from observed data. It is useful to consider two sub cases:

Physical Modeling: A model structure can be built on physical grounds, which has a certain number of parameters to be estimated from data.

Semi-physical modeling: Physical insight is used to suggest certain nonlinear combinations of measured data signal. These new signals are then subjected to model structures of black box character.

Black Box models: No physical insight is available or used, but the chosen model structure belongs to families that are known to have good flexibility and have been "successful in the past".

A nonlinear black box [2.1] structure for a dynamical system is a model structure that is prepared to describe virtually any nonlinear dynamics. There has been considerable recent interest

in this area with structures based on simple LMS [2.3], RLS [2.4], Multilayer Perceptron, FLANN, radial basis networks [2.9] based methods. Here we have discussed the common framework for these approaches.

Basic techniques for estimating the parameters in the structures are criterion minimization, as well as two step procedures, where first the relevant basis functions are determined, using data, and then a linear least squares step to determine the coordinates of the function approximation. A particular problem is to deal with the large number of potentially necessary parameters. This is handled by making the number of "used" parameters considerably less than the number of "offered" parameters, by regularization, shrinking, pruning or regressor selection.

In Section 2.2, we present the general basic system identification problem, solution via adaptive approach and introduce the mathematical notation for representing the form and operation of the adaptive filter. We then discuss several different linear models that have been proven to be useful in practical applications for FIR channels in Section 2.3. We provide an overview of the many and varied applications in which adaptive filters have been successfully used. We give a simple derivation of the least-mean-square (LMS) algorithm, which is perhaps the most popular method for adjusting the coefficients of an adaptive filter, and we discuss some of this algorithm's properties and shortcomings in Section 2.4. We discuss recursive LMS algorithm & its limitation and finally, we discuss new algorithms and techniques, which can be applied in place of conventional methods for nonlinear case.

2.2 Basics of System Identification

System Identification is the art and methodology of building mathematical models of dynamical systems based on input-output data. We denote the output of the dynamical system at time *t* by *y(t)* and the input by *u(t)*. The data are assumed to be collected in discrete time. At time *t* we thus have available the data set & the most basic relationship between the input and output in form of linear differential equation

$$y(t) + a_1 y(t-1) + \ldots\ldots + a_n y(t-n) = b_1 u(t-1) + \ldots\ldots + b_m u(t-m) \qquad (2.1)$$

The system represented above is in discrete time, primarily since sampling always collects observed data. A pragmatic way to see Eq. (2.1) is to view it as a way of determining the next output value given the previous observations:

$$y(t) = -a_1 y(t-1) - \ldots\ldots - a_n y(t-n) + b_1 u(t-1) + \ldots\ldots + b_m u(t-m) \qquad (2.2)$$

This can be written in a more compact form as follows:

$$\theta = [a_1, ...a_n, b_1, ...b_m]^T \tag{2.3}$$

$$\varphi(t) = [-y(t-1).....-y(t-n)\, u(t-1).....u(t-m)]^T \tag{2.4}$$

With above two equations, we can write

$$y(t) = \varphi^T(t)\theta \tag{2.5}$$

To emphasize that the calculation of *y (t)* from past data indeed depends on the parameter θ, we shall rather call this calculated value $\hat{y}(t/\theta)$ and write

$$\hat{y}(t/\theta) = \varphi^T(t)\theta \tag{2.6}$$

2.3 Adaptive Filter

An adaptive filter [2.3, 2.4] is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. Adaptive filters are often realized either as a set of program instructions running on an arithmetical processing device such as a microprocessor or DSP chip, or as a set of logic operations implemented in a field-programmable gate array (FPGA) or in a semi-custom or custom VLSI integrated circuit. An adaptive filter is defined by four aspects:

1. The signals being processed by the filter .
2. The structure that defines how the output signal of the filter is computed from its input signal.
3. The parameters within this structure that can be iteratively changed to alter the filter's input-output relationship.
4. The adaptive algorithm that describes how the parameters are adjusted from one time instant to the next.

By choosing a particular adaptive filter structure, one has to specify the number and type of parameters that has to be adjusted. An adaptive algorithm is use to update the parameter values of the system to minimize the mean square between its output and that of an unknown system.

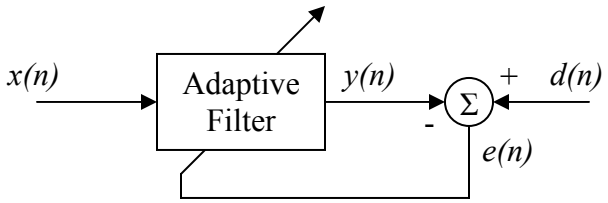


Fig 2.1: The general adaptive filtering problem.

Fig. 2.1 shows a basic block diagram in which a sample from a digital input signal $x(n)$ is fed into a device, called an adaptive filter, that computes a corresponding output signal sample $y(n)$ at time n. For the moment, the structure of the adaptive filter is not important; except for the fact that it contains adjustable parameters whose values affect how $y(n)$ is computed. The output signal is compared to a second signal $d(n)$, called the desired response signal, by subtracting the two samples at time n. This difference signal, given by

$$e(n). = d(n) - y(n) \tag{2.7}$$

is known as the error signal. The error signal is fed into a procedure which alters or adapts the parameters of the filter from time n to time $(n+1).$ in a well-defined manner. This process of adaptation is represented by the oblique arrow that pierces the adaptive filter block in the figure. As the time index n is incremented, it is desired that the output of the adaptive filter becomes a better matched to the desired response of the signal through this adaptation process, such that the magnitude of $e(n).$ decreases over time. In the adaptive filtering task, adaptation refers to the method by which the parameters of the system are changed from time index n to time index $(n+1).$ The number and types of parameters within this system depend on the computational structure chosen for the system. We now discuss different filter structures that have been proven useful for adaptive filtering tasks.

FIR Filter

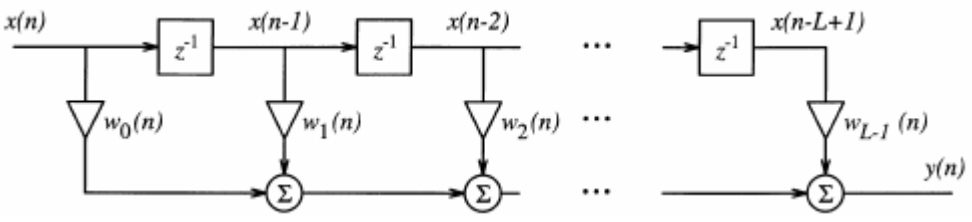


Fig. 2.2 Structure of an FIR filter

In general, any system with a finite number of parameters that affect how $y(n)$ is computed from $x(n)$ could be used for the adaptive filter in Fig. 2.1. Define the parameter or coefficient vector $W(n)$ as

$$W(n) = [w_0(n)\ w_1(n) \ldots w_{L-1}(n)]^T \tag{2.8}$$

Fig. 2.2 shows the structure of a direct-form FIR [2.5,2.6] filter, also known as a tapped-delay-line or transversal filter, where z^{-1} denotes the unit delay element and each $w_i(n)$ is a multiplicative gain within the system. In this case, the parameters in $W(n)$ correspond to the impulse response values of the filter at time n.

We can write the output signal *y(n)* as

$$y(n) = \sum_{i=0}^{L-1} w_i(n)x(n-i) \tag{2.9}$$

$$= W^T(n)X(n) \tag{2.10}$$

Where $x(n) = [x(n)\ x(n)\ x(n-1) \ldots x(n-L+1)]^T$ denotes the input signal vector.

The general form of an adaptive FIR filtering algorithm is

$$W(n+1) = W(n) + \mu(n)G(n)X(n) \tag{2.11}$$

Where G (-) is a particular vector-valued nonlinear function, μ *(n)* is a step size parameter, *e(n)* and *X(n)* are the error signal and input signal vector, respectively. Much research effort has been spent characterizing the role that $\mu(n)$ plays in the performance of adaptive filters in terms of the statistical or frequency characteristics of the input and desired response signals.

We now consider one particular cost function that yields a popular adaptive algorithm. Define the mean-squared error (MSE) [2.3] cost function as

$$J_{MSE}(n) = \frac{1}{2}\int_{-\infty}^{\infty} e^2(n)p_n(e(n))de(n)$$

$$= \frac{1}{2}E\{e^2(n)\} \tag{2.12}$$

Where $P_n(e)$ represents the probability density function of the error at time n and $E\{.\}$ is shorthand for the expectation integral of error square. $J_{MSE}(N)$ has a well-defined minimum with respect to the parameters in $W(n)$,The coefficient values obtained at this minimum are the ones that minimize the power in the error signal $e(n)$,indicating that $y(n)$has approached $d(n)$,J_{MSE} is a smooth function of each of the parameters in $W(n)$.such that it is differentiable with respect to each of the parameters in $W(n)$.

The third point is important in that it enables us to determine both the optimum coefficient values from the given knowledge of the of $e(n)$ and $x(n)$ as well as a simple iterative procedure for adjusting the parameters of an FIR filter. For the FIR filter structure, the coefficient values in $W(n)$ that minimize $J_{MSE}(n)$ are well defined if the statistics of the input and desired response signals are known. The formulation of this problem for continuous-time signals and the resulting solution was first derived by Wiener [2.4]. Hence, this optimum coefficient vector $W_{MSE}(N)$ is often called the Wiener solution [2.3] to the adaptive filtering problem. To determine $W_{MSE}(N)$ we note that the function $J_{MSE}(N)$ in Eq.(2.12) is quadratic in the parameters $[w_i(n)]$, and the function is also differentiable. Thus, we can use a result from optimization theory that states that the derivatives of a smooth cost function with respect to each of the parameters is zero at a minimizing point on the cost function error surface. Thus, $W_{MSE}(n)$ can be found from the solution to the system of equations

$$\frac{\partial J_{MSE}(n)}{\partial w_i(n)} = 0 , \quad 0 \le i \le L-1 \tag{2.13}$$

Taking derivatives of $J_{MSE}(N)$ in Eq.(2.12) and noting that $e(n)$ *and* $y(n)$ are given by Eq.(2.7) and (2.9), respectively, we obtain

$$\frac{\partial J_{MSE}(n)}{\partial w_i(n)} = E\{e(n)\frac{\partial e(n)}{\partial w_i(n)}\} \tag{2.14}$$

$$= -E\{e(n)\frac{\partial y(n)}{\partial w_i(n)}\}$$

$$= -E\{e(n)x(n-i)\}$$

$$= -\left(E\{d(n)x(n-i)\} - \sum_{j=0}^{L-1} E\{x(n-i)x(n-j)\}w_j(n)\right)$$

(2.15)

By defining the matrix $R_{xx}(n)$ and vector $P_{dx}(n)$ as

$R_{XX} = E\{X(n)X^T(n)\}$ And

$$P_{dx}(n) = E\{d(n).X(n)\} \tag{2.16}$$

respectively, we can combine Eq.(2.15) and (2.16) to obtain the system of equations in vector form as

$$R_{XX}(n)W_{MSE}(n) - P_{dx}(n) = 0 \tag{2.17}$$

Thus, so long as the matrix $R_{xx}(n)$ is invertible, the optimum Wiener solution vector for this problem is

$$W_{MSE}(n) = R_{XX}^{-1}(n)P_{dx}(n) \tag{2.18}$$

The method of steepest descent [2.3] is a celebrated optimization procedure for minimizing the value of a cost function $J(n)$ with respect to a set of adjustable parameters $W(n)$. This procedure adjusts each parameter of the system according to

$$w_i(n+1) = w_i(n) - \mu(n)\frac{\partial J(n)}{\partial w_i(n)} \tag{2.19}$$

In other words, the i^{th} parameter of the system is altered according to the derivative of the cost function with respect to the i^{th} parameter. Collecting these equations in vector form, we have

$$W(n+1) = W(n) - \mu(n)\frac{\partial J(n)}{\partial W(n)} \tag{2.20}$$

Where $dJ(n)/dW(n)$ is a vector of derivatives $dJ(n)/dw_i(n)$.

For an FIR adaptive filter that minimizes the MSE cost function, we can use the result in Eq. (2.15) to explicitly give the form of the steepest descent procedure in this problem. Substituting these results into Eq.(2.20) yields the update equation for $W(n)$ as

$$W(n+1) = W(n) + \mu(n)(P_{dx}(n) - R_{XX}(n)W(n) \tag{2.21}$$

However, this steepest descent procedure depends on the statistical quantities $E\{d(n)x(n-i)$ *and* $E\{x(n-i)x(n-j)\}$ contained in $P_{dx}(n)$ and $R_{xx}(n)$,respectively. In practice, we only have measurements of both $d(n)$ and $x(n)$ to be used within the adaptation procedure. While suitable estimates of the statistical quantities needed for Eq. (2.21) could be determined from the signals $x(n)$ and $d(n)$,we instead develop an approximate version of the method of steepest descent that depends on the signal values themselves. This procedure is known as the LMS algorithm.

We can propose the simplified cost function $J_{MSE}(N)$ given by

$$J_{LMS}(n) = \frac{1}{2}e^2(n) \tag{2.22}$$

This cost function can be thought of as an instantaneous estimate of the MSE cost function, as $J_{MSE}(n) = E\{J_{LMS}(n)\}$. Taking derivatives of $J_{MSE}(N)$ with respect to the elements of $W(n)$ and substituting the result into Eq. (2.20), we obtain the LMS adaptive algorithm given by

$$W(n+1) = W(n) + \mu(n)e(n)X(n) \tag{2.23}$$

The number and type of operations needed for the LMS algorithm is nearly the same as that of the FIR filter structure with fixed coefficient values, which is one of the reasons for the algorithm's popularity.

2.4 Adaptive Model for System Identification

Consider Fig. 2.3, which shows the general problem of system identification. In this diagram, the system enclosed by dashed lines is a "black box,"[2.1] meaning that the quantities inside are not observable from the outside. Inside this box, there is an unknown system which represents a general input-output relationship. In many practical cases, the plant to be modeled is noisy, that is, has internal random disturbing forces. In our problem it is represented by the signal $\eta(n)$, called the observation noise signal because it corrupts the observations of the signal at the output of the unknown system. Internal plant noise appears at the plant output and is commonly represented there as an adaptive noise. This noise is generally uncorrelated with the plant input. If this is the case and if the adaptive model weights are adjusted to minimize mean-square error, it can be shown that the least square solutions will be unaffected by the presence of plant noise. This is difficult to say that the convergence of the adaptive process will be unaffected by plant noise, only that the expected weight vector of the adaptive model after convergence will be unaffected.

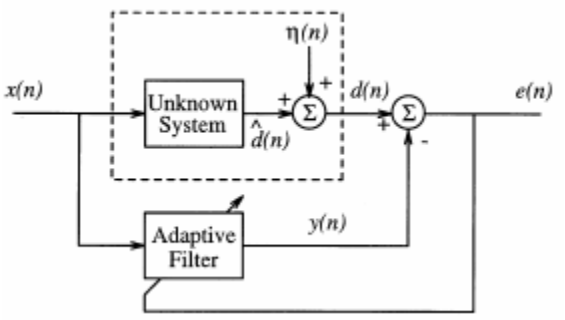


Fig. 2.3 Model for System Identification

Let $d(n)$ represent the output of the unknown system with $x(n)$ as its input. Then, the desired response signal in this model is $d(n) = \hat{d}(n) + \eta(n)$.

Here, the task of the adaptive filter is to accurately represent the signal $d(n)$ at its output. If $y(n) = d(n)$, then the adaptive filter has accurately modeled or identified the portion of the unknown system that is driven by $x(n)$.

For our problem we have assumed that the input signal $x(n)$ and noise signal $\eta(n)$ are mutually uncorrelated white random sequences with zero mean. And hence

$$E\left[x_n^2\right] = 1/12 \quad E\left[\eta_n^2\right] = 1/12 \tag{2.34}$$

The white noise is Gaussian in nature having probability density function as follows:

$$p(\delta) = \frac{1}{\sqrt{2\pi{\sigma_\delta}^2}} \exp\left(\frac{-(\delta - m)^2}{2{\sigma_\delta}^2}\right) \tag{2.35}$$

As the adaptation process reaches Wiener solution, the power of error signal will be exactly equal to the extra noise added. It is not exactly required that the impulse response of both unknown plant and adaptive model shown in Fig. 2.3 should match, but the desired output and estimated output should match.

2.5 Nonlinear Issues

A system could be either linear or nonlinear, depending on the relation between the input and the output of the system. Accordingly, various linear and nonlinear filtering techniques have been developed to achieve certain predefined design goals. Systems are often modeled as linear since it makes the design and analysis tasks mathematically tractable. Accordingly, the theory of linear filtering has been extensively studied in above section and has reached a stage of sufficient maturity. If the system is either inherently linear or the degree of nonlinearity is negligible, the behavior of the system is as expected. Otherwise, there could be significant deviation from expected behavior and the performance of the system could degrade severely. In such cases, it is essential to apply nonlinear methods that properly characterize the system behavior. Moreover, when either the system is time-varying or certain parameters of the system are unknown, it is essential to adapt the filter to track the dynamics of the system or learn the unknown parameters. Hence in the next chapter we have focused on developing novel nonlinear adaptive filtering techniques and their applications to some practical problems of interest.

Nonlinear systems/filters do not satisfy the property of superposition. Conventional linear filtering does not suffice in presence of nonlinearities. A few practical cases wherein nonlinearities are commonly encountered in signal processing and communications applications are listed here.

• High power amplifiers (HPAs): In wireless communications, for higher power efficiency the HPAs are driven close to saturation. The HPAs are found to introduce nonlinear amplitude and phase distortion when operated near saturation. This causes degraded bit-error rate (BER) performance and

also introduces adjacent channel interference (ACI) to systems operating in the neighboring frequency bands.

• Magnetic Recording Channels: In high-density magnetic recording channels, nonlinearity is introduced in the form of nonlinear bit-shifts and partial erasure. They are modeled as nonlinear inter-symbol interference (ISI) channels.

• Optical channels: Fiber optic receivers suffer from various sources of nonlinear distortion including the photo detector, which converts incident light into photocurrent, intensity dependence of the index of refraction of the fiber and amplified spontaneous emission, which has non-Gaussian distribution.

• Speech and Image processing: Nonlinear modeling is frequently used to analyze the data in applications involving speech and image processing like image segmentation, image restoration, edge enhancement, speech coding, speech enhancement, etc.

• Loudspeakers: Loudspeakers generate nonlinear distortion that degrades the quality of the audio. The sources of nonlinearity are nonlinearity in the suspension system and inhomogeneity in the flux density.

• Echo Cancellation: Acoustic echo cancellers are predominantly used in speakerphones and video conferencing systems to minimize the undesirable echo. But the echo path is usually highly nonlinear and hence nonlinear echo cancellation methods are being used.

• Biomedical Engineering: There is a lot of scope for modeling and analysis of nonlinear systems/signals in biomedical engineering like study of neural response, human visual system, nonlinear properties of tissue, speech pathology assessment to name a few.

Thus, as seen above, nonlinear distortions manifest in many practical systems and they need to be compensated for satisfactory performance. Hence we have considered some standard

nonlinearity for our experiment. Instead of using linear channel we have used nonlinear channel. Some standard nonlinear equations are given.

$f(u) = u^3 + 0.3u^2 - 0.4u \rightarrow$

$f(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) - 0.1\sin(5\pi u)$

$f(u) = \tanh(u)$

$f(u) = u + 0.2u^2 - 0.1u^3 + 0.5\cos(\pi u)$

2.6 Summary

Studies on linear system identification have been carried out for more than three decades [2.4]. However, identification of nonlinear systems is a promising research area. Conventionally the identification of linear system is performed by using Least Mean Square (LMS) algorithm. However most of the dynamical systems exhibit nonlinearity. It has been studied that the LMS based technique [2.3, 2.4] does not perform satisfactorily to identify nonlinear systems. To improve the identification performance of nonlinear systems various techniques such as Block Adaptive filter(BAF) both in time domain and frequency domain, Multilayer Perceptron (MLP), Functional Link Neural Network (FLANN), Radial Basis Function has been studied.

Chapter 3

ADAPTIVE CHANNEL EQUALIZATION

3.1 Introduction

DSP-based equalizer systems have become ubiquitous in many diverse applications including voice, data, and video communications via various transmission media. Typical applications range from acoustic echo cancellers for full-duplex speakerphones to video deghosting systems for terrestrial television broadcasts to signal conditioners for wire line modems and wireless telephony. The effect of an equalization system is to compensate for transmission-channel impairments such as frequency-dependent phase and amplitude distortion. Besides correcting for channel frequency-response anomalies, the equalizer can cancel the effects of multipath signal components, which can manifest themselves in the form of voice echoes, video ghosts or Raleigh fading conditions in mobile communications channels. Equalizers specifically designed for multipath correction are often termed *echo-cancellers* or *deghosters*. They may require significantly longer filter spans than simple spectral equalizers, but the principles of operation are essentially the same. A typical base band transmission system is depicted in Figure3.1.

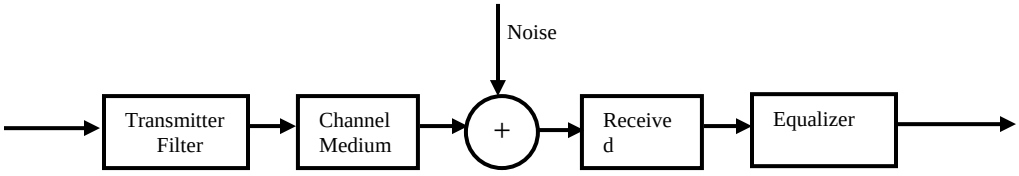


Fig.3.1 Baseband Communication System

3.2 Intersymbol Interference

When pulsed information is transmitted over an analog channel such as a phone line or airwaves Even though the original signal is a discrete time sequence, the received signal is a continuous time signal. Heuristically, one can consider that the channel acts as an analog low-pass filter, thereby spreading or smearing the shape of the impulse train into a continuous signal whose peaks relate to the amplitudes of the original pulses. Mathematically, the operation can be described as a convolution of the pulse sequence by a continuous time channel response. The operation starts with the convolution integral:

$$r(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

Where $r(t)$ is the received signal, $h(t)$ is the channel impulse response, and $x(t)$ is the input signal. The second half of the equation above is a result of the fact that convolution is a commutative operation.

Component $x(t)$ is the input pulse train, which consists of periodically transmitted impulses of varying amplitudes. Therefore,

$$x(t) = 0 \quad for \quad t \neq KT$$
$$x(t) = X_k \quad for \quad t = KT \tag{3.1}$$

Where T represents the *symbol* period. This means that the only significant values of the variable of integration in the above integral are those for which $\tau = KT$. Any other value of τ amounts to multiplication by 0. Therefore $r(t)$ can be written as

$$r(t) = \sum_{k=-\infty}^{\infty} x_k h(t - kT) \tag{3.2}$$

This representation of $r(t)$ more closely resembles the convolution sum familiar to DSP engineers. Note, however, that it still describes a continuous time system. It shows that the received signal consists of the sum of many scaled and shifted continuous time system impulse responses. The impulse responses are scaled by the amplitudes of the transmitted pulses of $x(t)$.

3.2.1 Symbol Overlap

The expected number of errors can be calculated by considering the amount of symbol interaction, assuming Gaussian noise. Taking any two neighboring symbols, the cumulative distribution function (CDF) can be used to describe the overlap between the two noise characteristics. The overlap is directly related to the probability of error between the two symbols and if these two symbols belong to opposing classes, a class error will occur.

Figure 3.2 shows two Gaussian functions that could represent two symbol noise distributions. The Euclidean distance, L, between symbol canters and the noise variance, σ^2, can be used in the cumulative distribution function of Equation to calculate the area of overlap between the two symbol noise distributions and therefore the probability of error, as in Equation (3.3)

$$CDF(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{x^2}{2\sigma^2}\right] dx$$
$$P(e) = 2CDF\left(\frac{L}{2}\right) \tag{3.3}$$

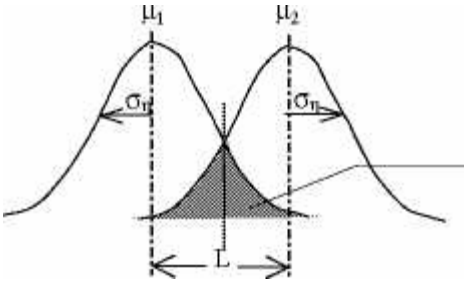


Figure 3.2 Interaction between two neighboring symbols.

