8-1-1991

Tree-Structured Nonlinear Adaptive Signal Processing

C. S. Ravishankar
Purdue University

Saul B. Gelfand
Purdue University

Edward J. Delp
Purdue University

TREE-STRUCTURED NONLINEAR ADAPTIVE SIGNAL PROCESSING

C. S. Ravishankar

Saul B. Gelfand

Edward J. Delp

School of Electrical Engineering

Purdue University

West Lafayette, Indiana 47906

TR-EE 91-27

August 1991

ACKNOWLEDGMENTS

The authors wish to thank the Engineering Research Center and the National Science Foundation for supporting this research work.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

ABSTRACT

In communication systems, nonlinear adaptive filtering has become increasingly popular in a variety of applications such as channel equalization, echo cancellation and speech coding. However, existing nonlinear adaptive filters such as polynomial (truncated Volterra series) filters and multilayer perceptrons suffer from a number of problems. First, although high order polynomials can approximate complex nonlinearities, they also train very slowly. Second, there is no systematic and efficient way to select their structure. As for multilayer perceptrons, they have a very complicated structure and train extremely slowly.

Motivated by the success of classification and regression trees on difficult nonlinear and nonparametric problems, we propose the idea of a tree-structured piecewise linear adaptive filter. In the proposed method each node in a tree is associated with a linear filter restricted to a polygonal domain, and this is done in such a way that each pruned subtree is associated with a piecewise linear filter. A training sequence is used to adaptively update the filter coefficients and domains at each node, and to select the best pruned subtree and the corresponding piecewise linear filter.

The tree structured approach offers several advantages. First, it makes use of standard linear adaptive filtering techniques at each node to find the corresponding conditional linear filter. Second, it allows for efficient selection of the subtree and the corresponding piecewise linear filter of appropriate complexity. Overall, the approach is computationally efficient and conceptually simple.

The tree-structured piecewise linear adaptive filter bears some similarity to classification and regression trees. But it is actually quite different from a classification and regression tree. Here the terminal nodes are not just assigned a region and a class label or a regression value, but rather represent a linear filter with restricted domain. It is also different in that classification and regression trees are determined in a batch mode offline, whereas the tree-structured adaptive filter is determined recursively in real-time.

We first develop the specific structure of a tree-structured piecewise linear adaptive filter and derive a stochastic gradient-based training algorithm. We then carry out a rigorous convergence analysis of the proposed training algorithm for the tree-structured filter. Here we show the mean-square convergence of the adaptively trained tree-structured piecewise linear filter to the optimal tree-structured piecewise linear filter. Some new techniques are developed for analyzing stochastic gradient algorithms with fixed gains and (nonstandard) dependent data. Finally, numerical experiments are performed to show the computational and performance advantages of the tree-structured piecewise linear filter over linear and polynomial filters for equalization of high frequency channels with severe intersymbol interference, echo cancellation in telephone networks and predictive coding of speech signals.

CHAPTER 1
INTRODUCTION

1.1 Introduction

Many problems encountered in communications and signal processing involve removing noise and distortion due to physical processes that are unknown and/or time varying [1]. These types of processes represent some of the most difficult problems in transmitting and receiving information. Adaptive signal processing and specifically adaptive filtering offers an effective approach for removing noise and distortion in signals, as well as extracting information about the unknown physical process. Whenever there is a requirement to process signals that result from operation in an environment of unknown statistics, the use of an adaptive filter offers an attractive solution to the problem as it can usually provide a significant improvement in performance over the use of a fixed filter designed by conventional methods. In this chapter we introduce adaptive filters and discuss some of their applications.

1.2 Introduction to Adaptive Filters

We first discuss the general filtering problem and then examine the need for adaptive filters.

1.2.1 The Filtering Problem

The term "filter" is often used to describe a device in the form of a piece of physical hardware or computer software that is applied to a set of noisy data in order to extract information about a prescribed quantity of interest[2]. The noise may arise from a variety of sources. For example, the data may have been derived by means of noisy sensors or may represent a useful signal component that has been corrupted by transmission through a communication channel. In any event, we may use a filter to perform three basic information-processing operations :

1. "Filtering", where the aim is to derive information about the quantity of interest at time t based on data measured up to and including time t.
2. "Smoothing", where the aim is to derive information about the quantity of interest at time t based on data measured past time t.
3. "Prediction", where the aim is to derive information about the quantity of interest at time $t+\tau$ for some $\tau > 0$, based on data measured up to and including time t.

We say that the filter is "linear" if the filtered, smoothed, or predicted quantity of interest is a linear function of the noisy data or observations, as they are sometimes called.

In the classical statistical approach to the linear filtering problem, it is assumed that the joint statistics of the useful signal and unwanted noise are known, and the requirement is to design a filter so as to minimize the effects of noise according to some statistical criterion. A standard approach is to minimize the mean square value of the "error signal" which is defined as the difference between some desired response and the actual filter output. For stationary signals and observations, the resulting solution is commonly known as the Wiener filter [2].

1.2.2 Linear Adaptive Filters

In this section, we examine the need for adaptive filters, and for simplicity, we focus our attention on linear adaptive filters that have finite impulse response(FIR).

The design of a Wiener filter requires a-priori information about the statistics of the data to be processed. When this information is not completely known, it is not possible to design the optimum Wiener filter. This leads to the concept of adaptive filters. By such a device, we mean one that is self designing, in which the filter relies on a recursive algorithm to perform satisfactorily in unknown environments. Figure 1 shows the block diagram of a general adaptive filter. $x(k)$, $\hat{y}(k)$ and $y(k)$ denote the input, output and desired signals of the adaptive filter, respectively. In this section the programmable filter in Figure 1 is assumed to be a linear FIR filter, and hence the elements of the vector of coefficients $\underline{c}(k)$ represents the impulse response of such a filter at time instant k. The algorithm starts from a predetermined set of initial conditions representing complete ignorance about the environment. In stationary environments the algorithm should converge ideally to the optimal finite Wiener filter. In nonstationary environments, the algorithm offers a tracking capability, whereby it can track time variations in the statistics of the data, provided that the variations are sufficiently slow.

A wide variety of recursive algorithms have been developed in the literature for the training of adaptive filters. The choice of one algorithm over the other depends upon several factors such as computational complexity, rate of convergence, misadjustment, robustness, and structure[2].

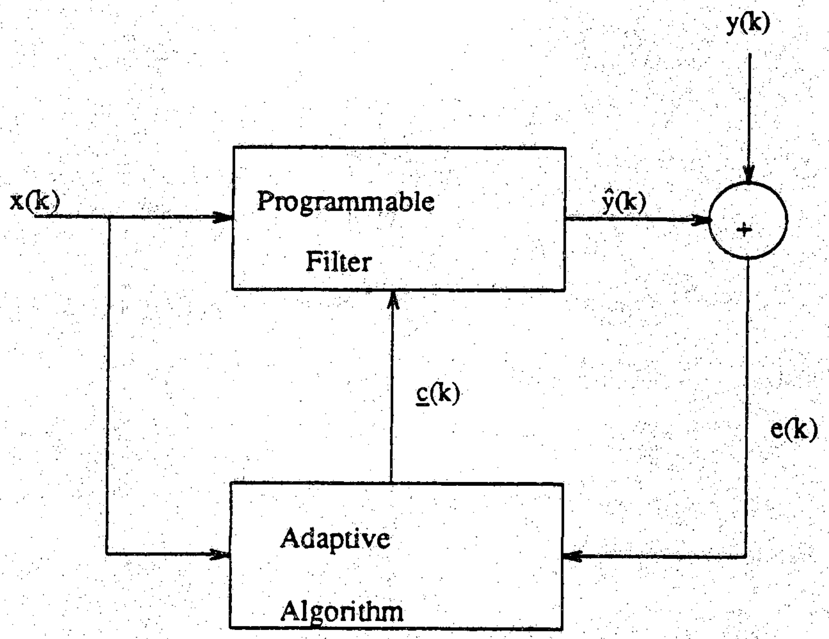


Figure 1. Block Diagram of a General Adaptive Filter

The simplest and most widely used recursive algorithm for training an adaptive filter is the Least Mean Square (LMS) algorithm[3] which is based on the steepest descent method for finding the Wiener filter. Here the finite tapped delay line or transversal filter shown in Figure 2 is used as the structural basis for development of the algorithm. The output of such a filter is given by