Since each channel symbol is equally likely to occur [2.3], the probability of unrecoverable errors occurring in the equalization space can be calculated using the sum of all the CDF overlap between each opposing class symbol. The probability of error is more commonly described as the BER. Equation 3.4 describes the BER based upon the Gaussian noise overlap, where N_{sp} is the number of symbols in the positive class, N_m is the number of number of symbols in the negative class and Δ_i, is the distance between the i th positive symbol and its closest neighboring symbol in the negative class.

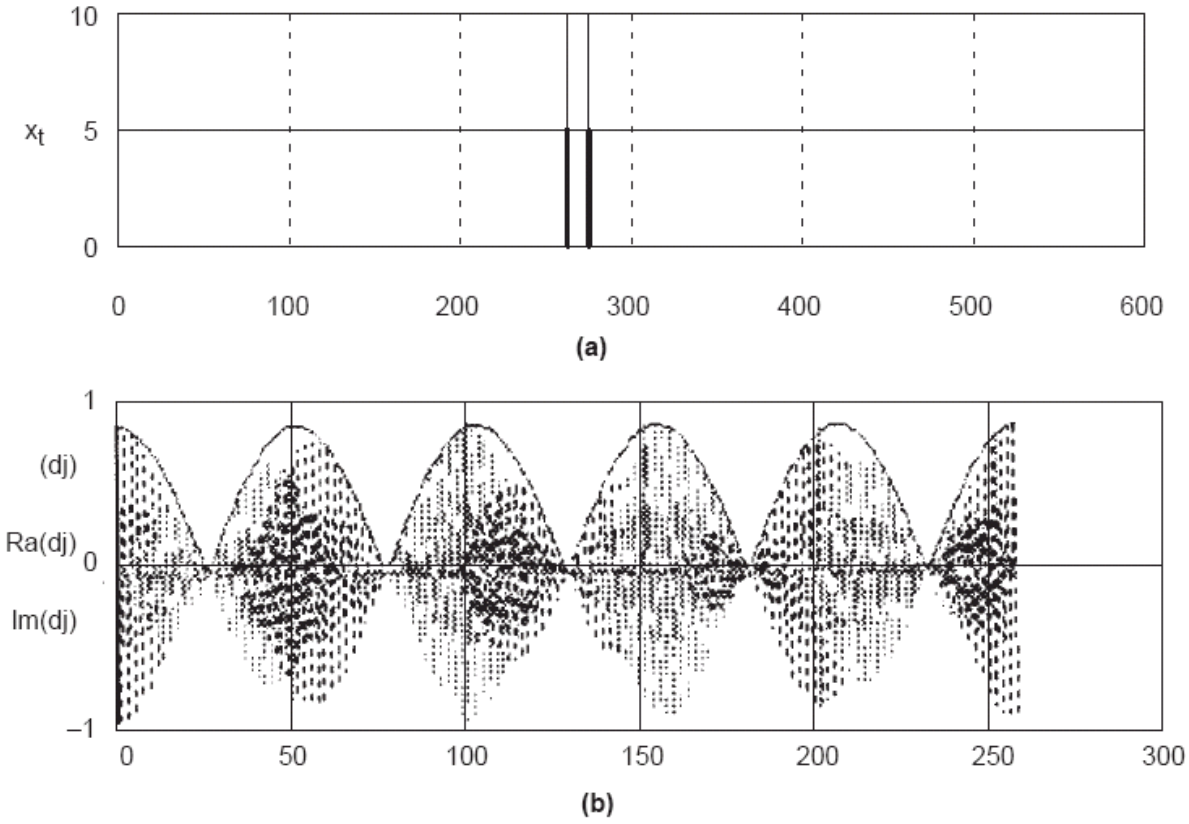
$$BER(\sigma_n) = \log\left[\frac{2}{N_{sp}+N_m}\sum_{i=1}^{N_{sp}} CDF\left(\frac{\Delta_i}{2\sigma_n}\right)\right] \tag{3.4}$$

3.3 Multipath Effects On Frequency Response

Multipath effects describe the situation in which there are several propagation paths from transmitter to receiver. Most commonly, this results when there are reflected signals detected at the receiver following the direct path. The multipath phenomenon can be modeled by an FIR system. The center tap represents the direct path, while the succeeding tap weights represent the amplitudes, delays, and phases of the reflected paths.

Figure 3.3(a) shows the time response of a system that contains a single multipath channel. The first nonzero sample of the response represents the direct path, while the second represents a delayed path to the receiver. In this instance, the pulses are identical in amplitude and phase and are separated by ten sample intervals. Notice in Figure 3.3(b) that the magnitude response exhibits $t_0/2$ nulls, where t_0 represents the sample delay. Even though you are effectively

Fig.3.3 System with a single Unattenuated Multipath channel



adding two identical flat spectra, the time delay results in a phase delay in the spectral domain. This phase delay results in nulls where the two signals are of equal amplitude but opposite phase. Obviously, multipath effects can have major effects on the system spectral response, thereby providing another justification for channel equalization.

3.4 Minimum And Nonminimum Phase Channels

When all the roots of the model z-transform lie within the unit circle, the channel is termed minimum phase. [2.5] The inverse of a minimum phase channel is convergent, illustrated by Equation (3.5):

$$\begin{aligned} H(z) &= 1.0 + 0.5z^{-1} \\ \frac{1}{H(z)} &= \frac{1}{1.0 + 0.5z^{-1}} \\ &= \sum_{i=0}^{\infty} \left(-\frac{1}{2} \right)^i z^{-i} \\ &= 1 - 0.5z^{-1} + 0.25z^{-2} - 0.125z^{-3} + \cdots \end{aligned} \tag{3.5}$$

Where as the inverse of non-minimum phase channels are not convergent, as shown in Equation (3.6)

$$\begin{aligned} H(z) &= 0.5 + 1.0z^{-1} \\ \frac{1}{H(z)} &= \frac{z}{1.0 + 0.5z} \\ &= z.\left[\sum_{i=0}^{\infty}\left(-\frac{1}{2}\right)^{i} z^{-i}\right] \\ &= z.\left[1 - 0.5z + 0.25z^{2} - 0.125z^{3}\right] \end{aligned} \tag{3.6}$$

Since equalizers are designed to invert the channel distortion process they will in effect model the channel inverse. The minimum phase channel has a linear inverse model therefore a linear equalization solution exists. However, limiting the inverse model to m-dimensions will approximate the solution and it has been shown that non-linear solutions can provide a superior inverse model in the same dimension [2.2].

A linear inverse of a non-minimum phase channel does not exist without incorporating time delays. A time delay creates a convergent series for a non-minimum phase model, where longer delays are necessary to provide a reasonable equalizer. Equation (3.7) describes a non-minimum phase channel with a single delay inverse and a four-sample delay inverse. The latter of these is the more suitable form for a linear filter.

$$\begin{aligned} H(z) &= 0.5 + 1.0z^{-1} \\ z^{-1}\frac{1}{H(z)} &= \frac{1}{1 + 0.5z} = 1 - 0.5z + 0.25z^{2} - 0.125z^{3} + \cdots (\textit{non causal}) \\ z^{-4}\frac{1}{H(z)} &= z^{-3} - 0.5z^{-2} + 0.25z^{-1} - 0.125z + \cdots (\textit{truncated and causal}) \end{aligned} \tag{3.7}$$

The three-tap non-minimum phase channel $H(z) = 0.26 + 0.93z^{-1} + 0.26z^{-2}$ is used throughout this thesis for simulation purposes. A channel delay, D is included to assist in the classification so that the desired output becomes $u(n - D)$.

3.5 Channel Equalization

Two main techniques are employed to formulate the filter coefficients: *automatic synthesis* and *adaptation*. In automatic-synthesis methods, the equalizer typically compares a received time-domain reference signal to a stored copy of the undistorted training signal. By comparing

the two, a time-domain error signal is determined that may be used to calculate the coefficient of an inverse filter. The formulation of this inverse filter may be accomplished strictly in the time domain, as is done in the LMS systems, which are examined in more detail in following sections. Other methods involve conversion of the received training signal to a spectral representation. A spectral inverse response can then be calculated to compensate for the channel response. This inverse spectrum is then converted back to a time-domain representation so that filter tap weights may be extracted.

3.5.1 LMS Equalization

The least mean squared (LMS) equalizer is a more general approach to automatic synthesis. The coefficients are gradually adjusted to converge to a filter that minimizes the error between the equalized signal and the stored reference. The filter convergence is based on approximations to a gradient calculation of the quadratic equation representing the mean square error. The beauty of the approach is that the only parameter to be adjusted is the adaptation step size. Through an iterative process, all filter tap weights are adjusted during each sample period in the training sequence. Eventually, the filter will reach a configuration that minimizes the mean square error between the equalized signal and the stored reference. As might be expected, the choice step size involves a tradeoff between rapid convergence and residual steady-state error. A too-large setting for step size can result in a system that converges rapidly on start-up, but then chops around the optimal coefficient settings at steady state.

The optimal BER equalization performance is obtained using a maximum likelihood sequence estimator (MLSE) on the entire transmitted data sequence [2.7]. A more practical MSE would operate on smaller data sequences but these can still be computationally expensive, they also have problems tracking time-varying channels and can only produce sequences of outputs with a significant time delay. Another equalization approach implements a symbol-by-symbol detection procedure and is based upon adaptive filters [2.1]. The symbol-by-symbol approach to equalization applies the channel output samples to a decision classifier that separates the symbol into their respective classes.

3.6 Summary

To compensate the ISI, Multipath channel effects on frequency response and other types of noise effects an equalizer placed at the receiver end. Since equalizer comes under inverse modeling it is difficult to design. Proper care is taken in choosing the while training the channel. LMS types

equalizer performs well in case of linear channels but its performance degrades while the channel becomes nonlinear. So different nonlinear structures are being used to design nonlinear equalizer like MLP, RBF, FLANN and many more.

Chapter 4

TIME DOMAIN BLOCK ADAPTIVE FILTER

4.1 Introduction

Adaptive filters are digital filters self adjust its transfer function according to an optimizing algorithm with the change in their input signals. The adaptive filter adjusts its coefficient to minimize the mean square between its output and that of an unknown system.

Block digital filtering calculates a block of data from a finite set of filter outputs from a block of input values. Block adaptive filter adjusts its filter coefficients once per each block according to some optimizing algorithm such as LMS or RLS [4.1]-[4.2]. Hence the traditional LMS adaptive filter, which adjusts the weights once each data sample, is a special case of block adaptive filter with a block length one. Block implementation of adaptive digital filter permits fast implementation while maintaining the performance equivalent to that of widely used LMS adaptive filter. Also efficient block algorithms such as Fast Fourier Transform (FFT) can be used when implementing filters in serial processors [4.3]-[4.5].

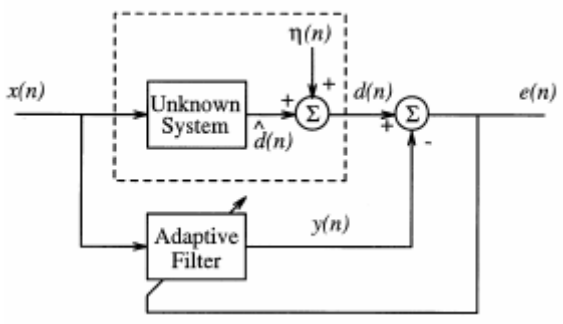


Fig.4.1 Basic Adaptive Digital Filter

Fig.4.1 shows the basic adaptive digital filter of LMS type presented by Widrow [4.6]-[4.8] for which the performance index is mean square error (MSE= ξ). All the inputs are real and for the FIR digital filter of order $N-1$ for which the output y_k at discrete instant k is given by the convolution sum of the input x_k and the filter weights w_{lk} :

$$y_k = \sum_{l=1}^{N} w_{lk} x_{k-l+1}\ , \qquad k = 1,2,3,.....,\tag{4.1}$$

The Widrow-Hoff LMS algorithm adjusts the filter weights in accordance with:

$$W_{k+1} = W_k + 2\mu\varepsilon_k X_k \tag{4.2}$$

Where μ is the convergence constant, and W_k and X_k are, respectively, the weight vector and the $N \times 1$ input vector:

$$W_k \overset{\Delta}{=} [w_{1k}\ w_{2k}\ \ldots . w_{Nk}]^T$$

$$X_k \overset{\Delta}{=} [x_k\ x_{k-1}\ \ldots . x_{k-N+1}]^T$$

and ε_k is the error vector at the kth instant given by the difference between the desired output d_k and the actual output y_k that is:

$$\varepsilon_k \overset{\Delta}{=} d_k - y_k \tag{4.3}$$

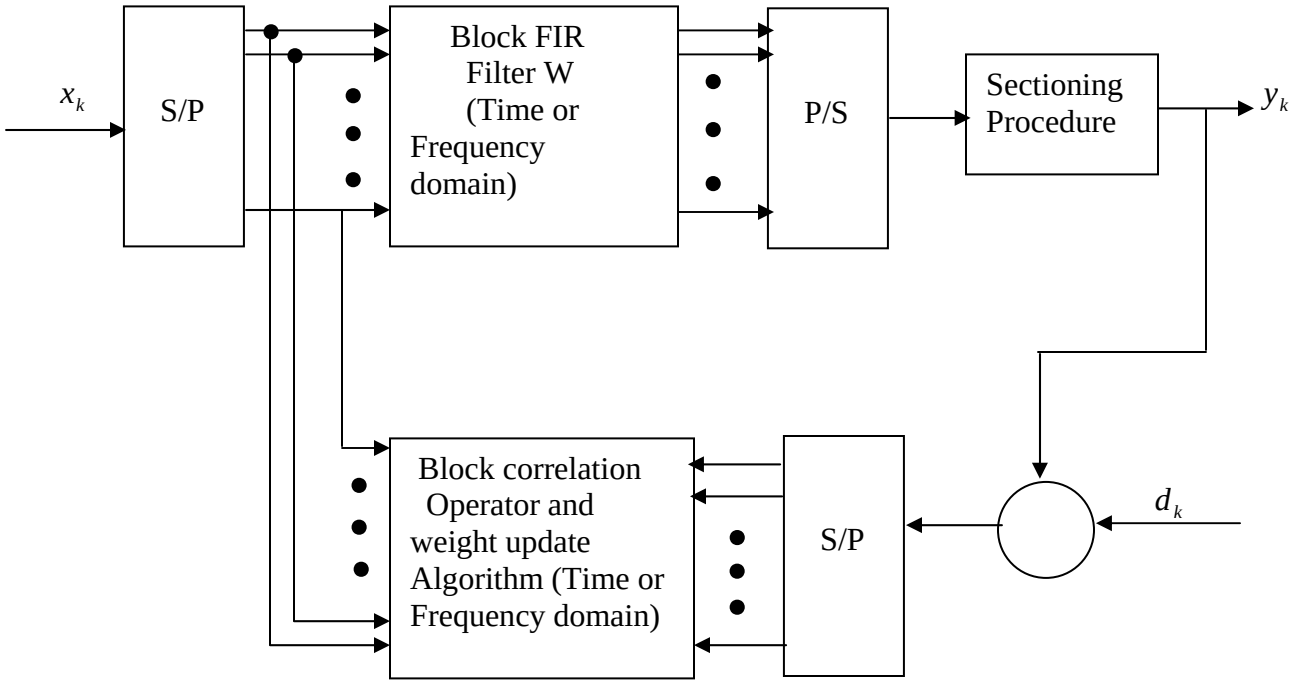


Fig.4.2 General block adaptive filtering configuration.

S/P = Serial to Parallel converter

P/S = Parallel to Serial converter

Fig.4.2 shows the basic block diagram of the Block Adaptive Digital Filter for which a block mean square error (BMSE) performance criterion is defined, resulting in a BMSE gradient estimate that is a correlation (over a block of data) between the error and the input signal. This gradient estimate leads to a weight adjustment that allows the block implementation with either parallel processor or serial processor and the FFT.

For the time invariant case, (4.1) can be written in matrix form as:

$$y_k = W^T X_k = X_k^T W . \tag{4.4}$$

Letting L represents block length and the following low order $(L = 3, N = 3)$ example shows how this convolution is written in block form:

$$\begin{matrix} x_1\{ \\ x_2\{ \end{matrix} \begin{bmatrix} x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & x_2 & x_1 \\ x_4 & x_3 & x_2 \\ x_5 & x_4 & x_3 \\ x_6 & x_5 & x_4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{matrix} \} Y_1 \\ \} Y_2 \end{matrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} \tag{4.5}$$

Using the block notation the output can be written as:

$$Y_j = x_j W \tag{4.6}$$

Where j are the block index, Y_j and x_j are, respectively, the jth output vector of length L, and the $L \times N$ matrix of input vectors:

$$Y_j \stackrel{\Delta}{=} [y_{(j-1)L+1} \; y_{(j-1)l+2} \; \ldots . y_{jL}]^T$$
$$x_j \stackrel{\Delta}{=} [X_{(j-1)L+1} \; X_{(j-1)L+2} \; \ldots . X_{jL}]^T \quad .$$

4.2 The Block Wiener Filtering Problem

The classical Wiener filter can be extended to the block input case as shown next. Refer to Fig. 4.1. Assuming that all the inputs are stationary let

$$d_j \stackrel{\Delta}{=} [d_{(j-1)l+1} \;\; d_{(j-1)l+2} \;\; \ldots d_{jl}]^T \tag{4.7}$$

be the $L \times 1$ vector of desired responses for block j, and let

$$\varepsilon_j \stackrel{\Delta}{=} [\varepsilon_{(j-1)l+1} \;\; \varepsilon_{(j-1)L+2} \;\; \ldots \varepsilon_{jl}]^T \tag{4.8}$$

be the $L \times 1$ vector of errors for block j where ε_k is defined in (4.3) The key element of this analysis is the performance index, chosen to be a combination of the standard MSE and the sum square error used in least squares problems [16]. BMSE is defined by

$$\text{BMSE} = \equiv \stackrel{\Delta}{=} \frac{1}{L} E\left[\varepsilon_j^T \varepsilon_j\right] = E\left[\frac{1}{L} \sum_{k=(j-1)l+1}^{jl} \varepsilon_k^2\right]. \tag{4.9}$$

Clearly, the BMSE is the expected value of a smoothed estimate of the squared error over one block. This is appropriate because it combines a block's worth of error information into one

number for each value of the block index j. The MSE considers the error information one point at a time. Using (4.6) and (4.8), (4.9) becomes

$$L \equiv= E\{d_j^T d_j\}W - W^T E\{x_j^T d_j\} + W^T E\{x_j^T x_j\}W$$

Where d_j is the jth desired output block.

The following correlation matrices are now defined:

$$R = E[X_k X_k^T] \tag{4.10}$$

$$P = E[X_k d_k] \tag{4.11}$$

$$\begin{aligned} \Re &= E[x_j^T x_j] \\ \wp &= E[x_j^T d_j]. \end{aligned} \tag{4.12}$$

Note that R and $\Re$, are, respectively, the $N \times N$ input autocorrelation and block input correlation matrices. Likewise, P is the $N \times 1$ cross-correlation vector between the input and the desired response whereas $\wp$ is the $N \times 1$ cross-correlation vector between the block input and the block desired response. Using these definitions, the BMSE can be written compactly as:

$$\equiv= \frac{1}{L}\left[E\{d_j^T d_j\} - \wp^T W - W^T \wp + W^T \Re W\right] \tag{4.13}$$

Using (4.6) and (4.12), and invoking stationarity, it can be shown that $\Re = LR$. A similar argument reveals that $\wp = LP$. Taking advantage of the above, (4.13) can be re-written as

$$\equiv= E[d_k^2] - 2P^T W - W^T RW = \xi \tag{4.14}$$

Thus the BMSE is equal to the MSE when the inputs are stationary. It follows then that the optimal set of filter weights W^* for the block Wiener filter is the same as for the Wiener filter, i.e.,

$$W^* = R^{-1}P \tag{4.15}$$

This can be shown using an extension of the orthogonality principle, which states that the weight vector W^* minimizing the BMSE is the one for which the error vector ε_j, is orthogonal to the block data x_j. Also the minimum BMSE $Z_{\min}$ is given by

$$\frac{1}{L}E\left[d_j^T \varepsilon_j\right]$$

and is equal to $\xi_{\min}$. The Wiener filtering problem is clearly a special case (for $L = 1$) of the block Wiener problem.

4.3 Block Adaptive Filtering and the BLMS Algorithm

Analogous to LMS adaptive filtering, a block wise algorithm can be derived to sequentially to solve for the Wiener weight vector in real time by a gradient search technique. Because it is desired to keep the weights constant while each block of data is being processed, the weight vector is adjusted once per data block rather than one per data sample as in the LMS algorithm. The algorithm then becomes

$$W_{j+1} = W_j - \mu_B \Delta_{Bj} \tag{4.16}$$

Where μ_B is the convergence constant, Δ_{Bj} is the $N \times 1$ BMSE gradient at block j, and W_j is the $N \times 1$ weight vector at block j. The gradient is taken with respect to the weights as follows:

$$\Delta_{Bj} \triangleq \frac{1}{L} \frac{\partial E[\varepsilon_j^T \varepsilon_j]}{\partial W}\bigg|_{W=W_j} \tag{4.17}$$

Because the computation of an ensemble average is difficult, an estimate of the gradient $\hat{\Delta}_{Bj}$ *is* used in place of Δ_{Bj}. The BMSE gradient estimate at block j is defined as

$$\hat{\Delta}_{Bj} \triangleq \frac{1}{L} \frac{\partial \varepsilon_j^T \varepsilon_j}{\partial W_j} = -\frac{2}{L} x_j^T \varepsilon_j \tag{4.18}$$

Use of this unbiased block gradient estimate in the weight adjustment algorithm (4.16) gives the block least mean square (BLMS) algorithm:

$$\begin{aligned} W_{j+1} &= W_j + \frac{2\mu_B}{L} x_j^T \varepsilon_j \\ &= W_j + \frac{2\mu_B}{L} \sum_{k=(j-1)l+1}^{jl} \varepsilon_k x_k = W_j + \frac{2\mu_B}{L} \phi_j \end{aligned} \tag{4.19}$$

The BLMS algorithm is identical to the LMS algorithm when the block length L is equal to one. Also, the weight update term in (4.19) is an average of the L LMS-like terms $\varepsilon_k X_k$, generated by a block of data. Consider ϕ_j written out for the *ith* weight:

$$\phi_{ij} = \sum_{k=(j-1)l-i+1}^{jl} \varepsilon_k x_{k-i+1} \qquad i = 1,2,....,N.$$

Substituting $n = k - i + 1$, this becomes

$$\begin{aligned} \phi_{ij} &= \sum_{n=(j-1)L-i+2}^{jL-i+1} \varepsilon_{n+i-1} x_n \qquad i = 1,2,....,N. \\ &= e_{-i} * x_i \end{aligned}$$

Where $*$ indicates convolution. Clearly, the weight update term is a correlation, implementable in block form with a parallel processor or with a serial processor and the FFT.

For BLMS adaptive filtering, both the convolution (4.6) and the weight update can be realized in block form, whereas neither can be realized in block form for LMS adaptive filtering.

The choice of block length is important. Examination of (4.19) reveals that the algorithm is valid for any block length greater than or equal to one; however, the L equal's N case is probably preferred in most applications. This is because for L greater than N, the gradient estimate, which is computed over L input points, uses more input information than the filter W uses, resulting in redundant operations. For L less than N, the filter length is larger than the input block being processed, which is a waste of filter weights.

4.4 Convergence Properties of The BLMS Algorithm

The convergence properties of interest in adaptive filtering are the required bounds on the convergence constant (μ or μ_B), adaption speed and adaption accuracy. Adaption speed refers to how fast the MSE is reduced to an estimate of the minimum MSE (MMSE or $\xi_{\min}$). The measure of how close the solution is to $\xi_{\min}$ (adaptation accuracy) is called misadjustment and is defined as average excess MSE divided by $\xi_{\min}$. These convergence properties are examined for block adaptive filters, and compared with the corresponding properties of conventional LMS adaptive filters.

4.4.1 Bounds On μ_B To Guarantee Convergence

It has been proved that the BLMS algorithm converges. The approach taken is to show that as the block number j approaches infinity, the expected value of the weight vector ($E[W_{j+1}]$) approaches the Wiener weight vector under the assumption that x_j and d_j are ergodic and that $E[x_j^T x_{j+1}] \approx 0$ for $l \neq 0$. The proof also shows that the requirements on the convergence constants (μ for LMS, for μ_B BLMS) are the same, that is, μ and μ_B must take on values in the same range in order to guarantee convergence of the respective algorithms. The bounds on the convergence constants are:

For LMS: $$0 < \mu < \frac{1}{\lambda_{\max}} \tag{4.20}$$

For BLMS: $$0 < \mu_B < \frac{1}{\lambda_{\max}} \tag{4.21}$$

Where $\lambda_{\max}$, is the largest eigen value of the matrix R.

4.4.2 Adoption Speed

Adoption speed is given in terms of a time constant, which indicates how fast the weight vector converges to the Wiener weight vector (see Fig.4. 2)**.** Actually, there are N time constants T_{pMSE}**,** one, for each (Pth) mode of the difference equation describing the adaption process. The derivations follow the form of the corresponding derivations for the LMS algorithm, but with some very important differences. The convergence constant μ (for LMS) is replaced by μ_B (for BLMS). The time unit for LMS is sample number (k) `where` as the time unit for BLMS is block number (j). Thus the equations for the two different algorithms have the same form, but much different meanings. This difference is resolved by converting the BLMS time constants to units of sample number so comparison with LMS time constants becomes meaningful. For the special case in which all eigen values of the input autocorrelation matrix R are equal, the N time constants can be lumped into one, giving τ_{MSE} for LMS and T_{MSE} for BLMS. It is shown that:

For BLMS:

$$T_{pMSE} = \frac{L}{4\mu_B \lambda_p}, T_{MSE} = \frac{NL}{4\mu_B trR} \tag{4.22}$$

For LMS:

$$\tau_{pMSE} = \frac{1}{4\mu\lambda_p}, \tau_{MSE} = \frac{N}{4\mu trR} \tag{4.23}$$

Where $\lambda_p = Pth$ eigen value of $R(p = 1,2,..., N)$ and trR is the trace of R or the sum of the diagonal elements of R.

4.4.3 Adoption Accuracy

Adoption accuracy, or a measure of the weight noise, is measured by misadjustment, defined as follows

For BLMS:

$$m \stackrel{\Delta}{=} \frac{AverageExessBMSE}{\xi_{\min}} \tag{4.24}$$

For LMS:

$$M \stackrel{\Delta}{=} \frac{AvrageExessBMSE}{\xi_{\min}} \tag{4.25}$$

The misadjustment is caused by gradient noise in the BLMS or LMS algorithm. Misadjustment for the BLMS algorithm is derived in [4.11], where it is shown that:

For BLMS:

$$\text{Average Excess BMSE} = \frac{\mu_B}{L}\xi_{\min} trR, m = \frac{\mu_B}{L} trR\, L \tag{4.26}$$

For LMS:

$$\text{Average Excess MSE } = \mu\xi_{\min} trR, m = \mu trR \tag{4.27}$$

4.4.4 Comparison of Convergence Properties For The LMS And BLMS Algorithms

Comparing the quantities presented above by taking ratios yields some interesting properties:

$$\frac{T_{pMSE}}{\tau_{pMSE}} = \frac{L\mu}{\mu_B} and \frac{m}{M} = \frac{\mu_B}{L\mu} \tag{4.28}$$

Hence it is observed that the BLMS and LMS algorithms converge at the same rate and achieve the same misadjustment if $\mu_B = L\mu$.

In using these relations for design purposes, one must remember that μ_B and μ have the same convergence bounds, because this fact limits the usable block length. For example, a possible situation is that $\mu_B = L\mu$ and, μ satisfies (4.20), but μ and L are so large that (4.21) is not satisfied. This is less likely to occur, of course, for the case of slow adaption than for the case of fast adaption.

All the relations regarding BLMS convergence reduce to the LMS case when the block length *L* equals one.

Convergence Properties When Data is correlated

Adaptive filter performance equations are traditionally derived assuming uncorrelated inputs because that is the case that is easily tractable. The convergence proof and derivations of convergence parameters for the BLMS algorithm are based on the assumption that the input matrices x_j *and* x_{j+1} are uncorrelated. For the LMS algorithm, the assumption is that X_k, and X_{k+1} are uncorrelated. These assumptions lead to the assumptions that W_j is independent of x_j for the BLMS algorithm and W_k is independent of X_k, for the LMS algorithm. These assumptions simplify the proofs but are not appropriate for all data types. Both proofs also assume input stationary.

Based upon the work of Gersho, Kim and Davisson use a sample average over a block of L data points to estimate the MSE gradient for adjusting the weights of an adaptive algorithm. They assume that the input data is M -dependent, which basically means that it is uncorrelated for autocorrelation lags greater than M (where M is a positive integer). Kim and Davisson show that when the inputs are M -dependent and the filter weights are adjusted once per block, the problems of analyzing convergence are overcome if $L \geq (M + N - 1)$, where N is the filter length and M is the M -dependence constant.

Keeler studied the, adaptive predictor with the LMS algorithm modified so that it adjusts the weights only once per h input samples. He showed that convergence when the inputs are correlated could be analyzed if h is chosen to be sufficiently large.

The point to remember about the above discussion is that block adaptive filtering has an analysis advantage over LMS filtering when inputs are correlated and fit the M dependence condition simply because the weights are adjusted once per block.

4.5 Computational Complexity of LMS and BLMS Adaptive Filtering

The main computational efficiency issues involved in algorithm implementation are storage (memory), time (number of machine cycles for CPU, input-output, etc.), and computational complexity measured in the number of real multiply and additions required. Because the first two issues are processor architecture-dependent, hence I concentrate on the computational complexity required when using a standard serial-type processor. This is done for convenience, even though the most efficient implementation of BLMS adaptive filters is probably with parallel processors.

4.5.1 Computational Complexity Of LMS Adaptive Filters

The convolution operation (1) is done in direct form. To produce one output point requires N real multiplies and $N-1$ real adds. Thus to produce L output points requires LN real multiplies and $L(N-1)$ real adds.

To produce L output points (one block) using the LMS algorithm requires LN adaptions. The term $(2\mu\varepsilon_k)X_k$ requires $L(N+1)$ real multiplies per block. The addition operation requires LN real additions per block. The cost of computing $\varepsilon_k = d_k - y_k$, is L real adds per block. The total cost per block for the LMS algorithm is $L(N+1)$ real multiplies and $L(N+1)$ real additions. Thus, the total computational complexity of LMS adaptive filtering is $L(2N+1)$ real multiplies and $2LN$ real adds. This result is shown in Table [2]

4.5.2 Computational Complexity of BLMS Adaptive Filters

The convolution operation is implemented directly, so it is the same as for the LMS case.

From (4.19) the weight update term ϕ_j requires LN real multiplies and $N(L-1)$ real adds per block. Adding $(2\mu_B / L)\phi_j$ to W_j require N adds per block. Calculation of $2\mu_B / L$ requires two multiplications, but this is true only for the first block, so it is ignored in the general count. Calculation of $(2\mu_B / L).\phi_j$ requires N real multiplies per block. The cost of calculating $\varepsilon_k = d_k - y_k$ is L real adds per block. Thus the total complexity for standard BLMS adaptive filtering is $N(2L+1)$ real multiplies and $2LN$ real adds per block.

4.5.3 Complexity Analysis

There is very little complexity difference between LMS and direct BLMS filtering. Therefore, the comparisons of interest are between the LMS adaptive filter and the two fast implementations of the BLMS adaptive filter are discussed above. A complexity ratio CR is computed and tabulated versus the block length L in Table **111** for these implementations.

$$CR = \frac{ComplexityofLMSFiltering}{ComplexityofBLMSFiltering} \tag{4.29}$$

Only the $L = N$ case is analyzed because it provides for the most efficient use of the input data (Section 111). **As** discussed in the Appendix, the convolution implementations require sequence lengths of $N^{'} \geq L + N - 1$ because N' must be a power of two for the equations in Table 11, and $L = N$ is assumed, $N' = 2N$ is used for simplicity in the complexity ratio calculations.

4.6 Simulation Study and Discussion

Extensive computer simulations were carried out using the two structures using LMS algorithm. For both System Identification and Channel Equalization problem, a uniformly distributed random signal over the interval [-.5,.5] was applied to the FIR structure and a white Gaussian noise of 30dB was added to the output of the system. The learning parameter μ both for LMS and BLMS algorithm was suitably chosen to obtain best result.

Four different channels were studied with the following transfer function:

$CH = 1 : 0.209 + 0.995z^{-1} + 0.209z^{-2}$

$CH = 2 : 0.260 + 0.930z^{-1} + 0.260z^{-2}$

$CH = 3 : 0.340 + 0.903z^{-1} + 0.304z^{-2}$

$CH = 4 : 0.341 + 0.876z^{-1} + 0.341z^{-2}$

To study the effect of nonlinearity on the system performance four different nonlinear channel models with the following nonlinearities has been introduced.

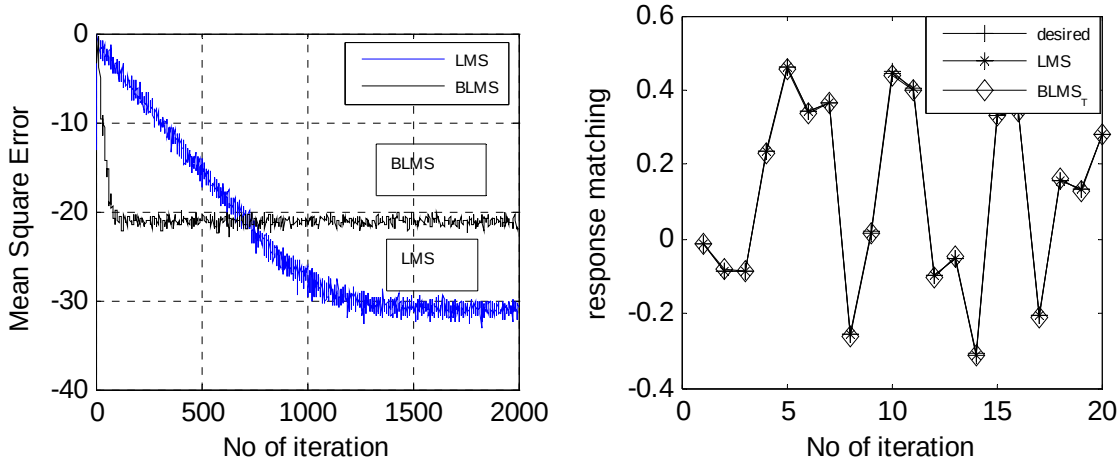
$NL = 0 : b(k) = a(k)$

$NL = 1 : b(k) = \tanh(a(k))$

$NL = 2 : b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k)$

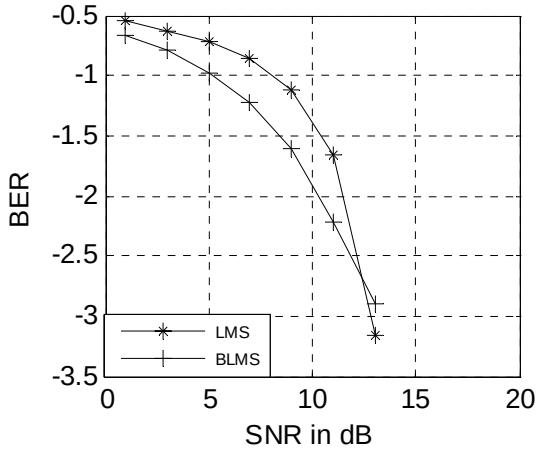
$NL = 3 : b(k) = a(k) - 0.9(a^3(k))$

The comparison of the LMS and BLMS algorithm for the convergence characteristics and response-matching plot for System Identification and BER plot for the entire linear and nonlinear channel model has been given. Simulation result for channel 2 with 30 dB noise has been simulated for different linear and non linear channel has been studied. From Fig**.** it is seen that convergence characteristics of BLMS algorithm faster converges than the LMS algorithm while from response matching plot for both the case is same. From BER plots it is seen that BLMS algorithm performs better than the LMS algorithm for all the linear and nonlinear channels.

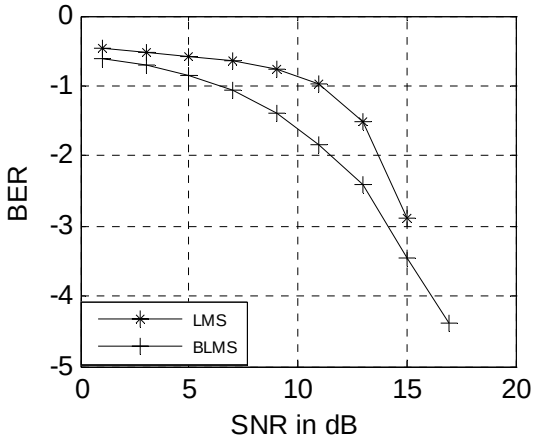


4.3(a) 4.3(b)

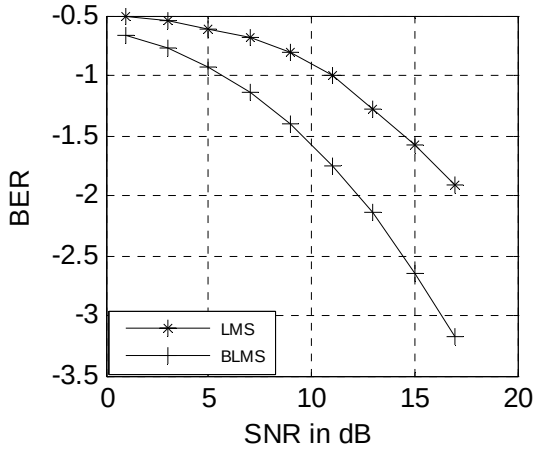
Fig 4.3(a) ,(b) are the corresponding MSE and Response matching plot for System Identification problem for LMS and BLMS algorithm without noise condition



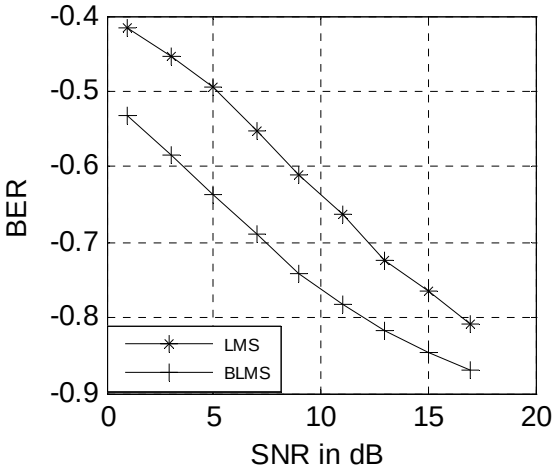
4.4(a)



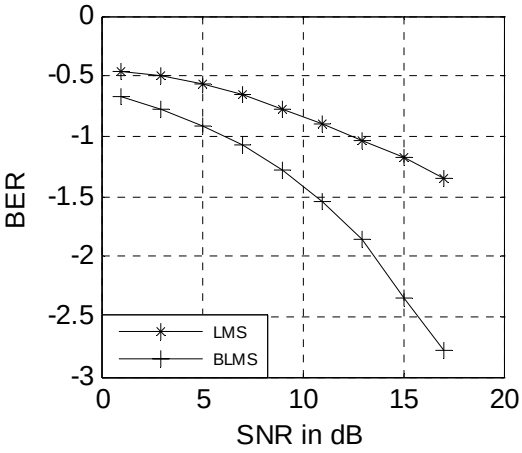
4.4(b)



4.4(c)



4.4(d)



4.4(e)

Fig.4.4 (a),(b),(c),(d),(e) ,corresponds to the respective BER plot for LMS and BLMS Equalizer structure for NL=0,NL=1,NL=2,NL=3 and NL=4

4.7 Conclusion

Here a BAF was derived which allows fast implementation while maintaining performance equal to that of the LMS adaptive filter in case of System Identification problem and better in Channel Equalization problem. It was seen that BLMS Adaptive filters have an advantage over an LMS algorithm when the inputs are correlated and finally BLMS involves less computational complexity when implemented using efficient convolution algorithms on serial processors.

Chapter 5

FREEQUENCY DOMAIN BLOCK ADAPIVE FILTER

5.1 Introduction

Adaptive filtering in the frequency domain is the Fourier Transform of the input signal and independent weightings of the contents of each frequency bin. FIR adaptive filters can be implemented efficiently in the time domain as well as in the frequency domain. Frequency domain filter performs in a manner similar to a conventional time domain adaptive transversal filter with a significant reduction in computational complexity. Using an FDAF results in block processing in which one block of input data is processed simultaneously, producing one block of output data. With different efficient algorithm such as Fast Fourier Transform (FFT) block processing is done. In this way, the amount of computational requirements in terms of multiply-adds per one block of N output samples can be greatly reduced compared with time-domain approaches. This is accomplished by replacing convolution with a multiplication of transforms, which implies a complexity reduction from $O(N^2)$ *to* $O(N\log(N))$.

The same frequency-domain filter is obtained whether time-domain mean-square error or frequency-domain mean square error is minimized, as long as data sectioning is done correctly. In other words, the time-domain block adaptive filter implemented in the frequency domain is equivalent to the frequency-domain adaptive filters (derived in the frequency domain) provided data sectioning is done properly. Data sectioning is the procedure for breaking the continuous data stream into blocks before processing and reassembling the resulting data blocks into a continuous data stream after processing. All block algorithms require such a procedure, whether they process the data in the time domain or the frequency domain, and the sectioning procedures must be carefully integrated into the processing algorithms. The two most common sectioning procedures are the overlap-save and overlap-add methods [5.1] - [5.3].

A large number of equivalent time- and frequency-domain filter structures are possible for the basic block adaptive filter. It can be shown that all of the frequency-domain adaptive filters in the literature [5.4] - [5.5] belong to the set of possible block adaptive filter structures, and that they differ only in the sectioning procedures used.

From the two most common sectioning procedures (overlap-save and overlap-add) in the frequency domain adaptive filter it shows that the overlap-save method is preferred over the overlap-add method because it requires fewer computations. Sectioning procedures are well known and straightforward for fixed coefficient filters. However, this is not the case for adaptive filters because of the necessity of changing the filter coefficients at each iteration of the filter, so special care must be exercised.

5.2. The Equivalence of Time And Frequency-Domain FIR Adaptive Filter

We have discussed the BAF in time domain in chapter-4, here we will discuss the weight adjustment in the frequency domain. Many such implementations exist, depending upon how the convolution, correlation and weight adjustment operation are realized by various combinations of time- and frequency- Domain methods.

The frequency-domain adaptive filter is derived from two perspectives: first, as an implementation of the time-domain block adaptive filter, and second, as the frequency-domain adaptive filter considered entirely in the frequency domain. From simulation study it is seen that the two filters are equivalent, provided data sectioning is done properly.

5.2.1 Implementation of the Time-Domain Block Adaptive Filter in the Frequency Domain

The $N' \times 1$ augmented vectors shall be denoted in general by $X_j^a, \underline{W}_j^a, \underline{Y}_j^a, \underline{\varepsilon}_j^a$ *and* $\underline{\phi}_j^a,$, but their actual contents depend upon whether overlap-save or overlap-add sectioning is used. x_j^a is an $N' \times N'$ circulant matrix composed of augmented input vectors. In this section, the contents of these augmented vectors/matrices are not unspecified, in the latter sections contents are given. *Frequency-Domain Notation:* Define the following single sample notation for the DFT of the above augmented vectors, where $l = 0, 1, 2, \ldots, N' - 1$ is the discrete frequency index.

$$X_j(l) = DFT\{X_j^a\} \tag{5.1}$$

$$W_j(l) = DFT\{\underline{W}_j^a\} \tag{5.2}$$

$$Y_j(l) = DFT\{\underline{Y}_j^a\} \tag{5.3}$$

$$E_j(l) = DFT\{\underline{\varepsilon}_j^a\} \tag{5.4}$$

$$\phi_j(l) = DFT\{\underline{\phi}_j^a\} = E_j(l)X_j^*(l) \tag{5.5}$$

The asterisk denotes complex conjugation. DFT $\{\underline{v}^a\}$ denotes the N' point discrete Fourier transform of the elements of the general $N' \times 1$ vector $\underline{v}^a$ for the discrete frequency index l [5.6].

Vector Frequency-Domain Notation*:* The quantities above can be written in a convenient matrix notation using the following definitions. Let $\underline{y}_j$ and $\underline{w}_j$ be $N' \times 1$ vectors of transforms, the elements of which are given by (21) and (20), respectively,

$$\begin{aligned} \underline{y}_j &= [DFT\ \{\underline{Y}_j^a\}] \\ &= [Y_j(0)\ \ Y_j(1)\ \ \ldots\ \ Y_j(N'-1)]^T \end{aligned} \tag{5.6}$$

$$\begin{aligned} \underline{w}_j &= [DFT\ \{\underline{W}_j^a\}] \\ &= [W_j(0)\ \ W_j(1)\ \ \ldots\ \ W_j(N'-1)]^T \end{aligned} \tag{5.7}$$

Let x_j be a $N' \times N'$ diagonal matrix with its nonzero elements consisting of the transforms of the inputs given by (5.1).

Let the $N' \times 1$ DFT of the augmented error vector be defined by

$$\xi_j(l) = DFT\{\underline{\varepsilon}_j^a\} \tag{5.8}$$

Frequency-Domain Weight Adjustment Algorithm: A frequency-domain adaptive filter **is** derived by augmenting the vectors of $\underline{y}_j$ and $\underline{w}_{j+1}$, and writing the DFT's of the resulting equations, using the definitions above. The result is an adaptive filter implemented as follows in single point form for $l = 0, 1, 2, \ldots, N' - 1$:

$$Y_j(l) = X_j(l) W_j(l) \tag{5.9}$$

$$W_{j+1}(l) = W_j(l) + \frac{2\mu_B}{L} \phi_j'(l) \tag{5.10}$$

Where

$$\phi_j'(l) = F\{\underline{\phi}_j(l)\}\ . \tag{5.11}$$

The notation $F\{.\}$ refers to a projection operator, which constrains to zero all but the first N time-domain values corresponding to the inverse DFT (IDFT) of the $N' \times 1$ sequence in brackets. This projection operator is necessary because the sectioning procedure used (overlap-save or overlap-add) with the DFT requires use of $N'-$ point sequences $(N' \geq L + N - 1)$ in both the time and frequency domains. Therefore, at each iteration the frequency-domain BLMS algorithm produces N' frequency domain weights, implying N' time-domain weights. This creates a problem because the filter has only N time domain weights, making the other $N' - N$ time-domain weights extraneous and harmful if used. The projection operation $F\{.\}$ above solves this problem by constraining the time-domain weights $w_{i,j}$ to be zero for $N \leq i \leq N' - 1$. It is shown in latter Sections that the implementations of $F\{.\}$ for overlap-save and overlap-add sectioning are similar. In matrix notation, (5.9) and (5.10) become

$$\underline{y}_j = x_j \underline{w}_j \tag{5.12}$$

$$\underline{w}_{j+1} = \underline{w}_j + \frac{2\mu_B}{L} F\{x_j^* \underline{\xi}_j\} \quad . \tag{5.13}$$

5.2.2 Derivation of the Frequency-Domain Adaptive Filter

The filter of Section 5.2.1-A time-domain filter is designed to minimize time-domain block mean-square error and implemented it in the frequency domain. Here, the frequency-domain adaptive filter is derived entirely in the frequency domain, using frequency-domain mean square error.

This derivation assumes no particular sectioning procedure. Although the sectioning method is unspecified, the analysis must allow for it because the frequency-domain convolutions/ correlations must correspond to time-domain linear convolutions/ correlations. Thus, all frequency-domain vectors are $N' \times 1$ and imply the use of $N' \times 1$ augmented time-domain vectors.The goal is to minimize frequency-domain sum-square error given by

$$J = \sum_{l=0}^{N'-1} \left| E_j(l) \right|^2 = \underline{\xi}_j^H \underline{\xi}_j \tag{5.14}$$

Where $E_j(l)$ is defined by *(22),* $L = N$, and the superscript H denotes the complex conjugate transpose.

The filter output in the frequency domain is given by

$$Y_j(l) = X_j(l) W_j(l), \qquad l = 0, 1, 2, \ldots, N' - 1 \tag{5.15}$$

Where $X_j(l), W_j(l)$ *and* $Y_j(l)$ are defined according to (5.1)- (5.3).