$$\hat{y}(k) = \sum_{j=0}^{L-1} c(j)\, x(k-j) = \underline{c}'\, \underline{x}(k)$$

where $\underline{c} = [c(0),\ldots,c(L-1)]'$ is the vector of tap weights and $\underline{x}(k) = [x(k),\ldots,x(k-L+1)]'$ is the input vector to the adaptive filter at time instant k. If x(k) and y(k) are jointly wide sense stationary sequences, then the mean square error (MSE)

$$\varepsilon = E\{(y(k) - \hat{y}(k))^2\}$$

is a constant convex function of $\underline{c}$, and if the joint second order statistics of x(k) and y(k) are known, then ε can be minimized over $\underline{c}$ analytically to yield the Wiener filter. When the statistics of x(k) and y(k) are unknown, the tap weight vector can be sequentially estimated based on a training sequence using a stochastic gradient algorithm :

$$\underline{c}(k+1) = \underline{c}(k) + \mu(y(k) - \hat{y}(k))\underline{x}(k) \tag{1.1}$$

where the gain parameter μ is a small positive number. Note that $\hat{y}(k)$ is evaluated at the current weight vector $\underline{c}(k)$ in the above recursion, i.e., $\hat{y}(k) = \underline{c}(k)'\, \underline{x}(k)$. This is the LMS algorithm. If x(k) and y(k) are in fact jointly wide sense stationary, it can be shown (under further suitable conditions) that $\underline{c}(k)$ converges to the Wiener filter in the mean and mean-square sense [2, 3].

Several other recursive algorithms have been developed based on the stochastic Newton method† (as opposed to the stochastic gradient method) and also the method of least squares[1, 2, 4, 5]. The algorithms based on these approaches are

† these algorithms are sometimes referred to as Kalman algorithms

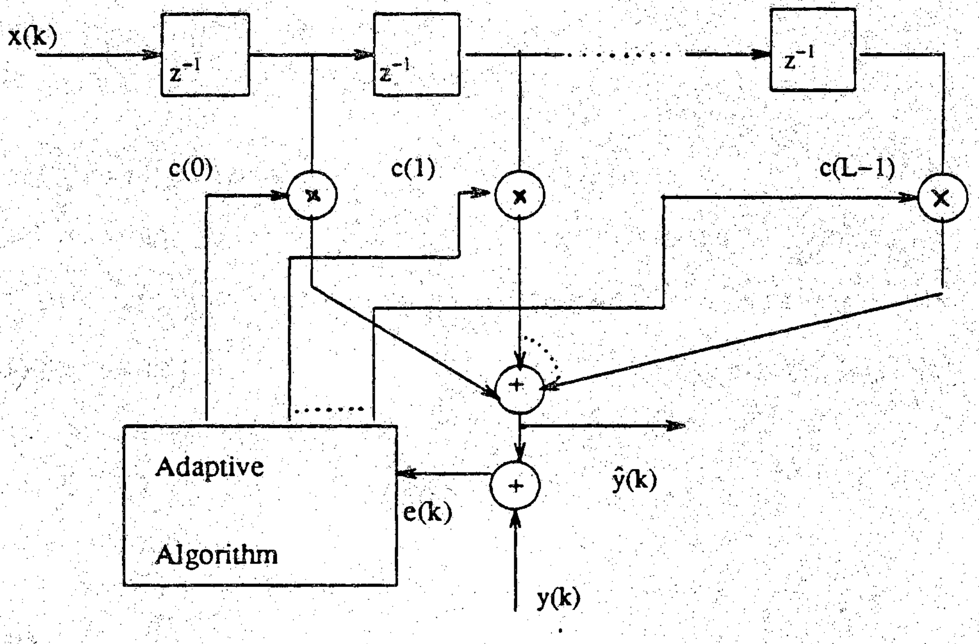


Figure 2. An Adaptive Transversal Filter

computationally expensive, but have a faster rate of convergence than the LMS algorithm. Hence these algorithms become useful whenever the rate of convergence is a critical factor.

1.3 Modes of Operation of Adaptive Filters

In this section we identify three basic modes of operation of adaptive filters [6]. We discuss these modes in a relatively general context, which means that the unknown systems need not be linear nor time invariant and the adaptive filters need not be linear.

The first mode of operation of an adaptive filter is the direct system modeling mode shown in Figure 3(a), which is used for system identification[3, 6]. In this mode, the adaptive filter is used to approximate the unknown system. Here, the time series x(k) is input simultaneously to the unknown system and the adaptive filter. The output of the unknown system y(k) then becomes the desired output for the adaptive filter. In practice, there is normally additive noise associated with the unknown system which could be measurement noise and/or noise within the system itself. The adaptive filter is designed to minimize the error between the output of the adaptive filter, $\hat{y}(k)$, and the output of the unknown system, y(k), in some statistical sense.

The second mode of operation of an adaptive filter is the inverse modeling mode shown in Figure 3(b), which is used for channel equalization[7]. In this mode the adaptive filter is used to approximate the inverse of the unknown system. Here the time series x(k) is the input to the unknown system. The output of the unknown system y(k) is the input to the adaptive filter, and the input to the unknown system x(k) is the desired output for the adaptive filter. Again, in practice there is normally additive noise present in the unknown system. The adaptive filter is designed to minimize the error between the output of the adaptive filter, $\hat{x}(k)$, and the input to the unknown system, $x(k)$, in some statistical sense.

The third mode of operation of an adaptive filter is the prediction mode shown in Figure 3(c), which is used in predictive coders for speech and images[8]. In this mode the adaptive filter is used to predict data at time instant k based on data observed upto time instant $k - T$, for some $T > 0$. Here, the time series $x(k)$ is delayed and input to the adaptive filter, and $x(k)$ is the desired output for the filter. Adaptivity is required here since the data $x(k)$ is usually nonstationary. Note that the predictive mode of operation is a special case of the inverse modeling mode where the unknown system in the inverse modeling mode of operation simply introduces a delay.

We note that when the adaptive filter is chosen to be linear and FIR, the LMS algorithm discussed in the previous section can be used in each mode of operation with obvious changes in notation.

1.4 Applications of Adaptive Filters

Due to the ability of the adaptive filter to operate effectively in unknown environments and also track the time variations of the input statistics, it has been successfully applied in such diverse fields as communications, control, radar, sonar, seismology, image processing and pattern recognition. In this section we briefly discuss three applications of adaptive filters drawn primarily from the field of communications, each representing a mode of operation described above.

Echo Cancellation : A practical example of the direct system modeling mode of operation is echo cancellation across the hybrid transformer used in telephone

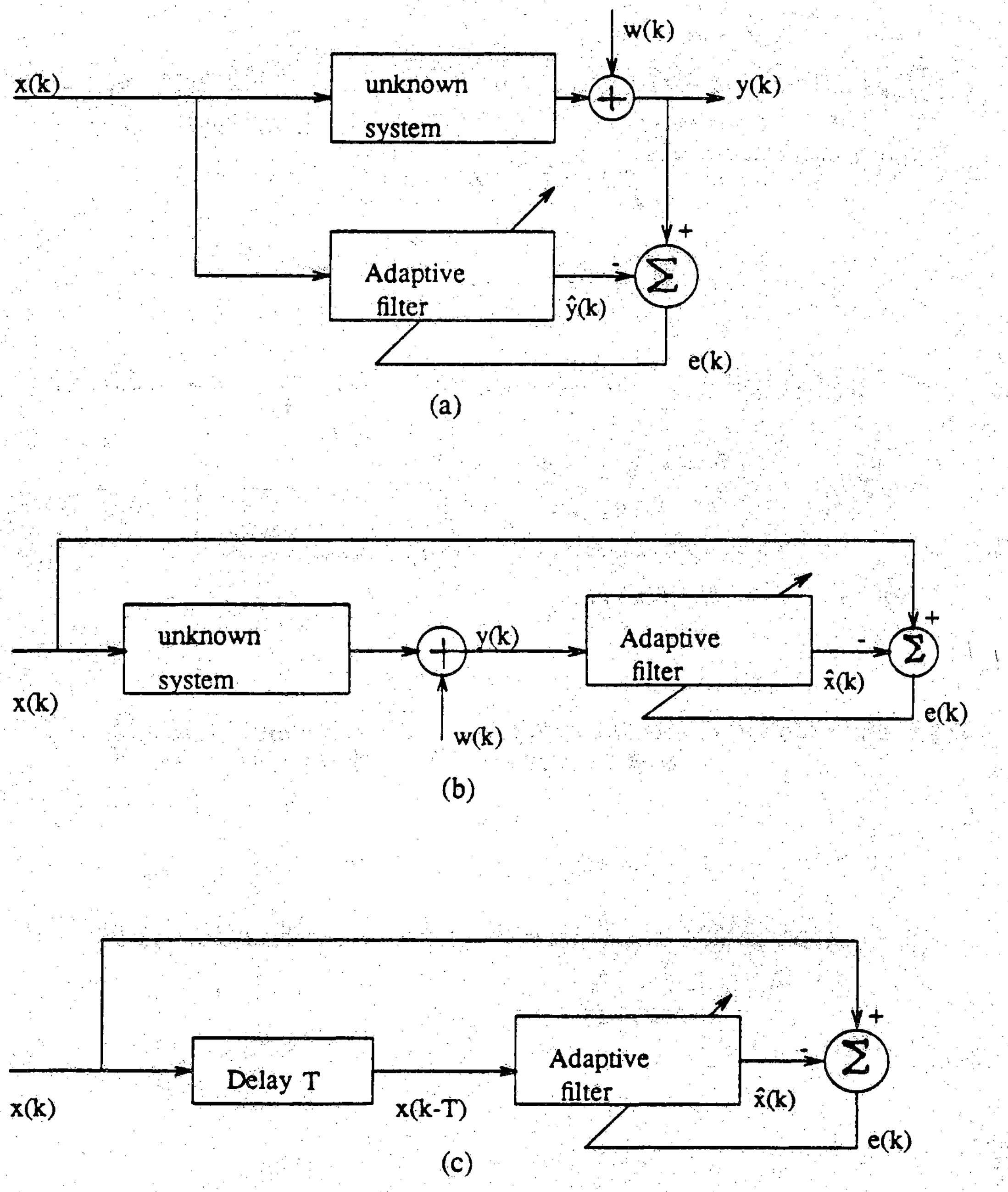


Figure 3. Modes of Operation of Adaptive Filters (a). Direct System Modeling (b). Inverse System Modeling (c). Prediction Mode

networks [9] (see Figure 4). At any point in a telephone network, when a signal encounters a mismatch in impedance, a portion of it gets reflected as an echo. This is annoying to the speaker and in many cases completely disrupts the conversation. The deleterious effects of such echoes depend upon their strength, spectral distortion and delay. The main source of echo in a telephone network is at the junction between the "two-wire" local subscriber loop and the "four-wire" long distance link called the hybrid. Whenever there is a mismatch of impedance between the subscriber loop circuit and long distance circuit, a portion of the signal from the transmitter reaches the receiver at the far end through the hybrid and this is called talker echo. One method of reducing the talker echo is to construct a filter in parallel with the hybrid which models the echo path across the hybrid. The echo can then be cancelled by subtracting the output of the adaptive filter from the output of the hybrid. Since the characteristics of the echo path is unknown a-priori and time varying, an adaptive filter is commonly employed.

Channel Equalization : A practical example of the inverse system modeling mode of operation is the equalization of digital communication channel to combat intersymbol interference[3, 6, 7] (see Figure 5). Such a channel may be modeled by an equivalent discrete-time transversal filter with additive noise[7]. The digital signal which is applied to the channel is a sequence of symbols taken randomly from a finite alphabet. If the equivalent discrete-time channel has finite bandwidth, each element of the channel output sequence will contain contributions from several input symbols. This is called intersymbol interference. The function of the adaptive filter is to reconstruct the transmitted symbol sequence with a very low probability of error. An adaptive equalizer is usually trained by transmitting a predetermined sequence known to the

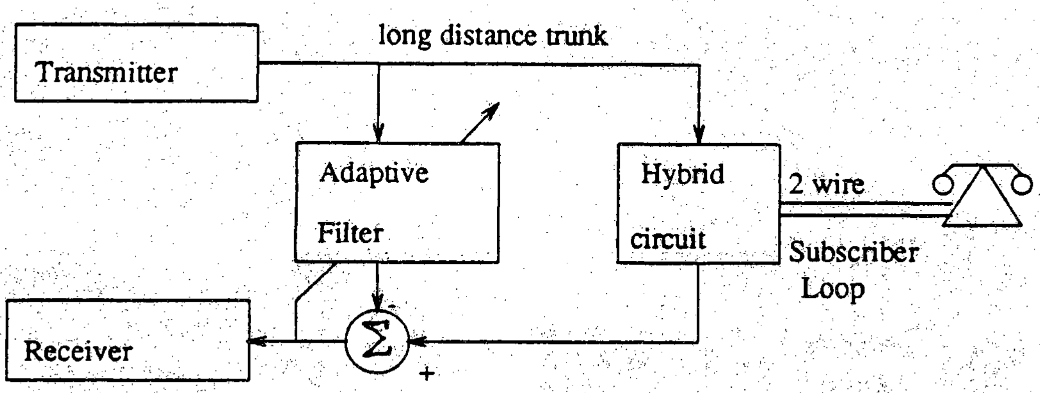


Figure 4. Echo Cancellation Application

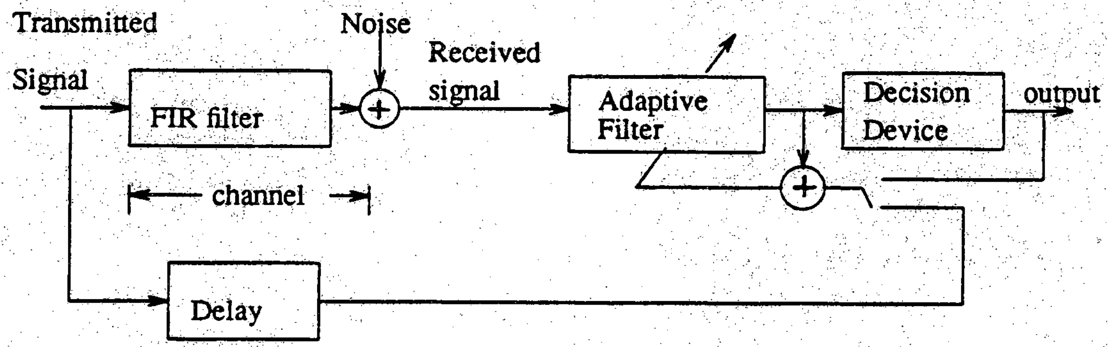


Figure 5. Channel Equalization Application

receiver, prior to the actual data. Subsequent to this training period, it is still possible to track slow variations in the channel characteristics by using the output of the decision circuit as a training sequence for the adaptive filter. This is known as a decision directed equalizer[7].