The gradient $\underline{\nabla} J$ of J is given by

$$\underline{\nabla} J = \frac{\partial \underline{\xi}_j^H \underline{\xi}_j}{\partial \underline{w}_j} = 2 \left[\frac{\partial \underline{\xi}_j^H}{\partial \underline{w}_j} \right] \underline{\xi}_j .$$

Since

$$\underline{\xi}_j = \underline{D}_j - x_j \underline{w}_j$$

Where $\underline{D}_j = [DFT\ \{\underline{d}_j^a\}]$ and $\underline{d}_j^a$ is the augmented desired response vector, it follows that

$$\nabla J = -2 x_j^* \underline{\xi}_j \quad . \tag{5.16}$$

A single element of this gradient is given by

$$\underline{\nabla} J(l) = -2 E_j(l) X_j^*(l), \quad l = 0, 1, 2, \ldots, N' - 1 \tag{5.17}$$

Using this gradient to adjust the frequency-domain weights once per data block yields the following weight adjustment formula used in [5.l] , [5.2] :

$$W_{j+1}(l) = W_j(l) - \beta F\{E_j(l)X_j^*(l)\} \qquad (5.18)$$

For $l = 0, 1, 2, \ldots, N' - 1$, where β is a convergence constant equivalent to $-2\mu_B / L$ in (A10) and $F\{.\}$ is the projection operator described in (5.11).

To summarize this section, the frequency-domain filter equations (5.15) and (5.18) are written in vector forma s follows:

$$\underline{y}_j = x_j \underline{w}_j \qquad (5.19)$$

$$\underline{w}_{j+1} = \underline{w}_j - \beta F\{x_j^* \underline{\xi}\} \qquad (5.20)$$

Note that techniques exist for improving the convergence speed of gradient algorithms by using a different step size for each weight at each iteration [5.7], [5.8], Such schemes can be applied to block adaptive filters in either the time domain or the frequency domain.

5.2.3 A General Structure for Frequency-Domain Adaptive Filters

Clearly Equations (5.12), (5.13) and (5.19), (5.20) are equivalent, and the same frequency-domain filter has been derived from two different approaches. In Section 5.2.1- time-domain mean-square error is minimized by a time-domain algorithm implemented in the frequency domain. In Section 5.2.1 -frequency-domain mean-square error is minimized by a frequency-domain algorithm. This equivalence is expected by Parseval's theorem

x_k

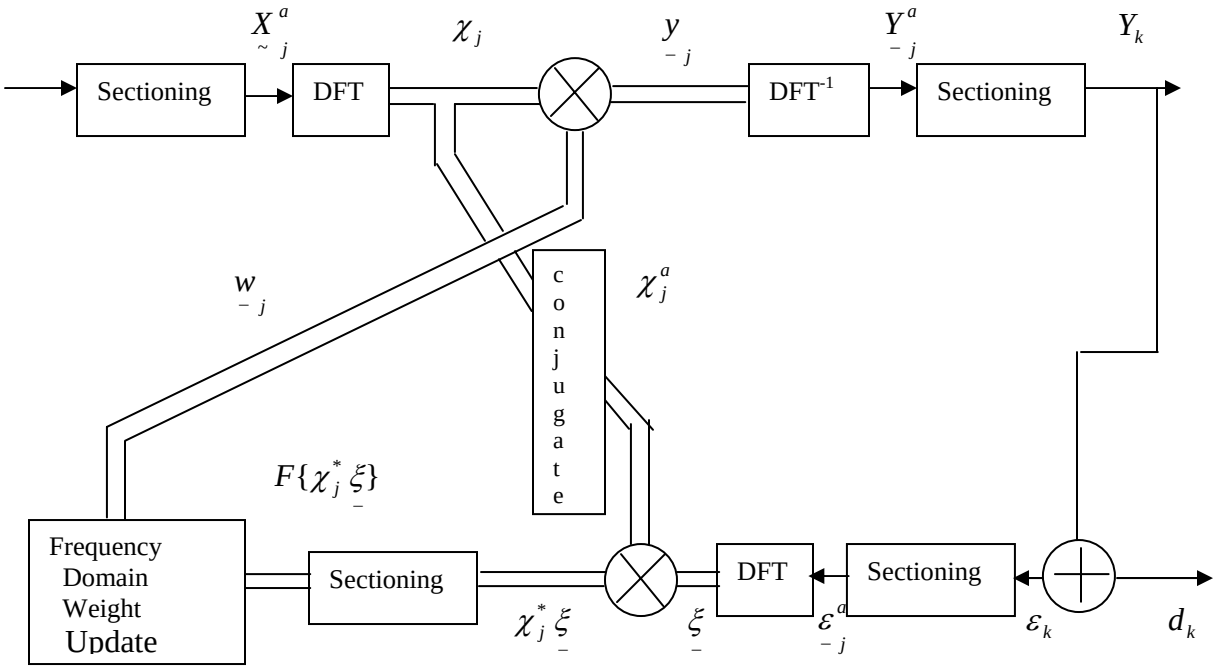


Fig.5.1 Generalized structure for all known FDAF

Fig. 5.1 shows the generalized structure for all known frequency domain adaptive filter and that they differ only in the boxes labeled "sectioning." This is shown in latter sections which describes in detail the overlap save and overlap-add sectioning procedures for adaptive filters,

5.3. Sectioning Procedure Applied to Frequency-Domain Adaptive Filters

Methods for doing overlap-save and overlap add sectioning with frequency-domain adaptive FIR filters are derived in this section.

5.3.1. Some Useful Definitions

In order to simplify analysis, the following projection operators for the general $N' \times 1$ vector $\underline{v}$ are defined, where K is an integer and $0 \le K \le N'$.

$$P_{K,0}\{\underline{v}\} = [v_0 \; v_1 \ldots v_{K-1} \; 0 \; 0 \ldots 0]^T \tag{5.21}$$

$$P_{K,}\{\underline{v}\} = [v_0 \; v_1 \ldots v_{K-1} \;]^T \tag{5.22}$$

$$P_K\{\underline{v}\} = [v_{N'-K} \; v_{N'-K+1} \ldots v_{N'-1} \;]^T$$

5.3.2. Example

In the following two sections, the overlap-save and overlap add implementations of the FDAF are demonstrated with a low-order $(L = N = 2, N' = 4)$ filter example. In all cases, it circular convolutions and correlations implemented using sectioning techniques are equal to the linear convolutions and correlations, which for the low order example are

$$\underline{y}_j = x_j \underline{w}_j = \begin{bmatrix} x_{2j} & x_{2j-1} \\ x_{2j+1} & x_{2j} \end{bmatrix} \begin{bmatrix} w_{0,j} \\ w_{1,j} \end{bmatrix} \tag{5.23}$$

$$\underline{\phi}_j = x_j{}^T \underline{\varepsilon}_j = \begin{bmatrix} x_{2j} & x_{2j+1} \\ x_{2j-1} & x_{2j} \end{bmatrix} \begin{bmatrix} \varepsilon_{2j} \\ \varepsilon_{2j+1} \end{bmatrix} \tag{5.24}$$

5.3.3. Overlap-Save Implementation

The convolution operation can be implemented with overlap-save sectioning by defining the following augmented vectors.

$$\underline{Y}_j^a = X_j^a \underline{W}_j^a = IDFT\{x_j \underline{w}_j\}. \tag{5.25}$$

Where,

$$\underline{Y}_j^a = [y_{jL-(N'-L)} \ldots y_{jL-2} \;\; y_{jL-1} \vdots y_{jL} \;\; y_{jL+1} \ldots y_{(j+1)L-1}]^T \tag{5.26}$$

$$\underline{W}_j^a = [w_{0,j} \; w_{1,j} \; \ldots \; w_{(N-1),j} \vdots 0 \;\; 0 \ldots 0]^T \tag{5.27}$$

$$x_j^a = [X_{jL}^a \quad X_{jL-1}^a \ldots X_{jL-N'+1}^a] \tag{5.28}$$

$$\underline{X}_{jL}^a = [x_{jL-(N'-L)} \ldots x_{jL-2} \quad x_{jL-1} \vdots x_{jL} \quad x_{jL+1} \ldots x_{(j+1)L-1}]^T \tag{5.29}$$

and

$$X_{jL-p}^a = CDS\{X_{jL-p+1}^a\} \tag{5.30}$$

Where $p = 0, 1, 2, \ldots, N' - 1$ and CDS means "circular down shift," so that successive columns of (5.28) are obtained by a vertical rotation of the previous column (shifting the elements of the previous column down one position and moving the bottom element to the top position). The first column is a CDS of the last column.

The matrix $x_j^a]$ is circulant [5.9] as defined by (5.28)-(5.30). To avoid overlap and obtain the desired linear convolution, the last $N' - N$ elements of $\underline{W}_j^a$ must be zero, and the first $N' - L$ elements of $\underline{Y}_j^a$ must be discarded as indicated in (5.26) and (5.27). Projection operator $P_{,L}$ indicates the discarding of points in (5.26).

$$\underline{Y}_j = P_{,L}\{\underline{Y}_j^a\} = P_{,L}\{IDFT[x_j \underline{w}j]\}. \tag{5.31}$$

Abutting the vectors end to end for $j = 0, 1, 2, \ldots$ now creates the final output sequence .

To illustrate the above convolution, consider the following low-order

$(L = N = 2, N' = 4)$ example for which (5.25) becomes

$$\begin{matrix} \textit{discard } N'-L \\ \textit{keep } L \end{matrix} \left\{ \begin{matrix} \\ \\ \\ \\ \end{matrix} \right. \left[\begin{matrix} x_{2j-2} & x_{2j+1} & x_{2j} & x_{2j-1} \\ x_{2j-1} & x_{2j-2} & x_{2j+1} & x_{2j} \\ x_{2j} & x_{2j-1} & x_{2j-2} & x_{2j+1} \\ x_{2j+1} & x_{2j} & x_{2j-1} & x_{2j-1} \end{matrix} \right] \left[\begin{matrix} w_{0,j} \\ w_{1,j} \\ 0 \\ 0 \end{matrix} \right] \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \begin{matrix} N \\ N'-N \end{matrix} \tag{5.32}$$

The results of (5.23) are obtained using (5.3 l).

$$\underline{Y}_j = \begin{bmatrix} y_{2j} \\ y_{2j+1} \end{bmatrix} = \begin{bmatrix} x_{2j} w_{0,j} + x_{2j-1} w_{1,j} \\ x_{2j+1} w_{0,j} + x_{2j} w_{1,j} \end{bmatrix} \tag{5.33}$$

The correlation operation is found by defining, in addition to the above quantities, the following augmented vectors.

$$\underline{\phi}_j^a = x_j^{aT} \underline{\varepsilon}_j^a = IDFT\{x_j^* \underline{\xi}_j\} \tag{5.34}$$

Where

$$\underline{\varepsilon}_j^a = [0 \ldots 0 \ \ 0 \vdots \varepsilon_{jL} \ \ \varepsilon_{jL+1} \ldots \varepsilon_{(j+1)L-1}]^T \tag{5.35}$$

$$\underline{\phi}_j^a = [\phi_{0,j} \ldots \phi_{1,j} \ \ \phi_{(N-1),j} \vdots \phi_{N,j} \ \ \phi_{(N+1)j} \ldots \phi_{(N'-1),j}]^T \tag{5.36}$$

The projection P_N indicates the discarding of points in (56).

$$\underline{\phi}_j = P_{N,}\{\underline{\phi}_j^a\} = P_{N,}\{IDFT[x_j^* \underline{\xi}_j]\} \tag{5.37}$$

Again the low order $(L = N = 2, N' = 4)$ example is used to describe (5.34)

$$\begin{matrix} keep \quad N \\ discard \ N'-N \end{matrix} \left\{ \begin{bmatrix} \phi_{0,j} \\ \phi_{1,j} \\ \phi_{2,j} \\ \phi_{3,j} \end{bmatrix} = \begin{bmatrix} x_{2j-2} & x_{2j-1} & x_{2j} & x_{2j+1} \\ x_{2j+1} & x_{2j-2} & x_{2j-1} & x_{2j} \\ x_{2j} & x_{2j+1} & x_{2j-2} & x_{2j-1} \\ x_{2j-1} & x_{2j} & x_{2j+1} & x_{2j-1} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \varepsilon_{2j} \\ \varepsilon_{2j+1} \end{bmatrix} \right\} \begin{matrix} N'-L \\ L \end{matrix} \tag{5.38}$$

Finally using (5.37) the results of (5.24) is obtained.

$$\underline{\phi}_j = \begin{bmatrix} \phi_{0,j} \\ \phi_{1,j} \end{bmatrix} = \begin{bmatrix} x_{2j}\varepsilon_{2j} + x_{2j+1}\varepsilon_{2j+1} \\ x_{2j-1}\varepsilon_{2j} + x_{2j}\varepsilon_{2j+1} \end{bmatrix} \tag{5.39}$$

Fig2. Shows the straightforward implementation of the above procedure in block form. From overlap sectioning the vector frequency domain weight adjustment algorithm is

$$\underline{w}_{j+1} = \underline{w}_j + \frac{2\mu_B}{L} F\{x_j^* \underline{\xi}_j\} \tag{5.40}$$

Operation of discarding all but the first N correlation points $(P_{N,0})$ shown in the realization of $F\{.\}$ in Fig. 2 is equivalent to multiplication in the time domain by a window function h_i where

$$h_i = \begin{cases} 1, & i = 0,1,\ldots,N-1 \\ 0, & N \le i \le N'-1 \end{cases} \tag{5.41}$$

By observing the multiplication of in the time domain is equivalent to the convolution with DFT in the frequency domain.

5.3.4. Overlap-Add Implementation

The convolution operation can be implemented with overlap-add sectioning by defining the following augmented vectors:

$$\underline{W}_j^a = [w_{0,j} \ \ w_{1,j} \ \ \ldots w_{(N-1),j} \vdots 0 \ \ 0 \ldots 0]^T \tag{5.42}$$

$$x_j^a = [X_{jL}^a \quad X_{jL-1}^a \ldots X_{jL-N'+1}^a] \tag{5.43}$$

Where,

$$X_{jL}^{a} = [\, x_{jL} \quad x_{jL+1} \ldots x_{(j+1)L-1} \vdots 0 \quad 0 \ldots 0]^{T} \tag{5.44}$$

and

$$X_{jL-p}^{a} = CDS\{X_{jL-p+1}^{a}\} \qquad \text{For} \quad p = 0, 1, 2, \ldots, N' - 1$$

and CDS means “circular down shift” .

To obtain the final outputs for block j, the partial results from block j must be added to the partial results obtained by convolving the inputs of block $j-1$ with the weights of block This is written as follows using the projection notation of (5.21)-(5.23) for the case in which $L = N$ and $N' = 2L$:

$$\begin{aligned} \underline{Y}_j &= P_{,L}\{X_j^{a} \underline{W}_j^{a}\} + P_N\{X_{j-1}^{a} \underline{W}_j^{a}\} \\ &= P_N\{IDFT[x_j \underline{w}_j]\} + P_N\{IDFT[x_{j-1} \underline{w}_j]\}. \end{aligned} \tag{5.45}$$

The general $L \neq N$, $N' \geq L + N - 1$ form of **(**5.45) is more difficult to write. As discussed in [5.9] **,** N' must be a power of two for use with fast DFT algorithms, and the most efficient block length is $L = N$ **.** Therefore, the set of parameters most likely to be used in practice is $L = N$ and $N' = 2L$.

The low-order $(L = N = 2, N' = 4)$ example for (5.45) gives

$$X_j^{a} \underline{W}_j^{a} = \begin{bmatrix} x_{2j} & 0 & 0 & x_{2j+1} \\ x_{2j+1} & x_{2j} & 0 & 0 \\ 0 & x_{2j+1} & x_{2j} & 0 \\ 0 & 0 & x_{2j+1} & x_{2j} \end{bmatrix} \cdot \begin{bmatrix} w_{0,j} \\ w_{1,j} \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \} N \\ \\ \} N' - N \end{matrix} \tag{5.46}$$

$$X_{j-1}^{a} \underline{W}_j^{a} = \begin{bmatrix} x_{2j-2} & 0 & 0 & x_{2j-1} \\ x_{2j-1} & x_{2j-2} & 0 & 0 \\ 0 & x_{2j-1} & x_{2j-2} & 0 \\ 0 & 0 & x_{2j-1} & x_{2j-2} \end{bmatrix} \cdot \begin{bmatrix} w_{0,j} \\ w_{1,j} \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \} N \\ \\ \} N' - N \end{matrix} \tag{5.47}$$

Assembling the results of (5.45) and (5.46) we get

$$\begin{aligned} \underline{Y}_j &= \begin{bmatrix} x_{2j} w_{0,j} \\ x_{2j+1} w_{0,j} + x_{2j} w_{1,j} \end{bmatrix} + \begin{bmatrix} x_{2j-1} w_{1,j} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} x_{2j} w_{0,j} + x_{2j-1} w_{1,j} \\ x_{2j+1} w_{0,j} + x_{2j} w_{1,j} \end{bmatrix} \end{aligned} \tag{5.48}$$

The correlation operation for overlap add sectioning is found similarly by defining, in addition to the above quantities, the following error vector:

$$\underline{\varepsilon}_j^a = [\varepsilon_{jL} \quad \varepsilon_{jL+1} \ldots \varepsilon_{(j+1)L-1} \vdots 0 \quad 0 \ldots 0]^T \tag{5.49}$$

The final correlations for block j are obtained by adding the partial results from block j to the partial results obtained by correlating the inputs of block $j-1$ with the errors of block j. This is written as follows using the projection notation of (5.21)-(5.23) for the case in which $L = N$ and $N' = 2L$.

$$\begin{aligned} \underline{Y}_j &= P_L\{X_j^{aT}\underline{\varepsilon}_j^a\} + P_N\{X_{j-1}^{aT}\underline{\varepsilon}_j^a\} \\ &= P_N\{IDFT[x_j^*\underline{\xi}_j]\} + P_N\{IDFT[x_{j-1}^*\underline{\xi}_j]\}. \end{aligned} \tag{5.50}$$

From (5.50) that for the overlap-add sectioning, the vector frequency-domain weight adjustment algorithm of (5.13) and (5.20) becomes

$$\underline{w}_{j+1} = \underline{w}_j + \frac{2\mu_B}{L} F\{x_j^*\underline{\xi}_j, x_{j-1}^*\underline{\xi}_j\} \tag{5.51}$$

The low-order example for (5.50) gives

$$X_j^{aT}\underline{\varepsilon}_j^a = \begin{bmatrix} x_{2j} & x_{2j+1} & 0 & 0 \\ 0 & x_{2j} & x_{2j+1} & 0 \\ 0 & 0 & x_{2j} & x_{2j+1} \\ x_{2j+1} & 0 & 0 & x_{2j} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{2j} \\ \varepsilon_{2j+1} \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \} N \\ \\ \} N'-L \\ \\ \end{matrix} \tag{5.52}$$

$$X_{j-1}^{aT}\underline{\varepsilon}_j^a = \begin{bmatrix} x_{2j-2} & x_{2j-1} & 0 & 0 \\ 0 & x_{2j-2} & x_{2j-1} & 0 \\ 0 & 0 & x_{2j-2} & x_{2j-1} \\ x_{2j-1} & 0 & 0 & x_{2j-2} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{2j} \\ \varepsilon_{2j+1} \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \} N \\ \\ \} N'-L \\ \\ \end{matrix} \tag{5.53}$$

Assembling the results of (5.52) and (5.53) and according to (5.50) it becomes

$$\begin{aligned} \underline{\phi}_j &= \begin{bmatrix} x_{2j}\varepsilon_{2j} + x_{2j+1}\varepsilon_{2j+1} \\ x_{2j}\varepsilon_{2j+1} \end{bmatrix} + \begin{bmatrix} 0 \\ x_{2j-1}\varepsilon_{2j} \end{bmatrix} \\ &= \begin{bmatrix} x_{2j}\varepsilon_{2j} + x_{2j+1}\varepsilon_{2j+1} \\ x_{2j-1}\varepsilon_{2j} + x_{2j}\varepsilon_{2j+1} \end{bmatrix} \end{aligned} \tag{5.54}$$

For overlap-add sectioning. Notice that seven DFT's are required, whereas five DFT's are required for overlap-save sectioning**.** This is because of the necessity to add the results of convolving the current weights $\underline{W}_j$ with the previous input block and correlating the current

error block with the previous input block. If the weights were not changing once per block (as in a time-invariant filter), the quantities $P_{,N}\{X^{a}_{j-1}\underline{W}^{a}_{j}\}$ and $P_{,N}\{X^{aT}_{j-1}\underline{\varepsilon}^{a}_{j}\}$ could be stored and used for the computation without requiring the two additional DFT's. This indicates that a price of two DFT's is paid for the luxury of changing weights midstream when using the overlap-add sectioning procedure. The overlap-save procedure automatically includes inputs from the previous block in the augmented input vector X^{a}_{jL}, eliminating the above problem.

5.4 Computational Complexity Of LMS And BLMS Adaptive Filtering

The main computational efficiency issues involved in algorithm implementation are storage (memory), time (number of machine cycles for CPU, input-output, etc.), and computational complexity measured in the number of real multiply and additions required. Here I have given the comparison of the computation complexity of LMS and BLMS in the frequency domain I have given.

5.4.1 Computational Complexity of LMS Adaptive Filters

The convolution operation (1) is done in direct form. To produce one output point requires N real multiplies and $N-1$ real adds. Thus to produce L output points requires LN real multiplies and $L(N-1)$ real adds.

To produce L output points (one block) using the LMS algorithm requires LN adaptions. The term $(2\mu\varepsilon_k)X_k$ requires $L(N+1)$ real multiplies per block. The addition operation requires LN real additions per block. The cost of computing $\varepsilon_k = d_k - y_k$, is L real adds per block. The total cost per block for the LMS algorithm is $L(N+1)$ real multiplies and $L(N+1)$ real additions. Thus, the total computational complexity of LMS adaptive filtering is $L(2N+1)$ real multiplies and $2LN$ real adds. This result is shown in Table [5.1]

5.4.2 Computational Complexity of FFT-Implemented Convolution

The FFT algorithm used is for complex data. The relationship between complex and real arithmetic operations is

1 complex add = 2 real adds

1 complex multiply = 4 real multiplies + 2 real adds.

A linear convolution of two sequences of length L and N produces a sequence of length $L+N-1$. Thus an FFT implementation of such a convolution must use sequences of length $N' \geq L+N-1$ to avoid overlapping sections.

The radix 4 FFT complexity formulas are useful only when the number of butterfly sections $(\log_2(N'/2))$ is even. For $(\log_2(N'/2))$ odd, an efficient radix 2 section is used for the first section, and $(\log_4(N'/4))$ radix 4 sections are used for the remainder. Sorting out the results of the efficient procedure for real data requires $2N'$ real multiplies and $5N'$ real adds. The FFT complexity per block of outputs is then $(3/4)N'Q - 4(N'/2-1) + 2N' + bN'$ real multiplies and $(11/8)N'Q - 2(N'/2-1) + 5N' + b(3N'/2)$ real adds, where

$$Q = \begin{cases} \log_2(N'/2). & \log_2(N'/2)\ even \\ \log_2(N'/4), & \log_2(N'/2)\ odd \end{cases}$$

and

$$b = \begin{cases} 0. & \log_2(N'/2)\ even \\ 1, & \log_2(N'/2)\ odd \end{cases}$$

A convolution requires one FFT each for the weights and the input. It also requires one complex multiply and one inverse FFT, leading to a total of $(9/4)N'Q - 12(N'/2-1) + 8N' + b(3N')$ real multiplies and $(33/8)N'Q - 6(N'/2-1) + 16N' + b(9N'/2)$ real adds per block.

The correlation required for the gradient estimate is similar to the convolution above but requires one less FFT because the transform of the inputs exists previously **(see Fig. 8).** This leads to a correlation complexity of $(3/2)N'Q - 8(N'/2-1) + 6N' + b(2N')$ real multiplies and $(11/4)N'Q - 4(N'/2-1) + 11N' + b(3N')$ real adds per block.

Table 5.1

(Computational Complexity (Number of Real ADD's and Real Multiplies) Required to Compute L output data points, $N' \geq L + N - 1$)

Implementation	Real multiplies	Real adds
LMS adaptive filter	$L(2N+1)$	$2LN$
BLMS adaptive filter	$N(2L+1)$	$2LN$
FFT-implemented,BLMS adaptive filter	$(15/4)N'Q - 20(N'/2-1)$ $+14N' + b(5N')$	$(55/8)N'Q - 10(N'/2-1) + 27N'$ $+ N + L + b(15N'/2)$

with radix 4 FFT and efficient algorithm for Real Adds	$Q = \begin{cases} \log_2(N'/2). & \log_2(N'/2)\ even \\ \log_2(N'/4), & \log_2(N'/2)\ odd \end{cases}$	$b = \begin{cases} 0. & \log_2(N'/2)\ even \\ 1, & \log_2(N'/2)\ odd \end{cases}$

5.4.3 FFT implementation of BLMS Adaptive Filters

With BLMS adaptive filtering, the convolution operation can be implemented using the FFT and an overlap-add or overlap-save procedure. Throughout this analysis, input and output signals are assumed to be real and the FFT is used for complex data. This permits use of the efficient convolution procedure in which the transform of an N'-point real sequence is computed by properly using the real and imaginary parts of an $N'/2$-point complex FFT algorithm. To make further complexity reductions, a radix 4 FFT with one radix 2 sections is used. The FFT length $N'/2$ must be a power of two where $N' \geq L + N - 1$. The complexity of the FFT algorithm and the convolution are discussed in the previous section.