Predictive Coding : A practical application of the predictive mode of operation of an adaptive filter is in the area of waveform coding of speech [5,8] (see Figure 6). A prediction $\hat{x}(k)$ is made of the original process, $x(k)$, from which the prediction error signal $e(k) = x(k) - \hat{x}(k)$ is generated. A quantized version $e_q(k)$ of $e(k)$ is then encoded and transmitted. The speech decoder located at the receiver computes $\tilde{x}(k) = \hat{x}(k) + e_q(k)$ based upon the (assumed errorless) reception of $e_q(k)$. It is easy to show that if both encoder and decoder predictors have the same initial conditions, the only error in the reconstructed signal at the receiver is equal to the quantization error at the transmitter which can be made as small as possible. Since the error signal $e(k)$ has a very small dynamic range, the number of bits required to represent it is much less than if we had encoded the speech signal directly. Hence one can achieve a good compression ratio. Note that adaptivity in the predictor is necessary, since the speech signal is nonstationary.

Other applications of adaptive filtering in the field of communications include adaptive spectral estimation, adaptive line enhancer and adaptive beam forming[2].

In the next chapter we demonstrate the need for nonlinear adaptive filtering and state some of the problems associated with existing nonlinear adaptive filters. In Chapter 3 we propose a tree-structured piecewise linear adaptive filter to overcome the problems in existing nonlinear adaptive filters. In Chapter 4 we derive a stochastic gradient based training algorithm for the tree-structured filter. In Chapter 5 we carry

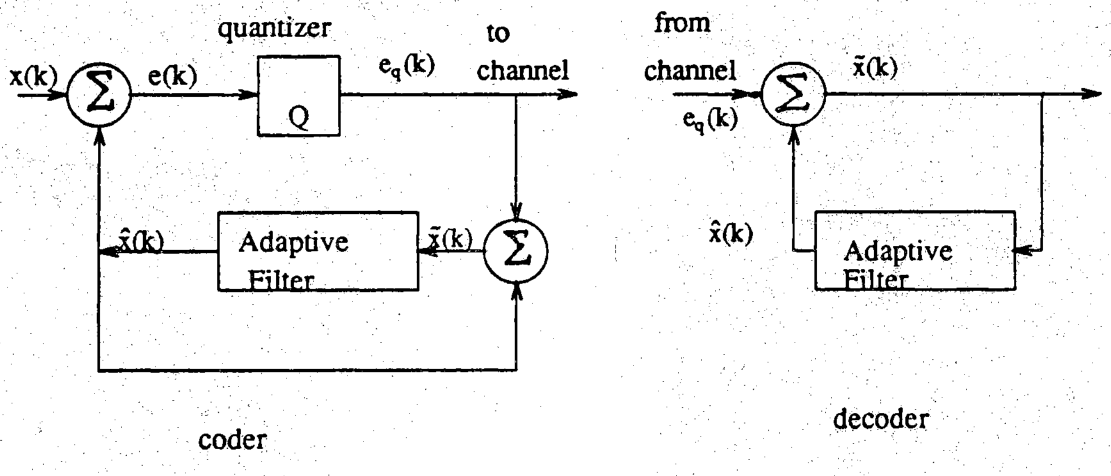


Figure 6. Speech Coding Application

out a rigorous convergence analysis of the tree-structured piecewise linear adaptive filter, wherein we show the mean square convergence of the adaptively trained tree structured piecewise linear filter to an optimal tree structured piecewise linear filter. In Chapter 6 we apply our tree-structured piecewise linear adaptive filter as an adaptive equalizer, adaptive echo canceler and adaptive predictor for speech signals. In each of the applications, we compare the performance of the tree-structured piecewise linear filter with that of linear, second order Volterra and third order Volterra types of adaptive filters. In Chapter 7 we investigate the problem of automatically inspecting the geometry of a fuel injector nozzle. Here we discuss the direct and indirect imaging techniques to accurately obtain the parameters that affect the spray process through the nozzle.

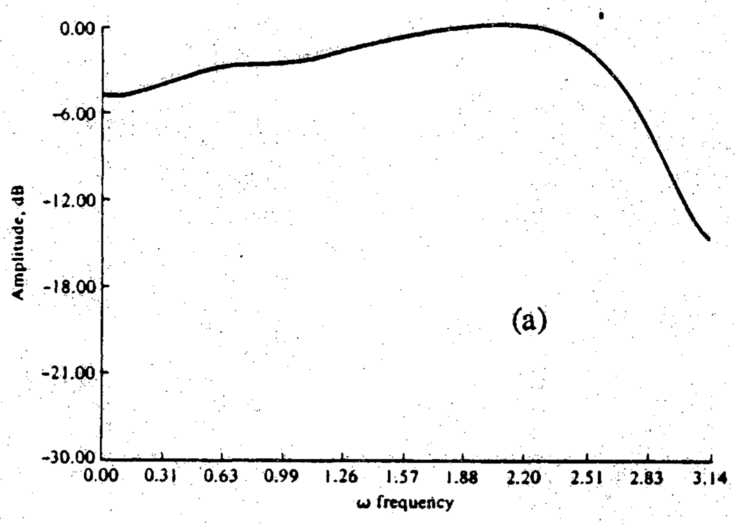
CHAPTER 2
NONLINEAR ADAPTIVE FILTERING

2.1 Need for Nonlinear Adaptive Filtering

An adaptive filter is referred to as linear if the estimate of the quantity of interest is obtained adaptively at the output as a linear combination of the available set of observations applied at the filter input [1]. An example of linear adaptive filter is the transversal filter shown in Figure 2 in Section 1.2. Although linear adaptive filters are simply implemented, their performance is inadequate in a variety of applications. Nonlinear adaptive filters have been used as equalizers[7] when the channel induces severe intersymbol interference. Nonlinear adaptive filters have been used to identify nonlinear systems which occur in noise and echo paths in telephone networks [10-13] and also digital satellite links where the satellite repeater uses nonlinear devices like a Travelling Wave Tube(TWT)[14]. Nonlinear adaptive filters have also been used for nonlinear prediction of speech in the area of speech coding. Such nonlinear systems are typically modelled by a truncated Volterra series or by a Wiener or Hammerstein model[13, 15-20]. Thus there is a general need for nonlinear adaptive filtering. In this section we illustrate in detail the need for nonlinear adaptive filtering in three applications, namely channel equalization, echo cancellation and speech coding. We also discuss some of the existing nonlinear adaptive filters and point out the problems associated with each of them. In chapter 3 we propose a tree-structured piecewise linear filter which overcomes the problems associated with the existing nonlinear adaptive filters.

The channel equalization application :- In Section 1.4 the use of an adaptive filter as a channel equalizer was discussed. In a bandwidth-efficient digital communication system, the effect of each symbol transmitted over a linear time dispersive channel (whose frequency response deviates from the ideal flat amplitude and linear phase) extends beyond the time interval used to represent that symbol. The distortion caused by the resulting overlap of received symbols is called "intersymbol interference" (ISI) [7,21]. ISI increases errors at the receiver and hence the reliability of high-speed data transmission over low background noise is reduced. An adaptive equalizer is used to compensate for the unknown time dispersion exhibited by the channel and to reconstruct the transmitted symbols[21-26].

For the equalization problem, it is common to assume that the channel is linear, bandlimited and corrupted by additive white Gaussian noise. For quality telephone channels, linear equalizers are sufficient to combat the correspondingly mild ISI. However, for multipath radio frequency channels which have deep nulls in their spectral characteristics, nonlinear equalizers are necessary to combat the comparatively severe ISI [7,27]. The conventional reasoning here is that, when the channel has deep nulls in its spectrum, a linear equalizer is too constrained to both invert the channel characteristics and at the same time moderate the noise in the vicinity of the dips in the channel response (and peaks in its inverse response). To illustrate the performance limitations of linear equalizers in presence of severe intersymbol interference, we show in Figure 7 the amplitude spectral characteristics of two different kinds of channels, and in Figure 8 the probability of error curves for a linear equalizer for these two kinds of channels. Figure 7(a) represents the spectrum



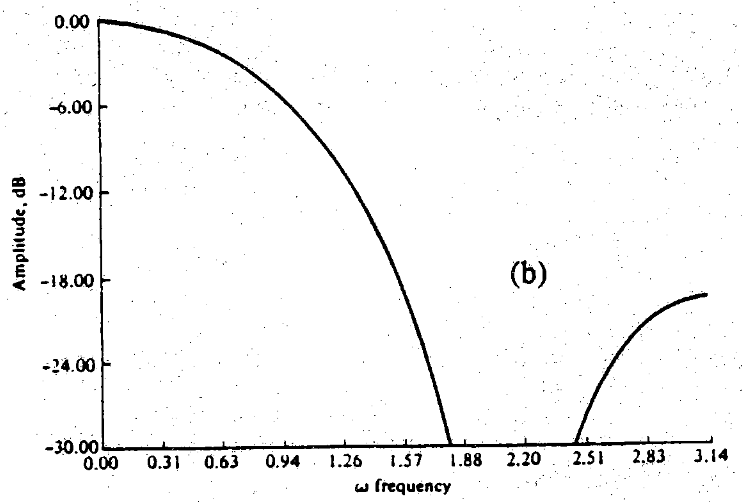


Figure 7. Amplitude Spectra of Two Different Channels (a) Typical Data-Quality Telephone Channel (b). Radio Frequency Channels with Deep Nulls

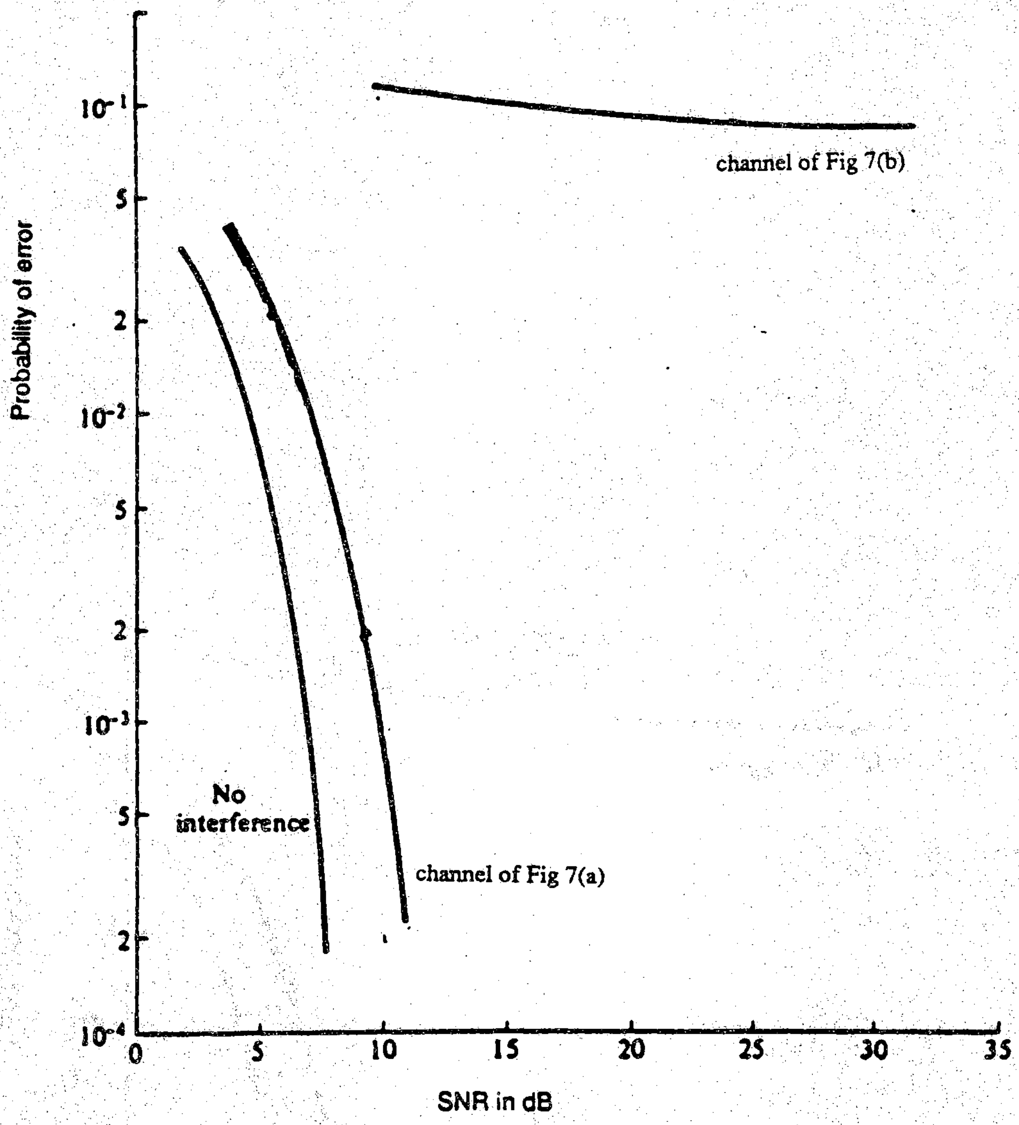


Figure 8. Performance Curves for a Linear Equalizer

of a typical data quality telephone channel, whereas Figure 7(b) represents spectrum of high frequency and radio frequency channels.

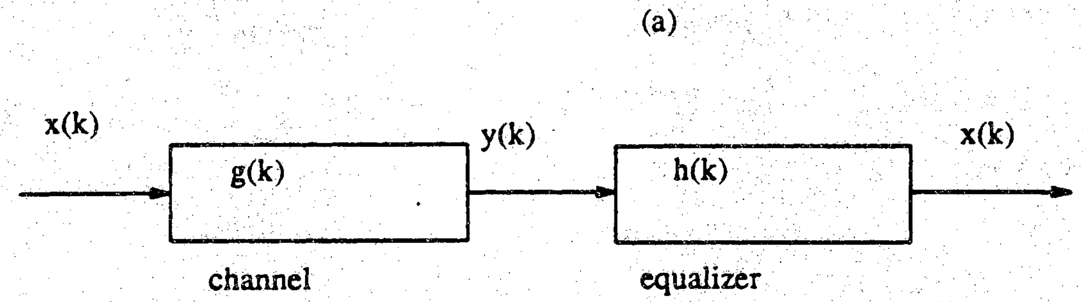
An alternative and more precise explanation of the need for a nonlinear equalizer can be based on the fact that the ultimate purpose of the equalizer is to minimize the probability of error. We shall give a simple argument that in the absence of noise, a linear equalizer of sufficiently large (but finite) length followed by a decision device can exactly reconstruct the sequence of transmitted symbols. The argument can be extended to show that in the presence of additive noise, the same receiver can estimate the sequence of transmitted symbols with arbitrarily small probability of error per symbol as the noise variance tends to zero. Note that these results hold for FIR or IIR channels; even when the channel is FIR and its inverse is IIR, only an FIR equalizer is required.