The gradient estimate term ϕ_j in the BLMS algorithm can be written in the form of a correlation. Therefore, it can be realized with the FFT by the same technique used for the convolution above. The only difference is that the FFT of ε_i must be conjugated before it is multiplied by the FFT of x_i. Its complexity is discussed before. Note that the correlation operation produces N' points, but that only the first N of them have meaning, because there are only N weights to adjust. The remaining $N' - N$ points are discarded (set to zero).

Once the correlation (gradient estimate) is computed, the BLMS algorithm requires N real multiplies and $N + L$ real adds per block of outputs. Table 5.1 shows the total complexity of the FFT-implemented BLMS adaptive filter.

5.5 Simulation and Results

Extensive computer simulations were carried out using the two structures using LMS algorithm. For both System Identification and Channel Equalization problem, a uniformly distributed random signal over the interval [-.5,.5] was applied to the FIR structure and a white Gaussian noise of 30dB was added to the output of the system. The learning parameter μ both for LMS and BLMS algorithm was suitably chosen to obtain best result.

Four different channels were studied with the following transfer function:

$CH = 1 : 0.209 + 0.995z^{-1} + 0.209z^{-2}$

$CH = 2 : 0.260 + 0.930z^{-1} + 0.260z^{-2}$

$CH = 3 : 0.340 + 0.903z^{-1} + 0.304z^{-2}$

$CH = 4 : 0.341 + 0.876z^{-1} + 0.341z^{-2}$

To study the effect of nonlinearity on the system performance four different nonlinear channel models with the following nonlinearities has been introduced.

$NL = 0 : b(k) = a(k)$

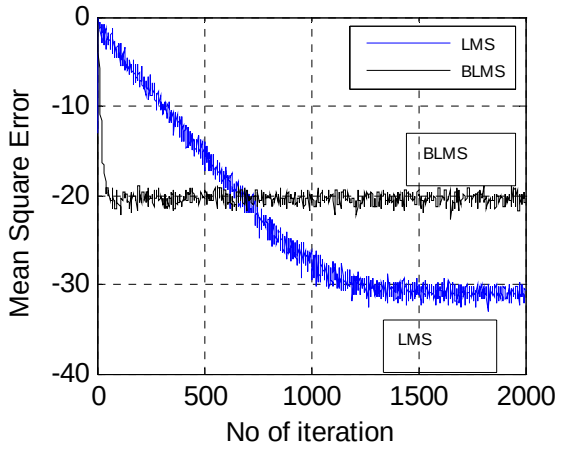
$NL = 1 : b(k) = \tanh(a(k))$

$NL = 2 : b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k)$

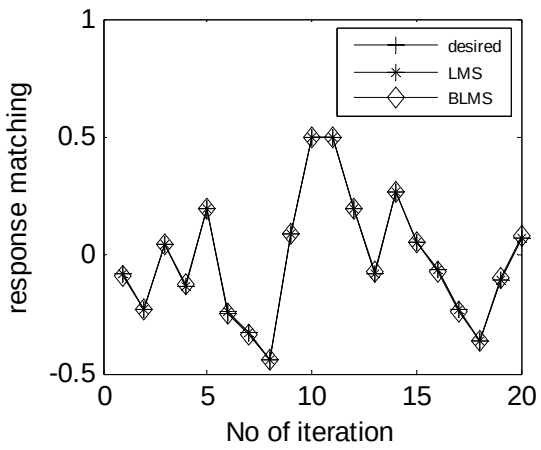
$NL = 3 : b(k) = a(k) - 0.9(a^3(k))$

The comparison of the LMS and BLMS algorithm in frequency domain for the convergence characteristics and response-matching plot for System Identification and BER plot for all the linear and nonlinear channel model has been given.

Simulation result for channel 2 with 30 dB noises has been simulated for different linear and non-linear channel has been studied.



5.2 (a)

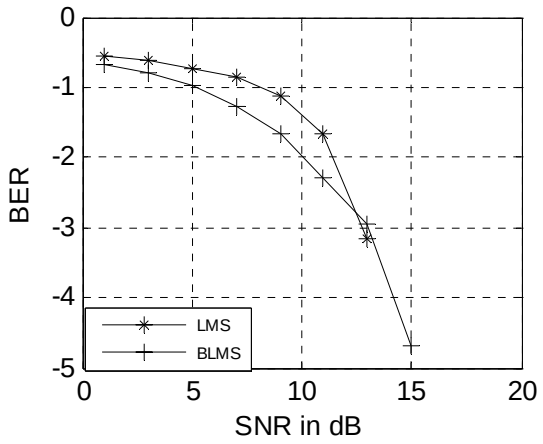


5.2 (b)

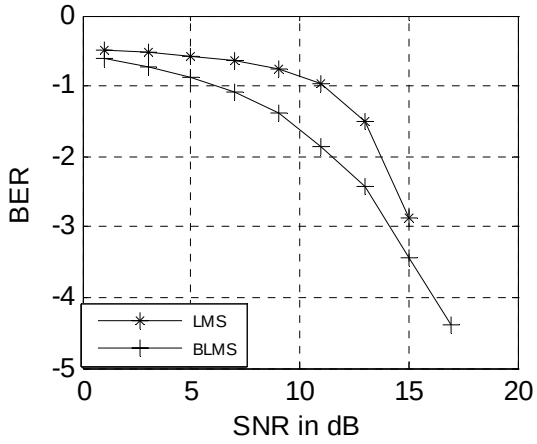
Fig 5.(a) ,(b) are the corresponding MSE and Response matching plot for System Identification problem for LMS and BLMS algorithm without noise condition

From MSE plot it is seen BAF in frequency domain converges much faster than the corresponding LMS adaptive filter. It takes around 100 samples where as LMS takes 500 samples to converge to the resultant value keeping the matching performance constant.

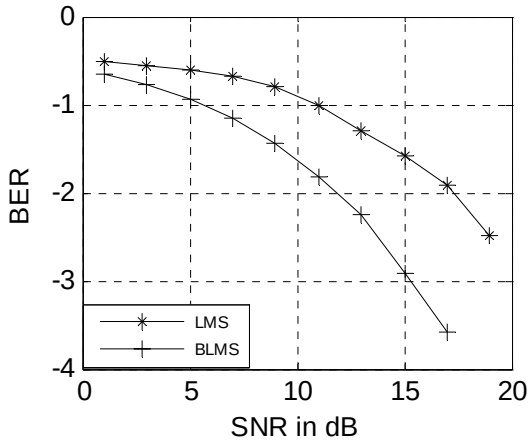
From computer simulation for the Channel equalization case it is seen that BLMS converges much faster and better accurately to the equalizer coefficients both in linear and nonlinear cases with having the advantage of less computation complexity.



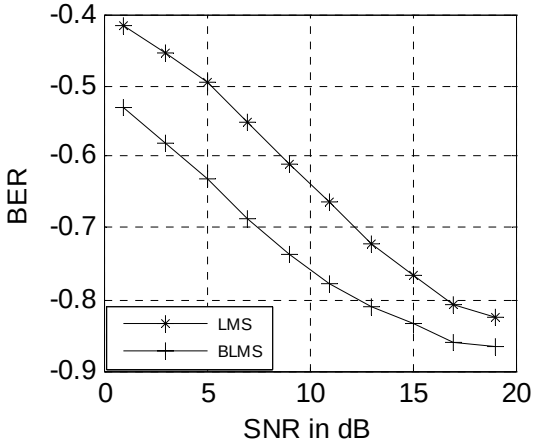
5.3(a)



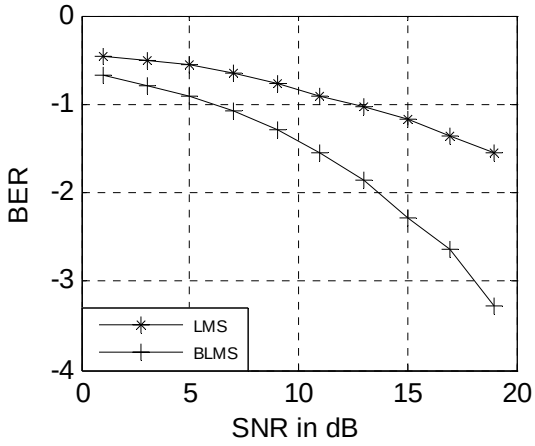
5.3(b)



5 .3(c)



5.3 (d)



5.3 (e)

Fig.5. (a),(b),(c),(d),(e) ,corresponds to the respective BER plot for LMS and BLMS

Equalizer structure with NL=0,NL=1,NL=2,NL=3 and NL=4

5.6 Conclusion

Specific implementation of the block adaptive filter in the frequency domain has been presented in this chapter. From extensive computer simulation it is seen that whether derived in time domain or in frequency domain frequency domain adaptive filter results the same if the data sectioning is done correctly. It has been seen that though results for the overlap save and overlap add sectioning is same overlap add method requires more mathematical computation so overlap save sectioning is preferred. Through computer simulation is seen that BLMS in frequency domain requires less computation complexity and converge faster than LMS AF in system Identification case and a much better candidate in comparison in channel equalization case seen from the BER plots.

Chapter 6

EQUALIZATION AND IDENTIFICATION USING ANN

6.1 Introduction

Recently, artificial neural networks (ANN) have emerged as a powerful learning technique to perform complex tasks in highly nonlinear dynamic environments. Some of the prime advantages of using ANN models are: their ability to learn based on optimization of an appropriate error function and their excellent performance for approximation of nonlinear function [6.1]-[6.2]. The ANN's are capable of generating complex mapping between the input and output and thus, arbitrarily complex decision boundaries can be formed by these networks.

The functional link ANN (FLANN) has been proposed by Pao [6.3]-[6.4]. It is shown that this network can be used for function approximation and pattern classification with faster convergence rate and lesser computational complexity than a MLP network. The performance of the FLANN for the task of identification of nonlinear systems has been reported [6.5]. Using trigonometric functions as functional expansion, superior performance of the FLANN with respect to MLP network has been obtained. In this paper, we propose an alternate FLANN structure, which has been shown to provide effective identification of nonlinear dynamic systems. For functional expansion of the input pattern, we have chosen the Chebyschev polynomials [6.6] instead of trigonometric and the network is updated with recursive least mean square algorithm. The input noise is also considered during the identification of the nonlinear systems and it is pointed out that this network has universal approximation capability and has faster than a MLP and FLANN network of trigonometric expansion.

6.2 Single Neuron Structure

In 1958, Rosenblatt demonstrated some practical applications using the perceptron .The perceptron is a single level connection of McCulloch-Pitts neurons sometimes called single-layer feed forward network*s*. The network is capable of linearly separating the input vectors into pattern of classes by a hyper plane. A linear associative memory is an example of a single-layer neural network. In such an application, the network associates an output pattern (vector) with an input pattern (vector), and information is stored in the network by virtue of modifications made to the synaptic weights of the network.

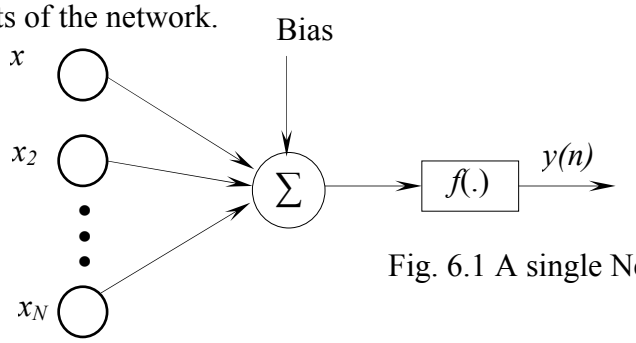


Fig. 6.1 A single Neuron

The structure of a single neuron is presented in Fig. 6.1. In an artificial neuron involves the computation of the weighted sum of inputs and threshold. The resultant signal is then passed through a non-linear activation function. The output of the neuron may be represented as,

$$y(n) = f\left[\sum_{j=1}^{N} w_j(n)\, x_j(n) + bias\right] \tag{6.1}$$

Where *bias* = threshold to the neurons at the first layer,

$w_j(n)$ = weight associated with the j^{th} input, and N = no. of inputs to the neuron.

6.2.1 Activation Function

The perceptron internal sum of the inputs is passed through an activation function, which can be any monotonic function. Linear functions can be used but these will not contribute to a non-linear transformation within a layered structure, which defeats the purpose of using a neural filter implementation. A function that squashes the amplitude range and limits the output strength of each perceptron of a layered network to a defined range in a non-linear manner will contribute to a nonlinear transformation. There are many forms of activation functions, which are selected according to the specific problem. All the neural network architectures employ the activation function ,which defines as the output of a neuron in terms of the activity level at its input (ranges from -1 to 1 or 0 to 1). Table 6.1 summarizes the basic types of activation functions. The most practical activation functions are the sigmoid and the hyperbolic tangent functions. This is because they are differentiable.

Table 6.1

(Common Activation Functions)

Name	Definition
Linear	$f(x) = kx$
Step	$f(x) = \beta, \quad if\ x \geq k$ $= \delta, \quad if\ x < k$
Sigmoid	$f(x) = \dfrac{1}{1+e^{-\alpha x}}, \alpha > 0$
Hyperbolic Tangent	$f(x) = \tanh(\gamma x) = \dfrac{1-e^{-\gamma x}}{1+e^{-\gamma x}}, \gamma > 0$
Gaussian	$f(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\dfrac{(x-\mu)^2}{2\sigma^2}\right]$

6.2.2 Learning Processes

The property that is of primary significance for a neural network is that the ability of the network to learn from its environment, and to improve its performance through learning. The improvement in performance takes place over time in accordance with some prescribed measure. A neural network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels. Ideally, the network becomes more knowledgeable about its environment after each iteration pf learning process. Hence we define learning as: "*Learning is a process by which the free environment of a neural network is adapted through a process of stimulation by the environment in which the network is embedded*."

The processes used are classified into two categories as described in [6.1]:

1. Supervised Learning (Learning With a Teacher)
2. Unsupervised Learning (Learning Without a Teacher)

6.2.2.1 Supervised Learning:

We may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment is, however unknown to neural network of interest. Suppose now the that the teacher and the neural network are both exposed to a training vector , by virtue of built-in knowledge , the teacher is able to provide the neural network with a desired response for that training vector. Hence the desired response represents the optimum action to be performed by the neural network. The network parameters such as the weights and the thresholds are chosen arbitrarily and are updated during the training procedure to minimize the difference between the desired and the estimated signal. This updation is carried out iteratively in a step-by-step procedure with the aim of eventually making the neural network emulate the teacher. In this way knowledge of the environment available to the teacher is transferred to the neural network. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself. This is the form of supervised learning.

6.2.2.1unsupervised Learning:

In unsupervised learning or self-supervised learning there is no teacher to over-see the learning process, rather provision is made for a task independent measure of the quantity of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become turned to the statistical regularities of the input data, it develops the ability to form the internal representations for

encoding features of the input and thereby to create new classes automatically. In this learning the weights and biases are updated in response to network input only. There are no desired outputs available. Most of these algorithms perform some kind of clustering operation. They learn to categorize the input patterns into some classes.

6.3 Multilayer Perceptron

In 1969 Minsky and Papert mathematically demonstrated that there were fundamental limits on what a single layer perceptron could compute and he suggested the concept of the multilayer perceptron or Multilayer perceptron (MLP) feed forward networks.

The MLP is one of several non-linear architectures that use layers of processing functions to map signals onto a series of planes so that they can be grouped into disconnected and non-linear classes. Any linearly non-separable pattern problem can be solved if a mapping transformation of sufficient dimension is used.The Multilayer Perceptron (MLP) or Multilayer Artificial Neural Network (MLANN) introduce one more hidden layers, whose computation nodes are correspondingly, called hidden neurons. The function of the hidden neurons is to intervene between the external input and the network output. The activation functions are listed in Table 6.1. The scheme of multi-layer neural network (MLANN) using three layers structure is shown in Fig. 6.2.

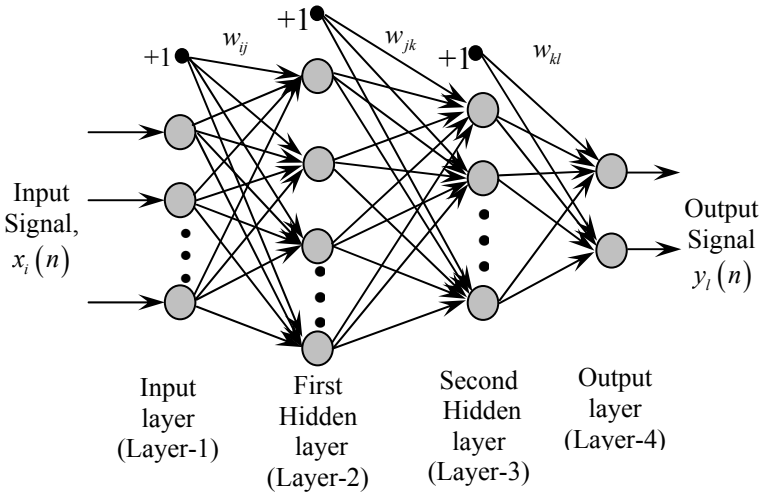


Fig. 6.2 Structure of multilayer perceptron

$x_i(n)$ represent the input to the network, f_j and f_k represent the output of the 1st hidden and 2nd hidden layers respectively and $y_l(n)$ represents the output of the neural network. The connecting weights between the input to the first layer, first to second layer and the second layer to the output layers are represented by w_i, w_{jk} and w_{kl} respectively. If P_1 is the number of neurons in the first layer, each element of the output vector may be calculated as,

$$f_j = \varphi_j \left[\sum_{i=1}^{N} w_{ij} x_i(n) + b_j \right] \tag{6.2}$$

$$j=1, 2, 3 \ldots P_1$$

Where b_j is the threshold to the neurons at the first layer, N is the no. of inputs and $\varphi(.)$ is the non-linear activation function. The time index n has been dropped to make the equations simpler. Let P_2 be the number of neurons in the second layer. Each element of this output vector, f_k may be written as:

$$f_k = \varphi_k \left[\sum_{j=1}^{P_1} w_{jk} f_j + b_k \right] \tag{6.3}$$

$$k=1, 2, 3 \ldots P_2$$

Where, b_k is the threshold to the neurons at the second layer. The input signal is passed through a tapped delay filter.

The output of the final layer can be calculated as:

$$y_l(n) = \varphi_l \left[\sum_{k=1}^{P_2} w_{kl} f_k + b_l \right] \tag{6.4}$$

$$l=1, 2, 3 \ldots P_3$$

Where, b_l is the threshold to the neuron at the final layer and P_3 is the no. of neurons in the output layer. The activation functions are described in Table 6.1. The most popular and successful learning method for training the multilayer perceptron is the back propagation algorithm. Rumelhart Hinton and Williams reported the development of the back propagation learning in 1986 [6.1]. The algorithm employs an iterative gradient-descent method of minimization which minimizes the mean squared error (L^2 norm) between the desired output and network output (supervised learning). Using the Back Propagation (BP) Neural Algorithm, the parameters of the neural network are updated in a batching mode. The final output is compared with the desired output and the resulting error signal is obtained. This error signal is used to

update the weights and thresholds of the hidden layers as well as the output layer. The weights and the thresholds are updated in an iterative method until the difference between the desired and the estimated output becomes minimum. For measuring the degree of matching, the Mean Square Error (MSE) is taken as a performance measurement.

The updated weights are,

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n) \tag{6.5}$$

$$w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}(n) \tag{6.6}$$

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \tag{6.7}$$

Where, $\Delta w_{kl}(n), \Delta w_{jk}(n)$ and $\Delta w_{ij}(n)$ are the changes in weights of the output, hidden and input layer respectively.

$$\begin{aligned} \Delta w_{kl}(n) &= -2\mu \frac{d\xi(n)}{dw_{kl}(n)} = 2\mu e(n) \frac{dy_l(n)}{dw_{kl}(n)} \\ &= 2\mu e(n) \varphi_l' \left[\sum_{k=1}^{P_2} w_{kl} f_k + b_l \right] w_{kl} \end{aligned} \tag{6.8}$$

Where, μ is the convergence coefficient ($0 \le \mu \le 1$). Similarly the $\Delta w_{jk}(n)$ and $\Delta w_{ij}(n)$ can be computed.

The back propagation algorithm provides an "approximation" to the trajectory in the weight space computed by the method of steepest gradient rule. This algorithm has some drawbacks. The smaller we make the learning parameter μ, the smaller the change in synaptic weights, and the smoother will be the trajectory in weight space. The improvement, however, is attained at the cost of a slower rate of learning. If, we make the learning rate much higher in order to speed the rate of learning, the resulting large changes in the synaptic weights assume such a form that the network may become unstable (i.e., oscillatory). Another demerit of this algorithm is local minima. While the performance surface for a single layer has a single minimum point, and constant curvature, the performance surface for a multilayer network may have many local minimum points and the curvature can vary widely in different regions of the parameter space. For this reason it is difficult to choose an appropriate learning rate for the steepest descent rule. A simple method of increasing the rate of learning yet avoiding the danger of instability is to modify the weight update rule by including a momentum rule as follows:

$$
\begin{aligned}
w_{kl}(n+1) &= w_{kl}(n) + \Delta w_{kl}(n) + \alpha\, \Delta w_{kl}(n-1) \\
w_{jk}(n+1) &= w_{jk}(n) + \Delta w_{jk}(n) + \alpha\, \Delta w_{jk}(n-1) \\
w_{ij}(n+1) &= w_{ij}(n) + \Delta w_{ij}(n) + \alpha\, \Delta w_{ij}(n-1)
\end{aligned}
\tag{6.9}
$$

Where α is usually a positive number called the momentum constant having range between 0 and 1.

6.4 The FLANN

The FLANN, initially proposed by Pao is a single layer ANN structure capable of forming complex decision regions by generating nonlinear decision boundaries .It consists of a functional expansion block and a single layer perceptron network. The main purpose of the functional expansion block is to increase the dimension of the input pattern so as to enhance its representation in a high-dimensional space. This enhanced space is then used for the system identification problem. For this paper, we consider an m-dimensional input pattern at the kth

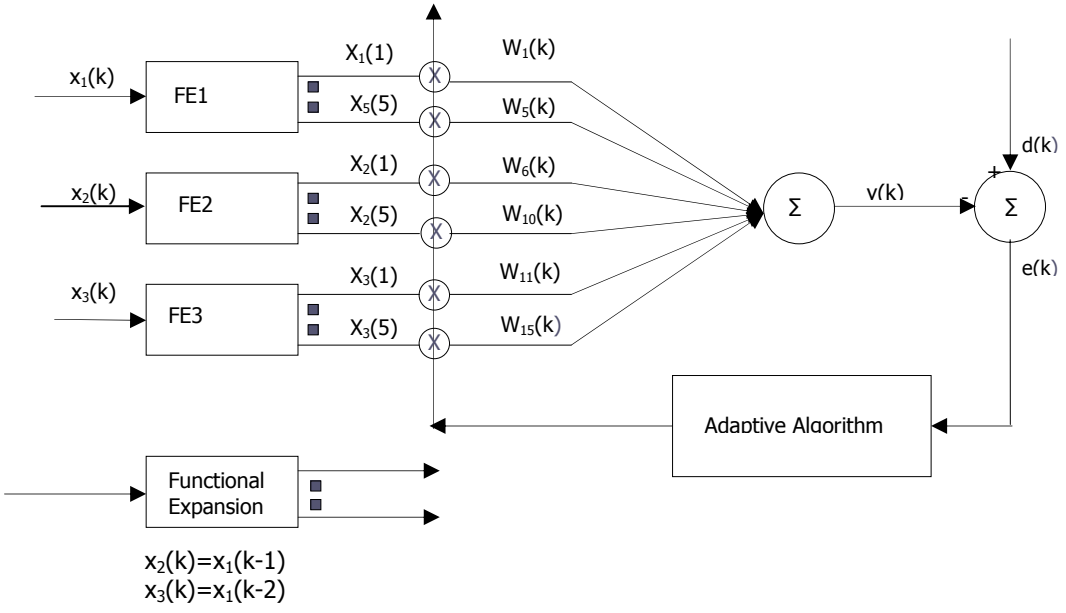


Fig.6.3. Block diagram for the FLANN system identification model

Instance given by T (k) = {T1 T2 ... Tm}. Each element in the vector is T(k) expanded into several terms, which is obtained by using trigonometric functions and the first few terms are given as :

$T_0(k) = 1$
$T_1(k) = \sin(\pi x)$
$T_2(k) = \cos(\pi x)$
$T_3(k) = \sin(2\pi x)$
$T_4(k) = \cos(2\pi x)$

This network is then used for system identification problem .The FLANN network is trained using the generalized delta-learning algorithm.