So let g(k) denote the impulse response for a noiseless channel and let G(z) be its z-transform. Let h(k) denote the impulse response of the inverse channel and let $H(z)=1/G(z)$ be its z-transform. We assume that g(k) and h(k) are BIBO stable. Of course we can reconstruct the information sequence x(k) from the channel output sequence y(k) using an (in general) IIR equalizer with impulse response h(k) (see Fig 9a). Suppose that x(k) takes on values ± 1. We argue that in this case x(k) can be reconstructed from y(k) using an FIR equalizer with some impulse response $\hat{h}(k)$ followed by a sign detector (see Fig 9b). To see this, fix L for the moment and let $\hat{h}(k) = h(k)$ for $k=-L+1,..,0,1,..,L-1$ and zero otherwise. Refering to Figure 9 we have

$$x(k) - \hat{x}(k) = \sum_{n=-\infty}^{-L} h(n)\, y(k-n) + \sum_{n=L}^{\infty} h(n)\, y(k-n)$$

Since x(k) is bounded and g(k) is stable, y(k) is bounded. Hence since h(k) is stable

$$\sup_k |x(k) - \hat{x}(k)| \le \sup_m |y(m)| \, \Big(\sum_{n=-\infty}^{-L} |h(n)| + \sum_{n=L}^{\infty} |h(n)| \Big) \to 0 \text{ as } L \to \infty,$$



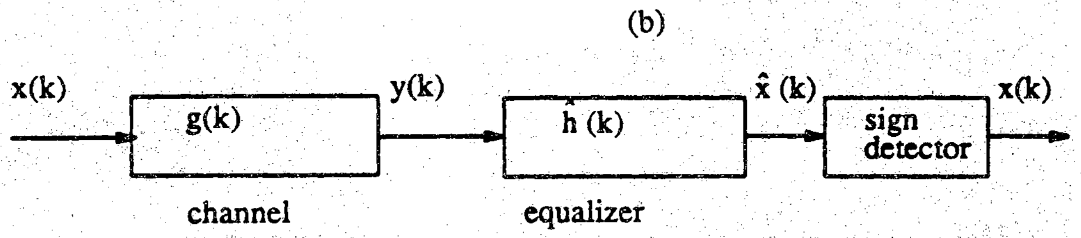


Figure 9. IIR and FIR Equalizers (a). Noiseless System with an IIR Equalizer (b). Noiseless System with FIR Equalizer and a Decision Device

uniformly for all input sequences x(k). Consequently, we can choose L large enough such that $\sup_k |x(k) - \hat{x}(k)| < 1$, and so $x(k) = \text{sign}(\hat{x}(k))$ for all k as required.

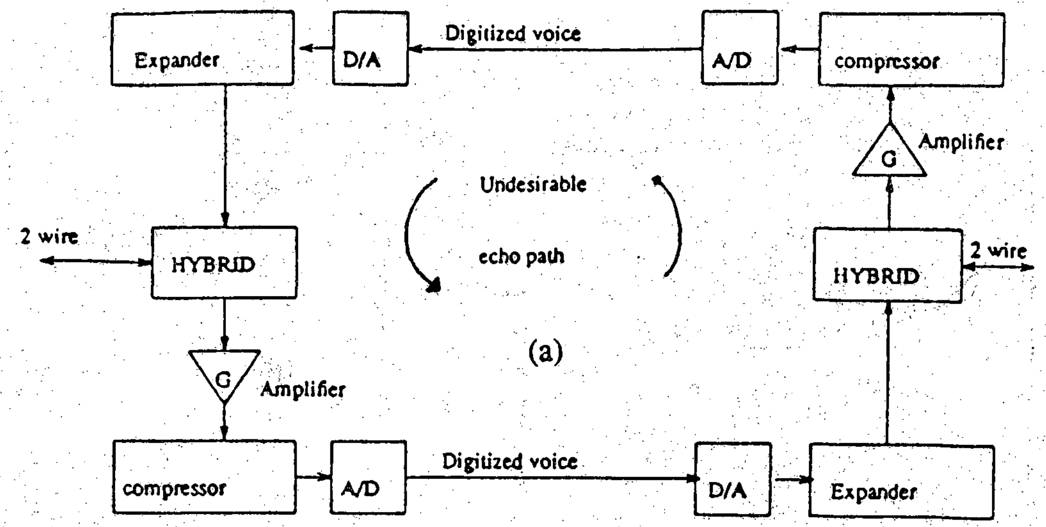
Now for the case of additive noise, a modification of this argument shows that for large enough filter length $\sup_k P\{x(k) \neq \text{sign}(\hat{x}(k))\} \rightarrow 0$ as the noise variance tends to zero. We remark that the length of the filter and the size of the noise variance that is required for the linear equalizer to perform well depends on how fast the tails of h(k) tend to zero. This provides a connection to the more heuristic argument that a linear equalizer performs well when $|G(e^{j\omega})|$ has no deep nulls : $|G(e^{j\omega})|$ has no large dips when $|H(e^{j\omega})|$ has no large peaks, which is essentially true when the tails of h(k) tend quickly to zero.

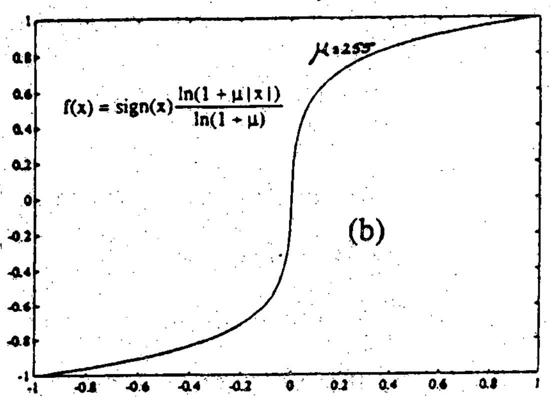
The above discussion provides some justification for using a linear equalizer provided we choose the filter length long enough and the noise is small enough. When either of these conditions is not satisfied, a nonlinear equalizer is more appropriate.

Echo cancellation application :- In Section 1.4 the use of an adaptive filter as an echo canceller was discussed. Echoes are generated as a consequence of impedance mismatch between the two-wire subscriber loop and four-wire long distance link at the hybrid junction of a telephone circuit. The basic idea is to synthesize a replica of the echo and subtract it from the actual echo generated in the telephone circuit. This is typically a system identification problem. Linear adaptive echo cancellers that have been proposed in the literature [9, 28, 29] are based on the assumption that the echo path in the telephone circuit is linear and all of them accept nonlinearity as an uncorrectable perturbation. Nonlinear echo paths arise in most practical situations due to two major reasons[13]. One is due to the mismatch in the compressor and expander characteristic and the other is due to the harmonic distortion in amplifiers and repeaters, the characteristics of both of these anomalies being unknown. In many cases, these anomalies are of sufficient magnitude to degrade the performance of linear adaptive echo cancellers. The causes for nonlinear echo paths are discussed in further detail below.

A highly simplified diagram of the interface between a four-wire digital line and a two wire analog line in a typical digital telephone network [30] is given in Figure 10(a). The compander and expander shown in the figure are used for the non-uniform quantization of speech signals[5]. This is necessary to improve the signal to quantization noise ratio (which is the ratio of signal power at the quantizer input and the quantization noise power at the output) of the system. A typical compression characteristic is shown in Figure 10(b). This is a highly nonlinear characteristic and is called the μ-law compression characteristic. At the transmitting end, the original speech signal is passed through a device with compression characteristic and then passed through an analog-to-digital(A/D) converter with a uniform quantizer to obtain the digitally encoded speech. At the receiving end, the output of the digital-to-analog converter(D/A) is passed through a device with expander characteristic which performs the inverse operation of compressor characteristic. The process of first compressing and then expanding a signal is referred to as "companding". A mismatch in the nonlinearities introduced by compressor and expander makes the echo path have a highly nonlinear characteristic.

Another source of nonlinearity in the echo path is the presence of automatic gain control devices which have characteristics similar to the one in Figure 10(b). A further source of nonlinearity in echo path is the presence of large signal amplifiers in hybrid circuits(see Figure 10(a)). In practice these amplifiers have a S-shaped gain curve with linearity in the middle of the curve and this produces harmonic distortion.





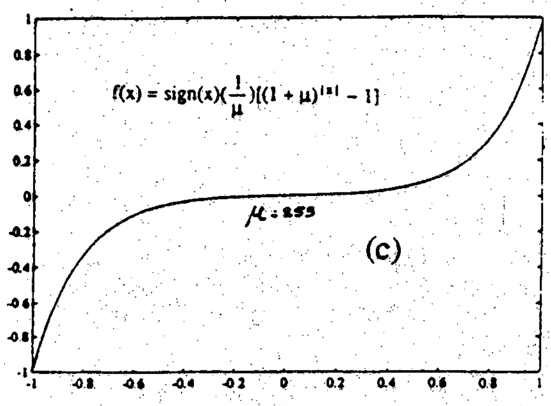


Figure 10. Four Wire Digital to Two Wire Analog Interface (a). Simplified Block Diagram (b). Compressor Characteristic (c). Expander Characteristic

This causes extremely annoying echoes to be heard by the talker.

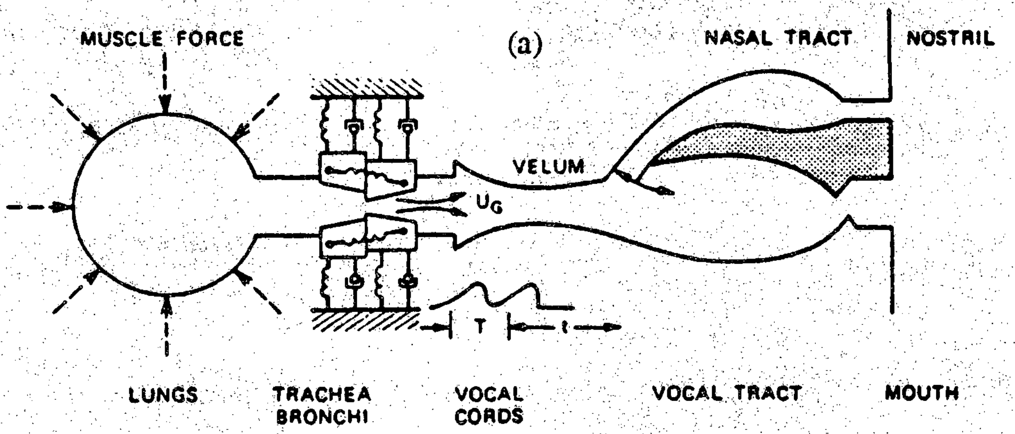
Thus nonlinear echo paths inherently exist in the current telephone network and there is a need to cancel the echoes generated by these nonlinear paths whose characteristics are not known a priori. Linear adaptive cancellers can perform poorly in such environments and hence the use of nonlinear adaptive filters is more appropriate.

Speech coding application :- For many classes of information signals, including speech, the value of the signal at a given instant is correlated with its values at the previous instants and hence represents redundant information [31]. The nature of the correlation depends on the manner in which the information signal was generated. An integral part of signal analysis is to determine the nature of the correlation and hence reduce the redundancy in the signal by representing it by a small number of parameters. This is especially useful in communication systems where the existing channel capacity cannot accommodate the digitized speech without redundancy removal. Thus, instead of transmitting the digitized speech, the parameters representing the speech are transmitted which requires considerably less channel capacity. One of the most popular analysis techniques for speech signals is the method of linear prediction analysis[32]. Over the last two decades, this method has become the predominant technique for estimating the basic speech parameters, e.g., pitch, formants, spectra, vocal tract area functions, and for representing speech for low bit rate transmission or storage.

The basic idea behind the linear predictive analysis is that a speech sample can be approximated by a *linear* combination of past speech samples. By minimizing a suitable cost function of the error (e.g., mean-squared error) between the actual speech sample and the linearly predicted one, a unique set of predictor coefficients can be determined which can be used to represent the parameters of the speech signal. This technique of representing the speech signal by a set of prediction coefficients or parameters is popularly known as Linear Predictive Coding or LPC [32-36]

Although LPC is well-established as an effective method for redundancy reduction in speech signals, its residual (defined as the error between the actual speech sample and the predicted output) still possesses a significant amount of redundancy and hence predictive component [37, 38]. This is because the LPC technique is based on the assumption that the speech production process can be modeled by an all-pole time varying linear filter. In reality, the speech production is inherently nonlinear and can best be represented by a nonlinear dynamical system [39, 40]. Furthermore, the LPC technique assumes that the sound source and the vocal tract are two decoupled linear systems. In fact, it is the interaction between the source and the vocal tract which actually contributes to the built-in naturalness in the produced speech. It has been established that this source-tract interaction is a major source of nonlinearity in the speech production process [36, 41, 42].

Many different models have been postulated for quantitatively describing certain factors involved in the speech production [38-45]. It can be stated with certainty that no single model has been developed which can account for all the observed characteristics of human speech. A highly simplified model of the vocal cord/vocal tract system is shown in Figure 11 [42]. A major source of nonlinearity in this simplified model is the presence of the nonlinear glottal impedance Z_g (representing the path of source-tract interaction) which depends upon the glottal flow and area A_g, which in turn depend upon the self-oscillating properties of the vocal cord model. Linear predictors can perform poorly in terms of removing redundancy in such



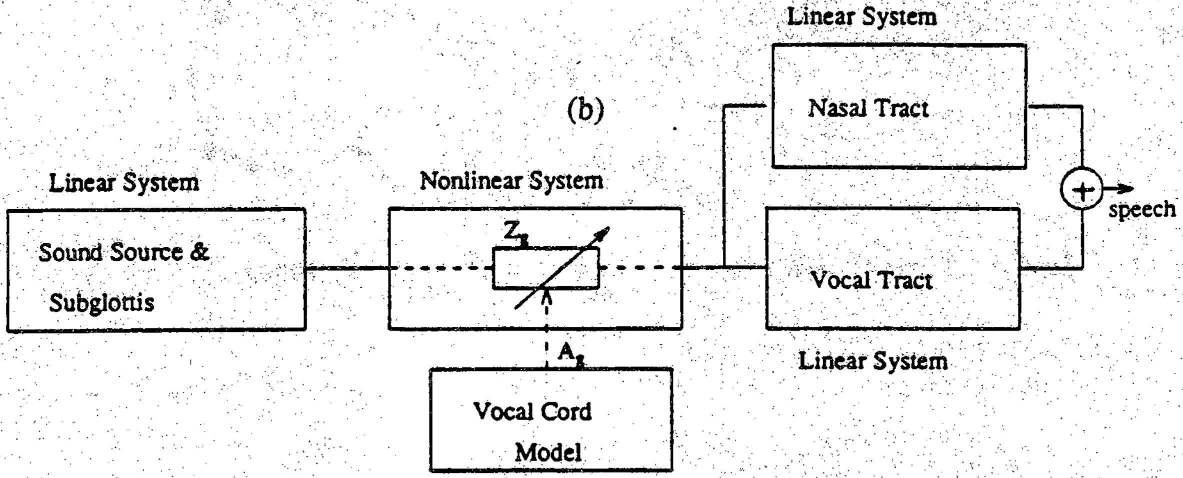


Figure 11. Vocal Cord/Vocal Tract System/Model (a). Schematic Diagram of Vocal Cord/Vocal Tract (b). Simplified Block Diagram of the Vocal Core/Vocal Tract Model

environments and hence nonlinear predictors are more appropriate. Furthermore, since the speech parameters vary from person to person and the speech signal is nonstationary, adaptive prediction is necessary.