6.5 The Chebyschev Neural Network (CFLANN)

The Chebyschev polynomials are a set of orthogonal Polynomials defined as the solution to the Chebyschev differential equation and denoted as $T_n(x)$ .It is similar to FLANN except the difference is that the input vector into the functional block is now expanded with Chebyschev polynomials. The higher order chebyschev polynomials for –1<x<1 can be generated using the recursive formula given by

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{6.10}$$

The first few chebyschev polynomials are given as:

$T_0(x) = 1$
$T_1(x) = x$
$T_2(x) = 2x^2 - 1$
$T_3(x) = 4x^3 - 3x$
$T_4(x) = 8x^4 - 8x^2 + 1$
$T_5(x) = 16x^5 - 20x^3 + 5x$

The advantage of CHNN over FLANN is that the Chebyschev polynomials are computationally more efficient than using trigonometric polynomials to expand the input space.

6.6 Recursive Least Square (RLS) Algorithms

The algorithms that result from the gradient descent methods has the disadvantages that they can be slow to approach the optimal weight vector and, once close to it, usually" rattle around" the optimal; vector rather than actually converge to it, due to the effects of approximations made in the estimate of the performance function gradient. To overcome these difficulties, another

approach is discussed in this section. Here we develop algorithms that use the input data *{x,d}* in such a way as to ensure optimality at each step. If we can be done, then clearly the result of the algorithm for the last data point is the overall optimal weight vector.

Suppose that we refine the sum squared performance function J_{ss} by the expression

$$J_k = \sum_{l=N-1}^{k-1} \left| y(l) - d(l) \right|^2 , \; N-1 \le k \le L-1 \tag{6.11}$$

This form of J simply reflects how much data have been used so far. Clearly, J_L uses all the available data from *k=0* to *k=L-1*. Suppose we define W_k^o as the impulse response vector that minimizes J_k .By this definition, W_{L-1}^o equals W_{ss}^o, and the optimal impulse vector over all the data.

The motivation for developing"recursive-in-time" algorithms can be seen as follows. Suppose *x(l)* and *d(l)* have been received for time up through k-1 and that W_k^o has been computed. Now suppose that *x(k)*and *d(k)*are received, allowing us to form

$$J_{k+1} = \sum_{l=N-1}^{k} \left| y(l) - d(l) \right|^2 \stackrel{\Delta}{=} J_k + \left| y(k) - d(k) \right|^2 \tag{6.12}$$

Wee desire to find some procedure by which W_k^o can be updated to produce W_{k+1}^o, the new optimal vector. If we can develop such a procedure, then we can build up the optimal weight vector step by step until the final pair of data points *x(L-1)* are received. With these points, W_{L-1}^o can be computed, which, by definition, is the global optimum vector W_{ss}^o.

The update formula:

The simplest approach to updating W_k^o is the following procedure:

(a) Update R_{ss} via	$R_{ss,k+1} = R_{ss,k} + X(k)X^t(k)$
(b) Update P_{ss} via	$P_{ss,k+1} = P_{ss,k} + d(k)X(k)$
(c) Invert	$R_{ss,k+1}$
(d)Compute W_{k+1}^o via	$W_{k+1}^o = R_{ss,k+1}^{-1} P_{ss,k+1}$

The autocorrelation matrix and cross correlation vectors are updated and then used to compute W_{k+1}^{o}. While direct, this technique is computationally wasteful. Approximately N^3+2N^2+N multiplications is required at each update, where N is the impulse response length, and have that N^3 are required for the matrix inversion if done with the classical Gaussian elimination technique.

In an effort to reduce the computational requirement for this algorithm, we focus first on this inversion. We notice that Gaussian elimination makes no use whatsoever have the special form of $R_{ss,k}$ or of the special form of the update from $R_{ss,k}$ to $R_{ss,k+1}$. We now set out to take advantage of it. We do so by employing the well-known matrix inversion lemma, also sometimes called the *ABCD* lemma,

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1} \tag{6.13}$$

We use this lemma by making the following associations:

$$\begin{aligned} A &= R_k \\ B &= X(k) \\ C &= 1 \\ D &= X^t(k) \end{aligned} \tag{6.14}$$

With these associations, R_{k+1} can be represented as

$$R_{k+1} = R_k + X(k)X^t(k) = A + BCD \tag{6.15}$$

and R_{k+1}^{-1} is given by

$$R_{k+1}^{-1} = R_k^{-1} - \frac{R_k^{-1}X(k)X^t(k)R_k^{-1}}{1 + X^t(k)R_k^{-1}X(k)} \tag{6.16}$$

Thus, given R_k^{-1} and a new input *x(k),* hence *X(k)* **,** we can compute R_{k+1}^{-1} directly. We never compute R_{k+1},nor do we invert it directly.

The optimal weight vector W_{k+1}^{o} is given by

$$W_{k+1}^{o} = R_{k+1}^{-1}P_{K+1} \tag{6.17}$$

This can be obtained by combining (3.43) with update *Pss*

$$W_{k+1}^{o} = \left\{ R_k^{-1} - \frac{R_k^{-1} X(k) X^t(k) R_k^{-1}}{1 + X^t(k) R_k^{-1} X(k)} \right\} . \{ P_k + d(k) X(k) \}$$

$$= R_k^{-1} P_k - \frac{R_k^{-1} X(k) X^t(k) R_k^{-1} P_k}{1 + X^t(k) R_k^{-1} X(k)} + d(k) R_k^{-1} X(k) - \frac{d(k) . R_k^{-1} X(k) X^t(k) R_k^{-1} X(k)}{1 + X^t(k) R_k^{-1} X(k)} \qquad (6.18)$$

To simplify this result, we make the following associations and definitions. The k[th] optimal weight vector:

$$R_k^{-1} P_k = W_k^o \qquad (6.19)$$

The filtered information vector:

$$Z_k \overset{\Delta}{=} R_k^{-1} X(k) \qquad (6.20)$$

The priori output:

$$y_o(k) \overset{\Delta}{=} X^t(k) W_k^o \qquad (6.21)$$

The normalized input power:

$$q = X^t(k) Z_k = X^t(k) R_k^{-1} X(k) \qquad (6.22)$$

With these expressions, the optimal weight vector W_{k+1}^o becomes

$$W_{k+1}^o = W_k^o - \frac{Z_k X^t(k) W_k^o}{1 + X^t(k) Z_k} + d(k) Z_k - \frac{d(k) Z_k X^t(k) Z_k}{1 + X^t(k) Z_k}$$

$$= W_k^o - \frac{Z_k y_o(k)}{1+q} + d(k) Z(k) - \frac{d(k) q Z_k}{1+q}$$

$$= W_k^o - \frac{Z_k y_o(k)}{1+q} + \frac{d(k) Z_k}{1+q} \qquad (6.23)$$

$$= W_k^o + \frac{\{ d(k) - y_o(k) \} . Z_k}{1+q}$$

Equations (6.16) and (6.19)-(6.23) comprise the *recursive least squares (RLS)* algorithm.

Steps for RLS Algorithm:

The step-by-step procedures for updating W_k^o are given in this section. This set of steps is efficient in the sense that no unneeded variable is computed and that no needed variable is

computed twice. We do, however, need assurance that R_k^{-1} exists. The procedure then goes as follows:

(i) Accept new samples *x(k), d(k).*

(ii) Form *X(k)* by shifting *x(k)* into the information vector.

(iii) Compute the *a priori* output $y_o(k)$:

$$y_o(k) = W_k^{ot} x(k) \tag{6.24}$$

(iv) Compute *a priori* error $e_o(k)$:

$$e_o(k) = d(k) - y_o(k) \tag{6.25}$$

(v) Compute the filtered information vector Z_k :

$$Z_k = R_k^{-1} X(k) \tag{6.26}$$

(vi) Compute the normalized error power q:

$$q = X^t(k) Z_k \tag{6.27}$$

(vii) Compute the gain constant *v:*

$$v = \frac{1}{1+q} \tag{6.28}$$

(viii) Compute the normalized filtered information vector $\widetilde{Z}_k$:

$$\widetilde{Z}_k = v.Z_k \tag{6.29}$$

(ix) Update the optimal weight vector W_k^o to W_{k+1}^o **:**

$$W_{k+1}^o = W_k^o + e_o(k)\widetilde{Z}_k \tag{6.30}$$

(x) Update the inversion correlation matrix R_k^{-1} to R_{k+1}^{-1} in preparation for the next iteration:

$$R_{k+1}^{-1} = R_k^{-1} - \widetilde{Z}_k \widetilde{Z}_k^t \tag{6.31}$$

This procedure assumes that R_k^{-1} exists at the initial time in the recursion. As a result, two initialization procedures are commonly used. The first is to build up R_k and Pk until R has full

rank, i.e. at least *N* input vectors X(k) are acquired. At this point R_k^{-1} is computed directly and then W_k. Given these, the recursion can proceed as described above indefinitely or until *k=L-1*. The advantage of the first technique is that optimality is preserved at each step. The major price paid is that is about N^3 computations are required once to perform that initial inversion.

A second, much simpler approach is also commonly used. In this case R_{N-1}^{-1} is

initialized as:

$$\hat{R}_{n-1}^{-1} = \eta I_N \tag{6.32}$$

Where η is a large positive constant and I_N is the *N-by-N* identity matrix. Since R_{N-1}^{-1} almost certainly will not equal ηI_N, this inaccuracy will influence the final estimate of R_k and hence W_k. A s a practical matter, however, η can usually be made large enough to avoid significant impact on W_{L-1}^o while still making R_{N-1} invertible. Because of the simplicity and the low computational cost, the second approach is the one of the most commonly used. It becomes even more theoretically justifiable when used with the exponentially weighted RLS algorithm to be discussed shortly.

The computational cost for the RLS algorithm:

As a prelude to developing even more efficient adaptive algorithms, we first should determine how much computation is required to execute the RLS algorithm.

We define that the 10 steps in the procedure can be grouped by their computational complexity:

(a) Order 1: Steps(iv) and (vii) require only a few simple operations, such as a subtraction or an addition and division. These are termed as *order1* and denoted *O(1)* because the amount of computation required is not related to the filter order.

(b) Order *N* : Steps (iii), (vi), (viii), and (ix) each require a vector dot product, a scalar-vector product, or a vector scale and sum operation. Each of these requires *N* additions for each iteration of the algorithm .The actual number of multiplications required for these steps is 4*N*, but we refer to them as order *N*, or *O(N)* ,because the computation requirement is proportional to *N*, the length of the filter impulse response.

(c) Order N^2: Step (v), a matrix vector product, and step (x) , the vector outer product, both require N^2 multiplications and approximately N^2 additions. These are termed $O(N^2)$ procedures.

The total number of computations needed to execute the RLS algorithm for each input sample pair { *x(k), d(k)* } is $2N^2+4N$ multiplications, an approximately equal number of additions ,and on division. Because this amount of computation is required for each sample pair, the total requirement of multiplications to process the sample window is

$$C_{RLS} = (L-N+1).\ 2N^2 + (L-N+1).\ 4N$$

There are several reasons for exploring and using RLS techniques:

(a) RLS can be numerically better behaved than the direct inversion of R_{ss};

(b) RLS provides an optimal weight vector estimate at every sample time, while the direct method produce a weight vector estimate only at the end of the data sequence; and

(c) This recursive formulation leads the way to even lower-cost techniques.

Table 6.2

(Comparison of Computational Complexity between an L-Layer MLP, a FLANN and a CFLANN in One Iteration with BP Algorithm)

Operations	MLP	FLANN	CFLANN
Weights	$\sum_{l=0}^{L-1}(n_l+1)n_{l+1}$	$n_1(n_0+1)$	$n_1(n_0+1)$
Additions	$3\sum_{l=0}^{L-1}n_l n_{l+1}+3n_l-n_0n_1$	$2n_1(n_0+1)+n_1$	$2n_1(n_0+1)+n_1$
Multiplications	$4\sum_{l=0}^{L-1}n_l n_{l+1}+3\sum_{l=1}^{L}n_l-n_0n_1+2n_L$	$3n_1(n_0+1)+n_0$	$3n_1(n_0+1)+n_0$
Tanh(.)	$\sum_{l=1}^{L}n_l$	n_1	n_1
Cos(.)/sin(.)	---	n_0	---

6.7 Comparison of Computational Complexity

Here, we present a comparison of computational complexity between an MLP and FLANN structure trained by the BP algorithm and CFLANN structure trained by RLS algorithm. Let us consider an L-layer MLP with n_l number of nodes (excluding the threshold unit) in layer l, $l = 0,1,\ldots,L$ where n_0 and n_l are the number of nodes in the input layer and output layer, respectively. Three basic computations, i.e., the addition, the multiplication and the computation of $\tanh(.)$ are involved for updating the weights of an MLP. In the case of FLANN, in addition, computations of $\cos(.)$ *and* $\sin(.)$ are also involved. The computations in the network are due to

1) Forward calculation to find the activation value of all nodes of the entire network;
2) Back error propagation for calculation of square error derivatives;
3) Updating of the weights of the entire network.

The total number of weights to be updated in one iteration in an MLP structure is given by $(\sum_{l=0}^{L-1}(n_l+1)n_l = 1)$ where as in the case of a FLANN the same is only (n_0+1). For the CFLANN case all the cases for FLANN are same except here $\cos(.)$ *and* $\sin(.)$ functions are not needed. Since hidden layer does not exist in a FLANN and CFLANN , the computational complexity is drastically reduced in comparison to that of an MLP. A comparison of computational load in one iteration, for an MLP and a FLANN structure is provided in Table 6.2.

6.8 Simulation and Results

Extensive computer simulations were carried out for MLP,FLANN and CFLANN structures. For both System Identification and Channel Equalization problem, a uniformly distributed random signal over the interval [-.5,.5] was applied to the FIR structure and a white Gaussian noise of 30dB was added to the output of the system. The learning parameter μ was suitably chosen for each structure to obtain best result.

Four different channels were studied with the following transfer function:

$CH = 1 : 0.209 + 0.995z^{-1} + 0.209z^{-2}$
$CH = 2 : 0.260 + 0.930z^{-1} + 0.260z^{-2}$
$CH = 3 : 0.340 + 0.903z^{-1} + 0.304z^{-2}$
$CH = 4 : 0.341 + 0.876z^{-1} + 0.341z^{-2}$

To study the effect of nonlinearity on the system performance four different nonlinear channel models with the following nonlinearities has been introduced.

$NL = 0 : b(k) = a(k)$

$NL = 1 : b(k) = \tanh(a(k))$

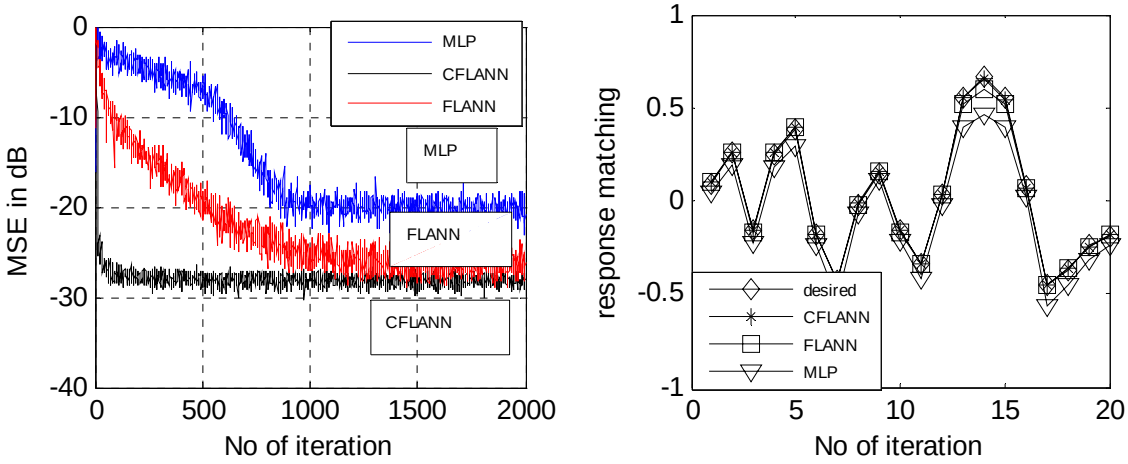
$NL = 2 : b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k)$

$NL = 3 : b(k) = a(k) - 0.9(a^3(k))$

The Convergence Characteristic

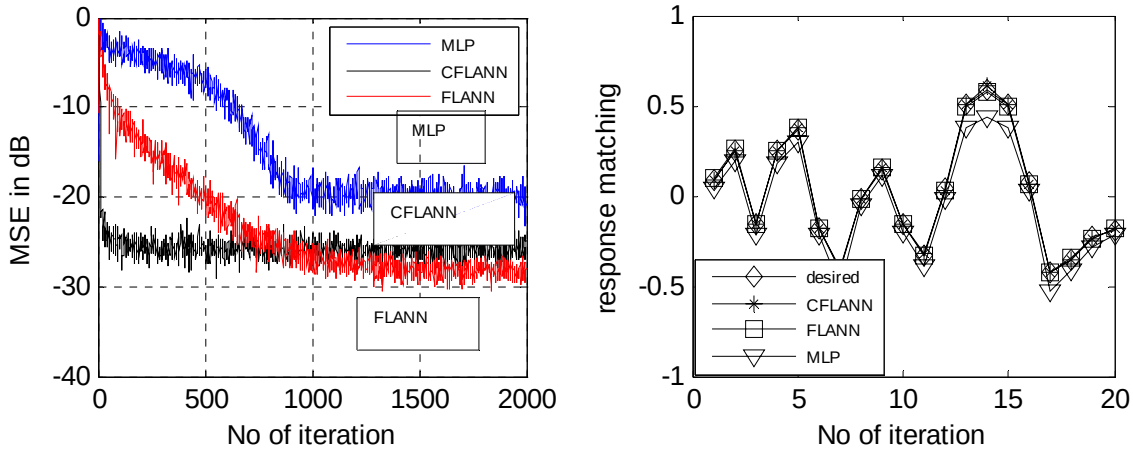
The convergence characteristics for CH=2 at SNR of 30 dB is simulated for the linear and nonlinear models. From MSE plot of the System Identification for NL=0,NL=1 and NL=3 was given. The MSE floor corresponding to the steady state value of the MSE is obtained after averaging over 100 independent runs each consisting of 3000 iterations to obtain optimal weight. The learning parameter μ is chosen to be 0.02. It can be observed that LMS based FLANN based structure shows much faster convergence and better MSE floor than MLP. Where as CFLANN updated with RLS shows faster and better convergence and it takes much less iteration than FLANN and MLP updated with LMS. Where as the repose matching plots for all the MLP, FLANN and CFLANN structure is same.

MSE = Mean Square Error



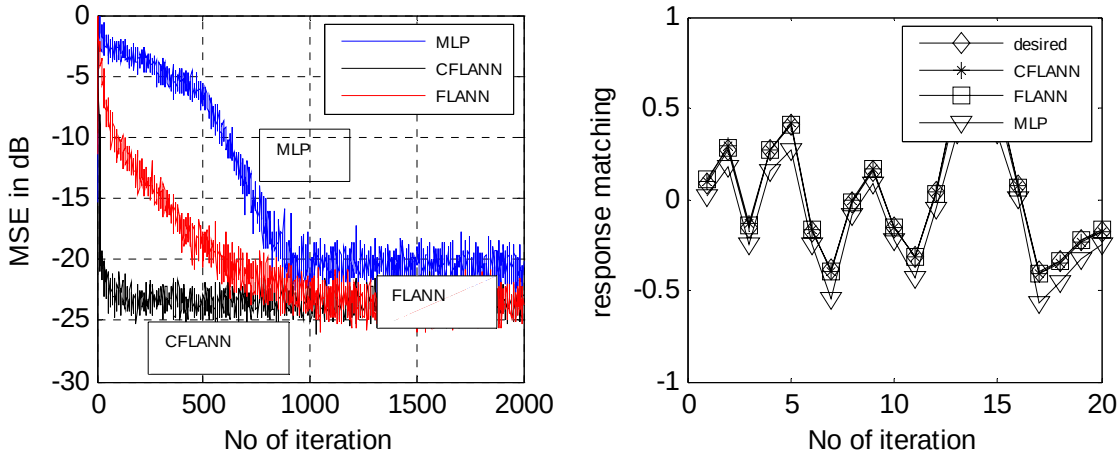
6.4(a) 6.4(b)

Fig.6.4 (a),(b) corresponds to the respective MSE and response matching plot of MLP, FLANN and CFLANN structure with the desired signal of NL=0.



6.5(a) 6.5(b)

Fig.6.5 (a),(b) corresponds to the respective MSE and response matching plot of MLP, FLANN and CFLANN structure with the desired signal of NL=1.



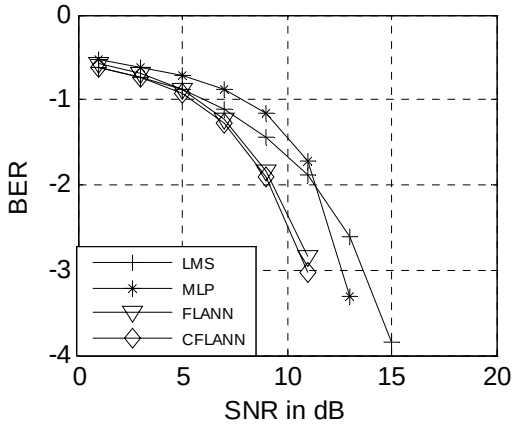
6.6(a) 6.6(b)

Fig.6.4 (a),(b) corresponds to the respective MSE and response matching plot of MLP, FLANN and CFLANN structure with the desired signal of NL=1.

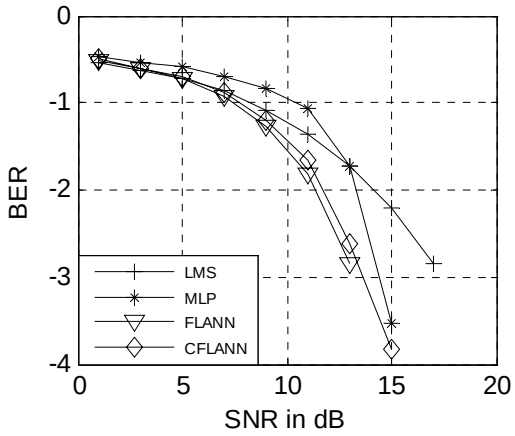
BER performance study

The BER provides the true picture of the performance of an equalizer. The computation of BER was carried out for the channel equalization using the three ANN structures and one FIR based structure updated with RLS algorithm is carried out. From the extensive computer simulation it

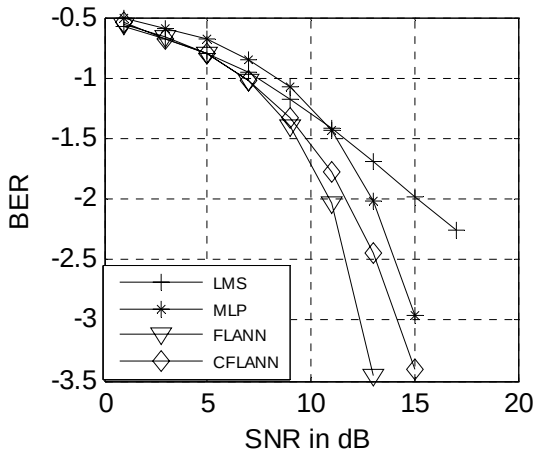
is seen that for all the linear and non linear cases works better than MLP and RLS based structure and performs almost same and in some cases better than FLANN structure with less no of computational complexity and faster convergence.



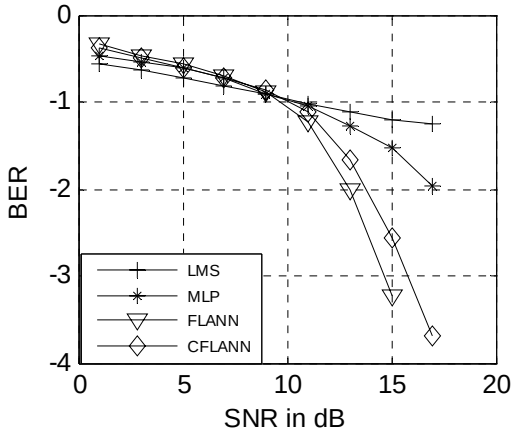
6.7(a)



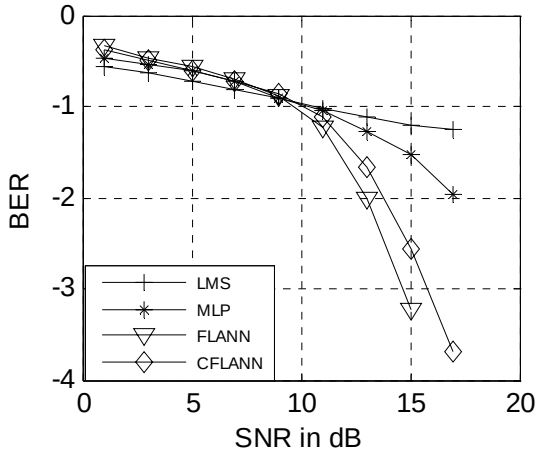
6.7(b)



6.7.(c)



6.7(d)



6.7(e)

Fig.6.7 (a), (b), (c), (d), (e), corresponds to the respective BER plot for RLS, MLP, FLANN and CFLANN equalizer structure with NL=0,NL=1,NL=2,NL=3 and NL=4

6.9 Summary

The present paper proposes a novel Chebyschev Functional Link ANN model for identification of nonlinear systems and equalizer structure for adaptive channel equalization with noise. Since it is a single layer structure and uses Chebyschev polynomials for expansion instead of trigonometric expansion it offers advantage in terms of computational complexity over MLP and FLANN structure. For faster and efficient training the RLS algorithm has been employed. Simulation study using known nonlinear plants has been carried out employing FLANN, MLP and the proposed model results show that the proposed model outperforms the other two models both in terms of convergence rate, MSE floor and BER performance. This structure may efficiently used in other signal processing applications including noise cancellation, prediction, system identification and control.

Chapter 7

ONLINE SYSTEM IDENTIFICAION

7.1. Introduction

Identification of a complex dynamic plant is a major concern in control theory. This interest stems from the need to give new solutions to some long standing necessities of automatic control; to work with more and more complex systems, to satisfy stricter design criteria, and to fulfill previous points with less and less a priori knowledge of the plant. In this context, a great effort is being made within the area of system identification, towards the development of nonlinear models of real processes [7.1].

Because of nonlinear signal processing and learning capability, artificial neural networks (ANN's) have become a powerful tool for many complex applications including functional approximation, nonlinear system identification and control, pattern recognition and classification, and optimization. The ANN's are capable of generating complex mapping between the input and the output space and thus, these networks can form arbitrarily complex nonlinear decision boundaries.

In contrast to the static systems that are described by algebraic equations, the dynamic systems are described by difference or differential equations. It has been reported that even if only the outputs are available for measurement, under certain assumptions, it is possible to identify the dynamic system from the delayed inputs and outputs using a multilayer perceptron (MLP) structure [7.2]. Narendra and Parthasarathy proposed the problem of nonlinear dynamic system identification using MLP structure trained by BP algorithm [7.3], [7.4]. At present most of the works on system identification using neural networks are based on multilayer feed forward neural networks with back propagation learning or more efficient variations of this algorithm n Identification based control approaches are reported in [7.8]-[7.9]. An approach for integrating evolutionary computation applied to the problem of system identification is presented in [7.10]. These methods have been applied to real processes and they have shown an adequate behaviour. However, most of the schemes for system identification have been demonstrated through empirical studies, or convergence of the output error has been shown under ideal conditions except in [7.11]. where detailed convergence analysis is given. As an alternative to the MLP, there has been considerable interest in radial basis function (RBF) networks [7.12]–[7.15], primarily because of its simpler structure. The RBF networks can learn functions with local variations and discontinuities effectively and also possess universal approximation capability [7.15]. This network represents a function of interest by using members of a family of compactly or locally supported basis functions, among which radially symmetric Gaussian functions, are

found to be quite popular. A RBF network has been proposed for effective identification of nonlinear dynamic systems [7.16], [7.17]. In these networks, however, choosing an appropriate set of RBF centers for effective learning, still remains as a problem. Considering as a special case of RBF networks, the use of wavelets in neural networks has been proposed [7.18], [7.19]. In these networks, the radial basis functions are replaced by wavelets, which are not necessarily radial-symmetric. Wavelet neural networks for function learning and nonparametric estimation can be found in [7.20], [7.21].

Originally, the Pao [7.22] proposed Functional link ANN (FLANN). He has shown that, this network may be conveniently used for function approximation and pattern classification with faster convergence rate and lesser computational load than an MLP structure. The FLANN is basically a single layer neural network and the need of the hidden layer is removed and hence, the BP learning algorithm used in this network becomes very simple. The functional expansion effectively increases the dimensionality of the input vector and hence the hyper planes generated by the FLANN provide greater discrimination capability in the input pattern space. Pao *et al.* have reported identification and control of nonlinear systems using a FLANN [7.23]. Chen and Billings [7.24] have reported nonlinear dynamic system modeling and identification using three different ANN structures. They have studied this problem using an MLP structure, a radial basis function (RBF) network and a FLANN and have obtained satisfactory results with all the three networks.

Pattern classification using Chebyschev neural networks has been reported in [7.25].It has been proved that Chebyschev neural network (CNN) has powerful representation capabilities whose input is generated by using a subset of Chebyschev polynomials [7.26]. CNN is a functional link networks based on Chebyschev polynomials. Being a single layer neural network, its computational complexity is less intensive as compared to (MLP) and can be used for on-line learning. Pattern classification using CNN has been reported in [7.25]. System identification using CNN in discrete time domain is reported in [7.27] where it is shown that CNN based identification requires less computation as compared to MLP. Additionally, the identification method uses off-line training of discrete time plants. In [7.28] on-line system identification using CNN of SISO systems in both discrete and continuous time domain is taken up.

The primary purpose of this chapter is to develop a computationally efficient and accurate algorithm for on-line system identification that is applicable to a variety of problems. This paper highlights the use of Chebyschev neural network models to identify time series problem as well

as discrete time plants. The identification scheme exhibits a learning-while-functioning feature instead of learning-then-functioning, so that the identification is on-line without any need of off-line learning phase. The training scheme is based on recursive least squares algorithm which guarantees convergence of the Chebyschev neural network weights. The proposed scheme also ensures good performance in the sense that the identification error is small and bounded. The convergence issue is shown through Lyapunov stability theory. The results are compared with certain existing identification algorithm.

7.2. Problem Statement

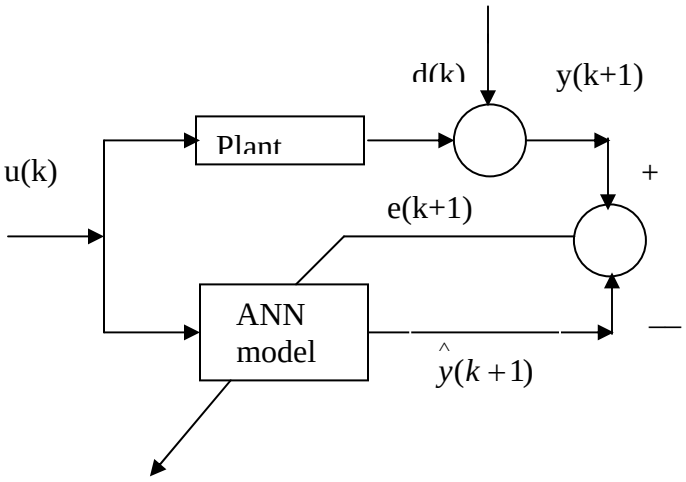


Fig.7.1. Basic Block diagram System Identification Model

The method for system identification of a time invariant, causal, discrete time plant is depicted in Fig.7.1**.** the plant is excited by a signal $u(k)$, and the output $y(k+1)$ is measured. The plant is assumed to be stable with known parameterization but with unknown values of the parameters. The objective is to construct a suitable identification model which when subjected to the same input $u(k)$ as the plant, produces an output which approximates $y(k+1)$ in the sense described by $\left\| y - \hat{y} \right\| \le \varepsilon$ for some desired $\varepsilon > 0$ and a suitably defined norm. The choice of the identification model and the method of adjusting its parameters based on the identification error constitute the two principal parts of the identification problem. This method of identification is applied to time series problem and SISO and MIMO discrete time plants.

The SISO and MIMO plants are described by the difference equations:

Model 1:

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g[u(k), u(k-l), ..., u(k-m+1)] + d(k) \quad (7.1)$$

Model 2:

$$y(k+1) = f[y(k), y(k-1), ..., y(k-n+1)] + \sum_{i=1}^{m-1} \beta_i u(k-1) + d(k) \quad (7.2)$$

Model 3:

$$y(k+1) = f[y(k), y(k-1), ..., y(k-n+1)] + g[u(k), u(k-1), ..., u(k-m+1)] + d(k) \quad (7.3)$$

Model 4:

$$y(k+1) = f[y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-m+1)] + d(k) \quad (7.4)$$

Where $u(k)$, $y(k)$ and $d(k)$ represent the input of the plant, output of the plant and disturbance acting on the plant, respectively, at the kth instant of time. Here, $f(.) \in \Re^n, g(.) \in \Re^n$, $y(k) \in \Re^n, u(.) \in \Re^n, \alpha_i \in \Re^{n \times m}, \beta_i \in \Re^{n \times m}, d(k) \in \Re^n$ with $\|d(k)\| \le d_M$ a known constant. These four models taken from the literature represent a fairly large class of systems. The ability of neural networks to approximate large classes of nonlinear function makes them prime candidates for the identification of nonlinear plants. Under fairly weak conditions on the functions f and/o g, CNN can be constructed to approximate such mappings over compact sets.

7.3. Chebyschev Neural Network

7.3.1 Structure of CNN

Chebyschev neural network is a single layer NN structure. CNN is a functional link network (FLANN) based on Chebyschev polynomials. One way to approximate a function by a polynomial is to use a truncated power series. The power series expansion represents the

function with very small error near the point of expansion, but the error increases rapidly as we employ it at points farther away. The computational economy to be gained by Chebyschev series increases when the power series is slowly convergent. Therefore, Chebyschev series are frequently used for approximations to functions and are much more efficient than other power series of the same degree. Among orthogonal polynomials, the Chebyschev polynomials occupy an important place, since, in the case of a broad class of functions, expansions in Chebyschev polynomials converge more rapidly than expansions in other set of polynomials. Hence, we consider the Chebyschev polynomials as basis functions for the neural network.

The Chebyschev polynomials can be generated by the following recursive formula:

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x), \quad T_0(x) = 1 \tag{7.5}$$

For example, consider a two dimensional input pattern $X = [x_1 \; x_2]^T$. An enhanced pattern obtained by using Chebyschev functions is given by:

$$\phi = [1 \; T_1(x1) \; T_2(x2) .... T_1(x2) \; T_2(x2) ....]^T \tag{7.6}$$

Where $T_i(x_j)$ is a Chebyschev polynomial, *ith* order of polynomials chosen and j = 1, 2. The different choices of $T_1(x)$ are x , $2x$, $2x-1$ *and* $2x+1$. In this chapter, $T_1(x)$ is chosen as x.

The following results are stated for the function approximation capability of CNN in the form of Theorem1.

Theorem 1: Assume a feed forward MLP neural network with only one hidden layer and activation functions of the output layer are all linear. If all the activation functions of the hidden layer satisfy the Riemann integrable condition, then the feed forward neural network can always be represented as a Chebyschev neural network. The detailed proof of the theorem can be found in [29].

The architecture of the CNN consists of two parts, namely numerical transformation part and learning part. Numerical transformation deals with the input to the hidden layer by approximate transformable method. The transformation is the functional expansion (FE) of the input pattern comprising of a finite set of Chebyschev polynomials. As a result the Chebyschev polynomial basis can be viewed as a new input vector. The learning part is a functional link neural network based on Chebyschev polynomials.

The output of the single layer neural network is given by:

$$\hat{y} = \hat{W}^T \phi \tag{7.7}$$

Where $\hat{W}$ are the weights of the neural network given by $\hat{W} = [w1\ w2\ .....]^T$.

A general nonlinear function $f(x) \in C^n(S), x(t) \in S$ can be approximated by CNN as:

$$f(x) = \hat{W}^T \phi + \varepsilon \tag{7.8}$$

Where ε is the CNN functional reconstruction error vector. In CNN, functional expansion of the input increases the dimension of the input pattern. Thus, creation of nonlinear decision boundaries in the multidimensional input space and approximation of complex nonlinear systems becomes easier.

7.3.2. Learning Algorithm

The problem of identification consists in setting up a suitably parameterized identification model and adjusting the parameters of the model to optimize a performance function based on the error between the plant and identification model outputs. CNN, which is a single layered neural network, is linear in the weights and nonlinear in the inputs is the identification model used in this paper. We shall use the recursive least squares method with forgetting factor as the learning algorithm for the purpose of on-line weight updation. The performance function to be minimized is given by:

$$E = \sum_{i=1}^{k} \lambda^{k-i} |e(i)|^2 \tag{7.9}$$

The algorithm for the discrete time model is given by:

$$\begin{aligned} \hat{W}(n) &= \hat{W}(n-1) + k(n)e(n) \\ k(n) &= \frac{\lambda^{-1} P(n-1)\phi(n)}{1 + \lambda^{-1}\phi^T(n)P(n-1)\phi(n)} \end{aligned} \tag{7.10}$$

$$\begin{aligned} e(n) &= y(n) - \hat{y}(n) \\ P(n) &= \lambda^{-1}P(n-1) - \lambda^{-1}K(n)\phi^T(n)P(n-1) \end{aligned} \tag{7.11}$$

Where λ is the forgetting factor and ϕ is the basis function formed by the functional expansion of the input and $P(0) = cI, c$ is a positive constant $\|P(t)\| < R_0, R_0$ is a constant that serves an upper bound for $\|P(t)\|$. All matrix and vectors are of compatible dimension for the purpose of computation. The following assumption is needed for the stability analysis.

A3. The ideal weights of the CNN are bounded so that $\|W^*\| \le W_M$ where W^* are the ideal weights. We only need to know that ideal weights exist to prove the convergence analysis. The

exact values of the ideal weights need not be known as they are not required for the purpose of identification.

7.3.3. Stability Analysis

The convergence of CNN weights is shown through Lyapunov stability theory. Consider a Lyapunov function candidate:

$$V_n = \lambda^{k-n}\tilde{W}^T(n)P(n)^{-1}\tilde{W}(n) \tag{7.12}$$

Where,

$$\tilde{W}(n) = W^*(n) - \hat{W}(n) \quad . \tag{7.13}$$

Then

$$\begin{aligned}\Delta V_n &= V_n - V_{n-1} \\ &= \lambda^{k-n}\tilde{W}^T(n)P(n)^{-1}\tilde{W}(n-1)\end{aligned} \tag{7.14}$$

From (7)

$$\begin{aligned}\tilde{W}(n) &= \tilde{W}(n-1) - k(n)e^T(n) \\ &= \lambda P(n)P(n-1)^{-1}\tilde{W}(n-1)\end{aligned} \tag{7.15}$$

Thus,

$$\begin{aligned}\Delta V_n &= \lambda^{k-n+1}[\tilde{W}^T(n) - \tilde{W}^T(n-1)] \times P(n-1)^{-1}\tilde{W}(n-1) \\ &= \frac{-\lambda^{k-n+1}e(n)^2}{\lambda + \phi^T(n)P(n-1)\phi(n)} < 0\end{aligned} \tag{7.16}$$

This shows that $V_n < 0$ and $\Delta V_n < 0$. By using Lyapunov second method, $\tilde{W} \to 0$ as $n \to \infty$ this implies that $W(n) \to W^*$ *as* $n \to \infty$.

Table 7.1

(Comparison of the number of variables chosen and the MSE obtained using Chebyschev neural networks.)

No. of inputs	Inputs chosen	Mean Square error
2	y(k-1),u(k-4)	9.214
3	y(k-1,u(k-1),u(k-2)	0.1016
6	y(k-1),y(k-2),y(k-3),u(k-1),u(k-2),u(k-3)	0.0695
10	y(k-1),…,y(k-4).u(k-1),…,u(k-6)	8.6684

Table 7.2

(Mean square error comparison by different identification methods)

Model	Identification method	Mean Square Error
Kukolj and Levi [7.14]	Neuro-fuzzy (off-line)	0.129
Oh and Pedryez [7.10]	Polynomial NN (off-line)	0.027
Proposed model	Chebyschev NN (on-line)	0.0695

7.4 Simulations

The developed model is now applied to three different problems: Box Jenkins identification problem, a SISO and a MIMO problem. The CNN identifier derived here require no apriori knowledge of the dynamics of the nonlinear system. Moreover no offline learning phase is required.

7.4.1. Box and Jenkins' Identification Problem

Box and Jenkins' gas furnace data are frequently used in performance evaluation of system identification methods. The data can be obtained from the site http://www.stat.wisc.edu/_reinsel/bjr-data/gasfurnace. The example consists of 296 inputs–output samples recorded with a sampling period of 9 s. The gas combustion process has one variable, gas flow u(k), and one output variable, the concentration of CO2, y(k). The instantaneous values of output y(k) have been regarded as being influenced by six variables $y(k-1), y(k-2), y(k-3), u(k-1), u(k-2), u(k-3)$. In the literature, the number of variables influencing the output varies from 2 to 10. In the proposed method, six variables were chosen after several trials. Table 7.1. gives a comparison of the number of variables chosen and the MSE obtained using Chebyschev neural networks. The MSE turned out to be the least with six variables. Fig 7.2. shows actual and estimated values, obtained by means of the proposed on-line neuro-identification model. An MSE of 0.0695 was achieved with the weights of the CNN initialized to zero and each of the six inputs in to two terms. The result achieved belongs to the category of the best available results that have been reported in the literature. The results obtained by the proposed method have been compared with two of the results that have been recently reported in the literature in Table 7.2. Each model is identified by the name of the author, publication year and reference number. The next column lists the model used and the

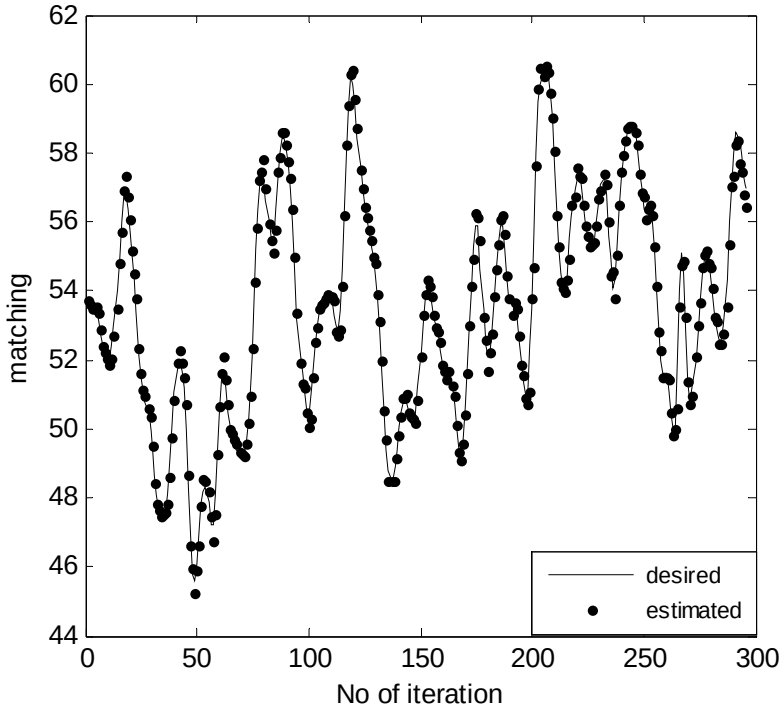


Fig.7.2. response matching plot for the Box and Jenkins' Identification problem

mode of identification (on-line or off-line). The last column illustrates the accuracy of the model using MSE. Table 7.2. Contrasts the performance of the proposed method with the other two models studied recently in the literature based on off-line techniques. The results clearly reveal that the proposed method being fast and simple can be used on-line whereas the other two methods being off-line methods involve a training phase and a testing phase. Moreover, the proposed model clearly outperforms [7.4] and also [7.10] where it can be seen that the MSE in the testing data is 0.085. The detailed comparisons of the various methods reported in the literature can be found in [7.4] and also [7.10]. When the six inputs are expanded into three terms the MSE in this case as can be seen from Table 7.3 is 0.1572. Table 7.3 gives the MSE for the proposed model for inputs expanded to different number of terms along with the number of weights to be updated in the CNN. From this table it becomes clear that when the order of the Chebyschev polynomial expansion is taken as two, the MSE is minimum. Therefore, for this problem we have expanded the six inputs to two terms each.

Table 7.3

(MSE for the proposed model for inputs expanded to different number of terms along with the number of weights to be updated)

No of Chebyschev Polynomials	No. of weights of CNN	Mean Squared Error
1	7	0.0740
2	13	0.0695
3	19	0.1572
4	25	8.7764

Table 7.4

(Comparison of computational complexity and performance between (CNN and MLP))

Number of	CNN	MLP
Weights	11	120
Tan *h*	-	20
MSE	2.77×10^{-4}	5.15×10^{-4}

7.4.2. SISO Plant

We consider a single input single output discrete time plant described by [7.26].

$$x(k+1) = f[x(k), x(k-1), x(k-2), u(k), u(k-1)] \tag{7.17}$$

$$\hat{x}(k+1) = \hat{f}[x(k), x(k-1), x(k-2), u(k), u(k-1)] \tag{7.18}$$

$$u(k) = \begin{cases} \sin\left(\dfrac{2\pi k}{250}\right) & for \quad 0 < k \le 250 \\ 0.8\sin\left(\dfrac{2\pi k}{250}\right) & for \quad 0 < k \le 250 \end{cases} \tag{7.19}$$

Where the unknown nonlinear function f is given by:

$$f[a_1, a_2, a_3, a_4, a_5] = \frac{a_1 a_2 a_3 a_5 (a_3 - 1) + a_4)}{(1 + a_2^{\,2} + a_3^{\,2})} \tag{7.20}$$

To identify the plant, the model is governed by the difference equation given by $\hat{x}(k+1)$ and $\hat{f}$ is estimated using a CNN. For the CNN, the input $\{x(k), x(k-1), x(k-2), u(k), u(k-1)\}$ is

expanded to 11 terms using Chebyschev polynomials. The input to the actual system and the neural network model is given by Eq (7.20). The CNN weights are initialized to zero. Weights of

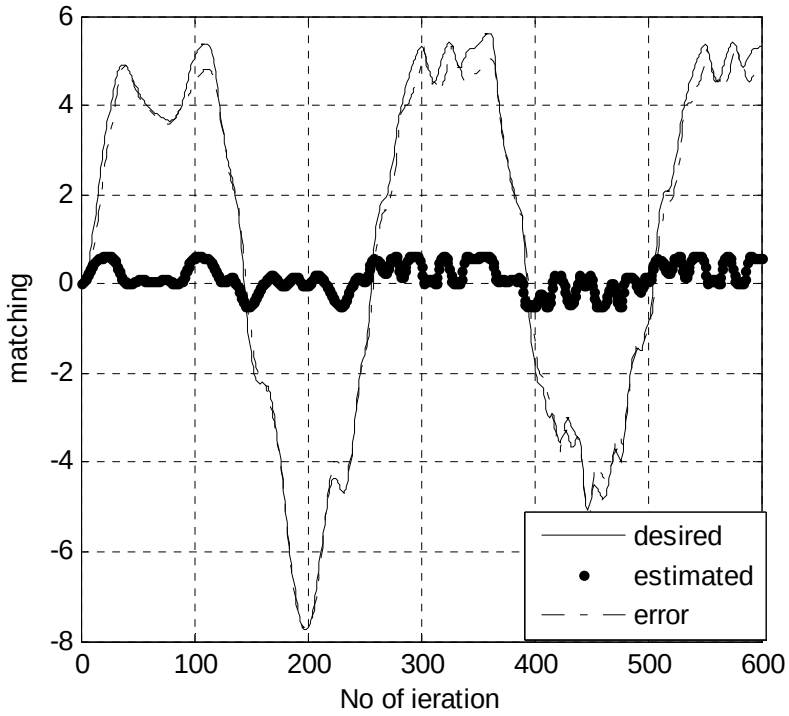


Fig.7.3. Response matching plot of the SISO Plant

the CNN are updated using the algorithm given by Eq. (7.10). The performance of the proposed CNN is compared with that of an MLP. For this purpose, the MLP architecture, initial weights of the neural network, the parameters of the learning law and the learning law are the same as used by them. Matlab's randn (-) function is used to generate noise, with mean value zero and covariance value $(0.01)^2 s^{-1}$. This noise is then added to the true output obtained from the system given by Eq (7.17). The performance of the identification model with this noise level is shown in Fig 7.3 for CNN. In both the cases the performance is satisfactory. A standard quantitative measure for performance evaluation is the mean squared error. Table 7.4 gives a comparison of the computational complexity and the performance of the proposed method using CNN and the method proposed by Yu and Li using MLP. From Table 7.4 it becomes clear that the CNN is not only computationally less intensive but also gives a better performance as compared to MLP.

7.4.3. MIMO Plant

Consider the two input two output nonlinear discrete time system described by [7.5]

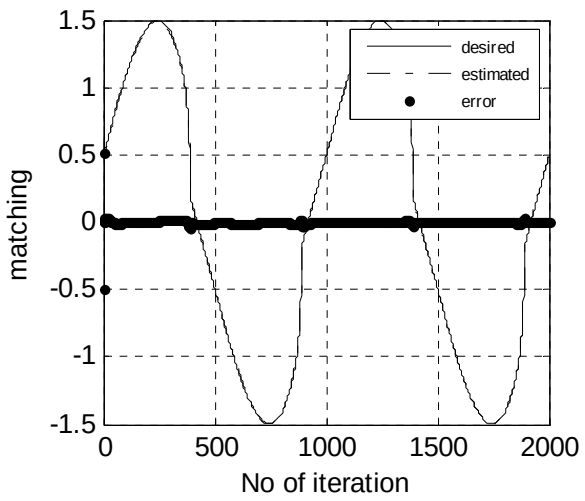
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \dfrac{x_2(k)}{1+x_1^{\,2}(k)} \\ \dfrac{x_1(k)}{1+x_1^{\,2}(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} + \begin{bmatrix} d_1(k) \\ d_2(k) \end{bmatrix} \tag{7.21}$$

$$\begin{aligned} \hat{x}_1(k+1) &= f_1[x_1(k), x_2(k), u_1(k), u_2(k)] \\ \hat{x}_2(k+1) &= f_2[x_1(k), x_2(k), u_1(k), u_2(k)] \end{aligned} \tag{7.22}$$

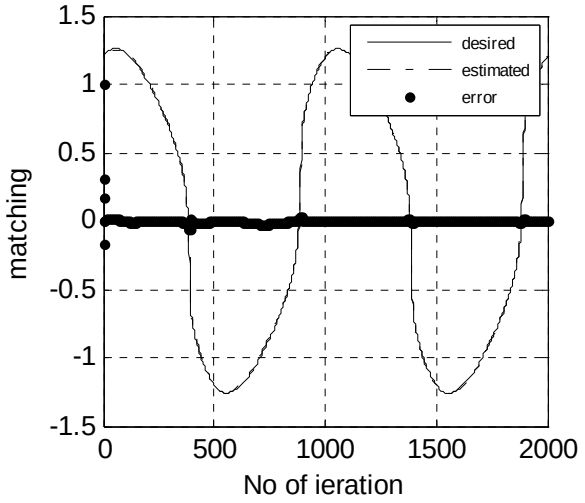
Where the inputs u1(k) and u2(k) is given by:

$$\begin{aligned} u_1(k) &= \cos\left(\frac{2\pi k}{100}\right) \\ u_2(k) &= \sin\left(\frac{2\pi k}{100}\right) \end{aligned} \tag{7.23}$$

A single CNN with two outputs is used to approximate f_1 *and* f_2. For the CNN, the inputs are $\{u_1(k), u_2(k), x_1(k), x_2(k)\}$ which are expanded to nine terms using Chebyschev polynomials. The neural network weights are initialized to zero. The weights of the neural network are updated using the algorithm given by Eq (7.10). A white Gaussian noise with mean zero and covariance of $(0.03)^2$ is then added to the true output obtained from the system given by Eq (7.21).Fig 7.4(a),(b) and Fig 7.5(a),(b) (presents the responses of the identifier for the proposed algorithm without and with noise condition. The upper graph gives the actual output, estimated output and error of the first output and the lower graph for the second output. It is clear from these figures that the response of the identifier is extremely impressive though the noise condition is extremely high.

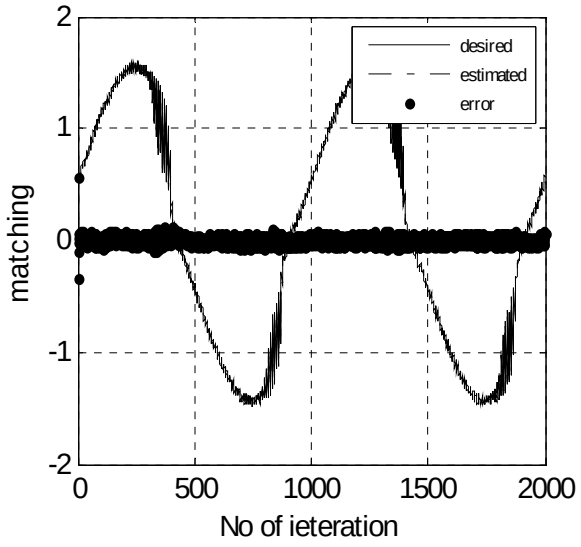


7.4 (a)

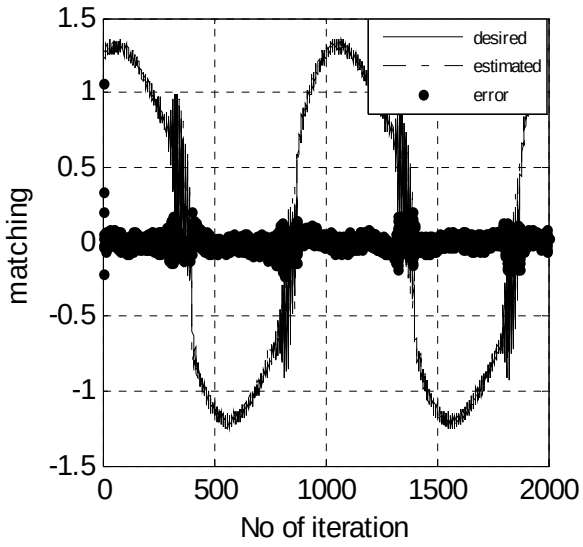


7.4(b)

Fig 7.4.(a),(b) are the corresponding matching of desired ,estimated and error plot of output1 and output2 without noise



7. 5(a)



7.5(b)

Fig 7.5.(a),(b) are the corresponding matching of desired ,estimated and error plot of output1 and output2 with noise

7.5 Summary

In this chapter, we have presented identification schemes in a feed forward neural network framework that ensures identification of general nonlinear dynamical systems with smooth nonlinearities. Our proposed scheme firstly does not need any off-line training, secondly requires

no initialization of neural network weights. Initially, the neural network weights are assumed to be zero. As the neural network structure is single layer network, it is computationally fast and simple. The proposed method is applied to a time series problem and nonlinear SISO and MIMO discrete time systems. It is important to remark that the identification model formed in this way can be used to characterize a wide class of very diverse problems. The recursive least squares algorithm captures the dynamics of the systems through the updation of the weights of the neural network model. The computational efficiency and the accuracy of the proposed methodology make it very well suited for applications in the design of on-line adaptive identification models for a wide class of complex systems. Currently, work is going on the use of evolutionary computation with CFLANN structure to approximate time series problem and to identify the time varying discrete time system.

Chapter 8

CONCLUSION AND REFERENCES

8.1 Conclusion

In this thesis with extensive computer simulation we have studied about the behavior of the nonlinear systems and channels with different types of linear and nonlinear structures like FIR, MLP, FLANN and CFLANN with different types of algorithm. The objective of this research is to evaluate the performance of the proposed model for the linear and nonlinear identification, equalization problem.

Through computer simulation we have seen that BLMS algorithm works much faster and gives better bit error rate performance than the conventional than the conventional LMS algorithm for System Identification and Channel Equalization problem. Furthermore efficient block algorithms such as the Fast Fourier Transform (FFT) was used to advantage in terms of faster convergence and less mathematical complexity when implementing block filters in frequency domain but its performance degrades for nonlinear conditions.

We have discussed and simulated a lot of nonlinear structures for nonlinear problems such as MLP, FLANN and CFLANN. Since RLS algorithm works much faster than the conventional LMS algorithm we have combined it with CFLANN structure and the proposed structure works much better than its counterparts in terms of faster convergence, less mathematical complexity, better MSE floor and BER plot in case of non linear equalizer case.

In the chapter,6 the proposed structure was successfully tested with non-linear dynamic systems Such as Box and Jenkins problem, SISO and MIMO plant identification problem ands its value was compared with standard nonlinear structures and it was seen that our proposed scheme much better than its counterpart. It was concluded that the proposed structure exhibits can learning-while functioning instead of learning then functioning likes the other structures and needs no offline-learning phase.

8.2 Future Work

Sometimes we face multimodal problems. There we fail to find the solution because of the inability of the gradient-based algorithms to reach the global solution. In those cases stochastic algorithms gives better result. Evolutionary computation technique such as Genetic Algorithm, Bacteria Foraging, PSO many more gives better results in theses conditions. But these algorithms takes a lots of time, so work should be carried on to develop some faster hybrid Evolutionary algorithms and be implemented with these systems to give better results.

8.3 References

Chapter1 References

[1.1] Haykin S., Neural Networks, Ontario Canada:Maxwell Macmillan, 1994

[1.2] Nguyen D. H**.** and Pidrow B. ," Neural networks for self- Learning control System," Int. J. Contr., vol. 54, no. 6, 1931, pp. 1439- 1451.

[1.3] Pao Y. H., Adaptive Pattern Recognition and Neural Networks ,Reading, MA: Addision-Wesley, 1989.

[1.4] Chen S., Billings S. A. and Grant P.M. , "Recursive hybrid algorithm for Nonlinear System identification using radial basis function networks," Int. J. Contr. vol. **55,** no. **5,**1992 **,** pp. 1051- 1070.

[1.5] Panda G. and Das D. P. , "Functional Link Artificial Neural Network for Active Control of Nonlinear Noise Processes", Intl. Workshop Acoust. Echo Noise Control (IWAENC"03), Kyoto, Japan, pp.163-166, Sep. 2003.

[1.6] Namatame A. and Uema N. , "Pattern Classification with Chebyschev Neural Network," Neural Networks, , vol 3, Mar. 1992, pp. 23-31.

[1.7] Namatame A. and Ueda N. , Pattern classification with Chebyschev neural network, Int. J. Neural Netw. Vol.3 (March) (1992), pp.23–31.

Chapter2 References

[2.1] Ljung L. , System Identification. Englewood Cliffs, NJ: Prentice-Hall, 1987

[2.2] Akaike H. , . "A new look at the statistical model identification". IEEE Trans. on Automatic Control, AC-19, 1974,pp.716-723, 1974

[2.3] Widrow Bernard and Streans Samuel D., Adaptive Signal Processing, Pearson Education Publisher.

[2.4] Haykin Simon , . Adaptive Filter Theory, Pearson Education Publisher,2003

[2.5] Oppenheim A. V. and Schafer R. W. , Discrete-Time Signal Processing, Prentice-Hall,1989.

[2.6] Proakis J. G. and Manolakis D. G. , Digital Signal Processing - Principles, Algorithms,and Applications, Prentice Hall, 1996.

[2.7] Larimore M.G. , Treicheler J.R. and.jhnson, C.R. "SHARF: an algorithm for adaptive IIR digital filters," IEEE Trans.Acoust.Speech Signal Process., vol. ASSP-28, 1980,pp.428,Aug..

[2.8] Chen S. , Billings S. A. and Grant P. M. "Nonlinear system identification using neural networks", Int. J. Contr., vol. 51, no. 6 ,June 1990, pp. 1191-1214.

[2.9] Narendra K. S. and Parthasarathy K. "Identification and control of dynamic systems using neural networks", IEEE Trans. Neural Networks, vol. 1, Mar, 1990, pp. 4-27.

[2.10] Shynk J. J. "Adaptive IIR filtering," IEEE Acoust., Speech, Signal,Processing Mag., 1989,pp. 4–21.

[2.11] Regalia P. A. , Adaptive IIR Filtering in Signal Processing and Control., New York: Marcel Dekker Inc., 1995.

[2.12] Crawford D.H. , Stewart R.W. and Toma E. , An Alternative and Effective Adaptive IIR Filter Structure" .IEE Electronics Letters, Vol. 31: 26th October 1995pp 1906-1907.

Chapter3 References

[3.1] Chen, S., Mulgrew, B., McLaughlin, S., "Adaptive Bayesian Equaliser with Decision Feedback", ", IEEE Trans Signal Processing, vol.41, no.9, , Sept. 1993,pp.2918-2927.

[3.2] Macchi, O., "Adaptive processing, the least mean squares approach with applications in transmission", West Sussex:England ,John Wiley and Sons, 1995.

[3.3] Siu, S., "Non-linear Adaptive Equalisation based on multi-layer perceptron Architecture", Ph.D. Thesis, Faculty of Science, University of Edinburgh, 1990

[3.4] Forney, G., "Maximum-likelihood sequence estimation of digital sequences in the presence of inter-symbol interference", IEEE Trans, inform, theory, vol.IT-18, 1972,pp.363-378,

[3.5] Qureshi, S., "Adaptive equalization", Proceedings of the IEEE, vol.73, Sept. 1985,no.9, pp.1349-1387.

Chapter4 References

[4.1] Burrus C. , "Block implementation of digital filters," IEEE Trans. Circuit Theory, vol. CT- 18, Nov. 1971.

[4.2] "Block realization of digital filters," IEEE Trans. Audio Electroacoust., vol. AU-20 Oct. 1972.

[4.3] Oppenheim A. V. and Schafer R. W., Digital Signal Processing. tion of IIR digital filters," Proc. IEEE, vol. 65, July 1977.

[4.4] Rabiner L. R. and Gold B. , Theov and Application of Digital Signal E. 0. Brigham,The Fast Fourier Transform. New Jersey: Prentice- Processing. New Jersey: Prentice-Hall, 1975.

[4.5] Brigham E. 0., The Fast Fourier Transform. New Jersey: Prentice- Processing. New Jersey: Prentice-Hall, 1975.

[4.6] Widrow B., McCool J., Larimore M., and Johnson C., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," Proc. IEEE, vol. 64, August 1976.

[4.7] Widrow B., McCool J.,, "A Comparison of Adaptive Algorithms Based on theMethods of Steepest Descent and Random Search," IEEE Trans. Antennas and Propagation. Vol. AP-24. No. 5. September 1976.

[4.8] Widrow B., Mantey P. E., Griffiths L. J., and Goode B. B., "Adaptive antenna systems," Proc. IEEE, vol. 55, Dec. 1967.

Chapter5 References

[5.1] Clark Gregory A. and Parker Sydney R.,"A Unified Approach to Time Domain Realization of FIR Adaptive Digital Filters"IEEE Trans. Acoust., Speech, signal Process., vol. ASSP-31,no.5, Oct. 1983,pp.1073-1083,

[5.2] Oppenheim A. **V.** and Schafer R. W., Digital Signal Processing. Engelwood, Cliffs, NJ: Prentice-Hall, 1975

[5.3] Brigham E.O., The Fast Fourier Transform. Englewood Cliffs, NJ: Prentice-Hall, 1974

[5.4] Walzman T. and Schwartz M., "Automatic equalization using the discrete frequency domain," IEEE Trans. Inform. Theory, vol. IT-19, Jan. 1973, pp. 59-68.

[5.5] Maiwald D., Kaeser H. P., and Closs F., "On reducing the number of operations in adaptive equalizers," IBM Rep. RZ 918,31394, pp. 1-29, Sept. 1978

[5.6] Dentino M. McCool J., J., and Widrow B., "Adaptive filtering in the frequency domain," Proc. IEEE, vol. 66, Dec. 1978,pp. 1658-1659.

[5.7] Reed F. A. and Feintuch P. L., "A comparison of LMS adaptive cancellers implemented in the frequency domain and the time domain," IEEE Trans. Circuits Syst., vol. CAS-28, June 1981, pp. 610-615, and IEEE Trans. Acoust., Speech, Signal Processing, Joint Special Issue on Adaptive Signal Processing, vol. ASSP-29, June 1981,pp. 770-775.

[5.8] Bershad N. J. and Feintuch P. L., "Analysis of the frequency domain adaptive filter," Proc. IEEE, vol. 67, Dec. 1979, pp. 1658-1659.

[5.9] Ferrara E**.** R., "Fast implementation of LMS adaptive filters,"IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-28, Aug.1980, pp.474-475.

[5.10] Mansour D. and Gray A. H., Jr., "Unconstrained frequency domain adaptive filter," IEEE Trans. Acoust., Speech, Signal Processing ", vol. ASSP-30, Oct. 1982.,pp. 726-734.

[5.11] Parikh D., Mansour D., and Markel J. D., "Study of echo canceling algorithms for full duplex telephone networks with vocoders," in Proc. ICASSP, Atlanta, GA, 1981, pp. 1074-1077.

Chapter6 References

[6.1] Haykin S., Neural Networks ,Ontario Canada, Maxwell Macmillan, 1994.

[6.2] Nguyen D. H**.**, and Pidrow B., "Neural networks for self- Learning control System," Int. J. Contr., vol. 54, no. 6, 1931, pp. 1439- 1451.

[6.3] Y. H., Adaptive Pattern Recognition and Neural Networks. Reading, MA: Addision-Wesley, 1989.

[6.4] Chen S., Billings S. A., and Grant P.M., "Recursive hybrid algorithm for nonlinear System identification using radial basis function networks," Int. J. Contr. vol. 55, no. 5, 1992,pp. 1051- 1070.

[6.5] Panda G. and Das D. P.,"Functional Link Artificial Neural Network for Active Control of Nonlinear Noise Processes" Intl. Workshop Acoust. Echo Noise Control (IWAENC"03), Kyoto, Japan, Sep. 2003, pp.163-166, .

[6.6] Namatame A. and Uema N., "Pattern Classification with Chebyschev NeuralNetwork," Neural Networks, vol 3, Mar. 1992, pp. 23-31.

[6.7] Haykin S. and Kailath Thomas, Adaptive Filter theory, Ontario Canada, 2002.

Chapter7 References

[7.1] Pachter M., Reynolds O.R., Identification of a discrete time Dynamical system," IEEE Trans. Aerospace Electronic Syst ", Vol.36 ,no 1, 2000, pp 212–225.

[7.2] Chen S., Billings S. A. and Grant P. M., "Nonlinear system identification using neural networks," Int. J. Contr., vol. 51, no. 6, 1990,pp. 1191–1214. "Recursive hybrid algorithm for nonlinear system identification using radial basis function networks," Int. J. Contr., vol.

[7.3] "Neural networks and dynamical systems, Part II: Identification," Tech. Rep. 8902, CT,. vol 55, no. 5, Feb. 1989,pp. 1051–1070, Center Syst. Sci., Dept. Elect. Eng., Yale Univ., New Haven, winter 2006

[7.4] Nguyen D. H. and Widrow B., "Neural networks for self-learning control systems," Int. J. Contr., vol. 54, no. 6, 1991, pp. 1439–1451.

[7.5] Jagannathan S., Lewis F.L., Identification of a class of nonlinear dynamical systems using multilayer neural networks, in: IEEE International Symposium on Intelligent Control Columbus, Ohio, USA, 1994), pp. 345–351.1994.

[7.6] W.T. Miller, Sutton R.S., Werbos P.J., Neural Networks for Control, MIT Press, Cambridge, MA, 1990

[7.7] Narendra K.S., Parthasarthy K., Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Net. Vol.1 ,1990,pp 4–26.

[7.8] Kiong L.C., Rajeswari M., Rao M.V.C., Nonlinear dynamic system identification andcontrol via constructivism inspired neural networks, Appl. Soft Comput. ,vol.3,no.3,2003,pp 237–257.

[7.9] Castillo O., Melin P., Intelligent adaptive model based control of robotic dynamic systems with a hybrid fuzzy neural approach, Appl. Soft Comput. ,vol. 3 ,no. 4, 2003,pp 363–378.

[7.10] Montiel O., Castillo O., Melin P., Sepulveda R., The evolutionary learning rule for system identification, Appl. Soft Comput. , vol. 3,no. 4, 2003.pp 343–352.

[7.11] Jagannathan S., Lewis F.L., Identification of a class of nonlinear dynamical systems using Multilayer neural networks, in: IEEE International Symposium on Intelligent Control, Columbus, Ohio, USA, 1994, pp. 345–351.

[7.12] Poggio T and Girosi F., "Networks for approximation and learning, "Proc. IEEE, vol. 78, Sep. 1990 pp. 1481–1497,.

[7.13] Moody J. and Darken C. J., "Fast learning in networks of locally-tuned processing units," Neural Comput., vol. 1, 1989, pp. 281–294,.

[7.14] Park J. and Sandberg I. W. , "Universal approximation using radial basis function networks," Neural Comput., vol. 3, 1991 pp. 246–257,

[7.15] Hartman E. J., Keeler J. D., and Kowalski J. M., "Layered neural networks with Gaussian hidden units as universal approximation," Neural Comput., vol. 2, 1990,pp. 210–215.

[7.16] Chen S., Billings S. A., and Grant P. M., "Recursive hybrid algorithm for nonlinear system identification using radial basis function networks," Int. J. Contr., vol. 55, no. 5, 1992, pp. 1051–1070.

[7.17] Elanayar S. V. T. and Shin Y. C., "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," IEEE Trans. Neural Networks, vol. 5, July 1994, pp. 594–603.

[7.18] Zhang Q. and Benveniste A., "Wavelet networks," IEEE Trans. Neural Networks, vol. 3, Mar. 1992,pp. 889–898.

[7.19] Pati Y. C. and Krishnaprasad P. S., "Analysis and synthesis of feed forward neural networks using discrete affine wavelet transforms," IEEE Trans. Neural Networks, vol. 4, Jan. 1993, pp. 73–85.

[7.20] Zhang J., Walter G. G., Miao Y., and Lee W. G. W., "Wavelet neural networks for function learning," IEEE Trans. Signal Processing, vol. 43, , June 1995, pp. 1485–1497.

[7.21] Zhang Q., "Using wavelet network in nonparametric estimation," IEEE Trans. Neural Networks, vol. 8, Mar. 1997,pp. 227–236.

[7.22] Pao Y.H., Phillips S. M. and Sobajic D. J., "Neural-net computing and intelligent control systems," Int. J. Contr., vol. 56, no. 2, 1992,pp. 263–289.

[7.23] Patra J. C, "Some studies on artificial neural networks for signal processing applications," Ph.D. dissertation, Indian Inst. Technol., Kharagpur, Dec. 1996.

[7.24] Elanayar S. V. T. and Shin Y. C., "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," IEEE Trans . Neural Networks, vol. 5, July 1994,pp. 594–603.

[7.25] Namatame A. and Ueda N., "Pattern classification with Chebyschev neural networks," Intl. J. Neural Networks, vol. 3, Mar. 1992, pp. 23–31.

[7.26] Namatame A., in: Boubakins N. (Ed.), Connectionist Learning with Chebyschev Neural Network and Analyses of its Internal Representation, World Scientific, 1991, pp. 33–48.

[7.27] Patra J.C., Kot A.C., Nonlinear dynamic system identification using chebyschev functional link artificial neural networks, IEEE Trans. Syst. Man Cybern. B vol.32 ,no. 4,2002,pp 505–511.

[7.28] Purwar S., Kar I.N., Jha A.N., On-line system identification using Chebyschev neural networks, in: Proceedings of IEEE TENCON-2003 Conference, Bangalore, India, 2003.

[7.29] Lee T.T., Jeng J.T., The chebyschev polynomial based unified model neural networks for function approximations, IEEE Trans. Syst. Man Cybern. B 28 ,1998,pp 925–935.