2.2 Existing Nonlinear Adaptive Filters and Their Shortcomings

In general, we expect a nonlinear adaptive filter to have the following desirable properties: simple structure and training algorithm; good performance (in terms of asymptotic mean-square error or probability of error and also rate of convergence); and uses established linear adaptive filtering techniques to the extent possible.

Several nonlinear adaptive filters have been proposed in the literature in a variety of applications. In this section we focus on nonlinear adaptive filters that have been used for equalization, echo cancellation and speech coding.

The decision feedback equalizer (DFE) [6,7,46,47] is by far the most popular nonlinear adaptive equalizer. The DFE consists of two sections, a feedforward section and a feedback section. Both sections have structures similar to the linear transversal filter shown in Figure 2. The input to the feedforward section is the channel output sequence and the input to the feedback section is the sequence of previously detected symbols. The output of the DFE is the sum of the outputs of these two sections. Functionally the feedforward section is the same as that of linear transversal equalizer and the feedback section is used to cancel the ISI caused by previously transmitted symbols. The idea behind the decision feedback equalizer is that if the previously detected symbols are correct, then the contribution of these symbols to the ISI of the future arriving samples can be exactly cancelled. The output of a decision feedback equalizer can be expressed as

$$\hat{y}(k) = \sum_{j=-L+1}^{0} c(j)\, x(k-j) + \sum_{j=1}^{L-1} c(j)\, \tilde{y}(k-j) = \underline{c}'\, \underline{\tilde{z}}(k)$$

where $\underline{\tilde{z}} = [x(k+L-1),...,x(k),\tilde{y}(k-1),...,\tilde{y}(k-L+1)]'$ and $\underline{c} = [c(-L+1),...,c(L-1)]'$. Here $\tilde{y}(k)$ denotes the detected output at time instant k. The weight vector $\underline{c}$ can be sequentially estimated based on a training sequence using the LMS algorithm analogously to (1.1) :

$$\underline{c}(k+1) = \underline{c}(k) + \mu(y(k) - \hat{y}(k))\underline{\tilde{z}}(k) \tag{2.1}$$

Note that $\hat{y}(k)$ is evaluated at the current weight vector $\underline{c}(k)$ in (2.1), i.e., $\hat{y}(k) = \underline{c}'(k)\, \underline{\tilde{z}}(k)$. In the training mode, $\underline{\tilde{z}} = [x(k+L-1),...,x(k),\tilde{y}(k-1),...,\tilde{y}(k-L+1)]'$ can be replaced by $\underline{z} = [x(k+L-1),...,x(k),y(k-1),...,y(k-L+1)]'$ in (2.1). In the fully adaptive mode y(k) is replaced by $\tilde{y}$ (k) in (2.1). Although the decision feedback equalizer is widely used, it suffers from two significant problems. First, at low signal to noise ratios(SNR), the decision feedback equalizer can feed back sufficiently many incorrectly detected symbols so as to seriously degrade its performance. Second, the decision feedback equalizer is limited in its ability to implement complicated nonlinearities; in fact the decision feedback equalizer only differs from a linear equalizer with feedback in that the feedback filter contains the previously detected symbols which are quantized versions of the output of the equalizer.

Polynomial filters [18, 48] which include linear and quadratic filters as special cases, can also be used as nonlinear adaptive equalizers. Polynomial filters can be viewed as linear filters with an extended input space. The output of the polynomial equalizer can be expressed as

$$\hat{y}(k) = \sum_{p=1}^{q} \sum_{-L+1 \le j_1 \le \cdots \le j_p \le L-1} c(j_1, \ldots, j_p)x(k-j_1) \cdots x(k-j_p) = \underline{C}'\underline{X}(k)$$

where $\underline{X}(k)$ is a column vector of inputs $x(k-j_1) \cdots x(k-j_p)$ and $\underline{C}$ is a column vector of weights $c(j_1, \ldots, j_p)$ (put another way, $\hat{y}(k)$ is a q-th order truncated Volterra series in the x(k)'s). The weight vector $\underline{C}$ can be sequentially estimated based on a training sequence analogously to (1.1) :

$$\underline{C}(k+1) = \underline{C}(k) + \mu(y(k) - \hat{y}(k))\underline{X}(k) \tag{2.2}$$

Note that $\hat{y}(k)$ is evaluated at the current weight vector $\underline{C}(k)$ in (2.2), i.e., $\hat{y}(k) = \underline{C}'(k)\,\underline{X}(k)$. In the fully adaptive mode x(k) is replaced by $\tilde{y}(k)$ in (2.2). Apriori, a polynomial equalizer seems like an obvious choice for a nonlinear equalizer as it appears to possess the desired properties listed above. However, it turns out that although sufficiently high order polynomials can yield small asymptotic probability of error they will also in general converge very slowly. The properties of polynomial filters are further discussed in Chapter 6.

Multilayer perceptrons [49] have also been proposed to be used as an adaptive equalizer, but they require enormous amounts of training and have a complicated structure.

Almost all the nonlinear adaptive filtering approaches to echo cancellation that have been proposed in the literature make use of the discrete Volterra series representation to model the nonlinear echo path. Note that truncated Volterra filters are identical to the polynomial filters mentioned above. This approach has two problems. First, it is not clear which higher order terms can properly be neglected, and second, the convergence rate may be extremely slow. In [18], a Kalman filtering approach to a second order adaptive Volterra filter is proposed. This approach provides a faster rate of convergence, but is computationally expensive. In [10, 12], echo cancelers based on the principle of "memory cancellation" or "look-up table" have been proposed. These approaches do not account for possible time variations in the echo path response.

The nonlinear adaptive filters that have been proposed as adaptive predictors in the literature are the discrete Volterra filters and the multilayer perceptron architectures[38, 50]. The problems associated with these kinds of nonlinear adaptive filters have been discussed above.

In the next chapter we propose a tree structured piecewise linear filter which overcome the problems associated with the nonlinear adaptive filters discussed above.

CHAPTER 3
TREE-STRUCTURED PIECEWISE LINEAR FILTER

A simple and logical alternative to the nonlinear adaptive filters described in Chapter 2 is to build a piecewise linear adaptive filter such that the overall response of the filter approximates the optimal nonlinear response. Figure 12 shows a piecewise linear filter characteristic with a scalar input. The filter we propose is adaptive in the sense that the number, length and slope of the linear segments get updated as new samples arrive.

A piecewise linear filter divides the input space into different regions and selects the best linear filter for those inputs belonging to a particular region. Thus the overall structure of a piecewise linear filter is of the form shown in Figure 13 and may be described mathematically as follows. Let χ denote the input space $\mathbb{R}^L$. Let $\chi_1, \chi_2, \ldots, \chi_N$ denote a partition of the input space into N disjoint regions i.e., $\chi = \bigcup_{i=1}^{N} \chi_i$; $\chi_i \cap \chi_j = \varnothing \ \forall \ i \neq j$ where $i,j = 1,2,\ldots,N$. Let $h_1(k), \ldots, h_N(k)$ denote the impulse responses of N FIR filters each of length L. If the input to the piecewise linear filter at the k-th time instant is x(k) then the output of the i-th filter at the k-th instant is given by

$$\hat{y}_i(k) = \sum_{j=0}^{L-1} h_i(j)\, x(k-j).$$

The overall output of the piecewise linear filter at the k-th instant is given by

$$\hat{y}(k) = \sum_{i=1}^{N} \hat{y}_i(k)\, I(\underline{x}(k) \in \chi_i),$$

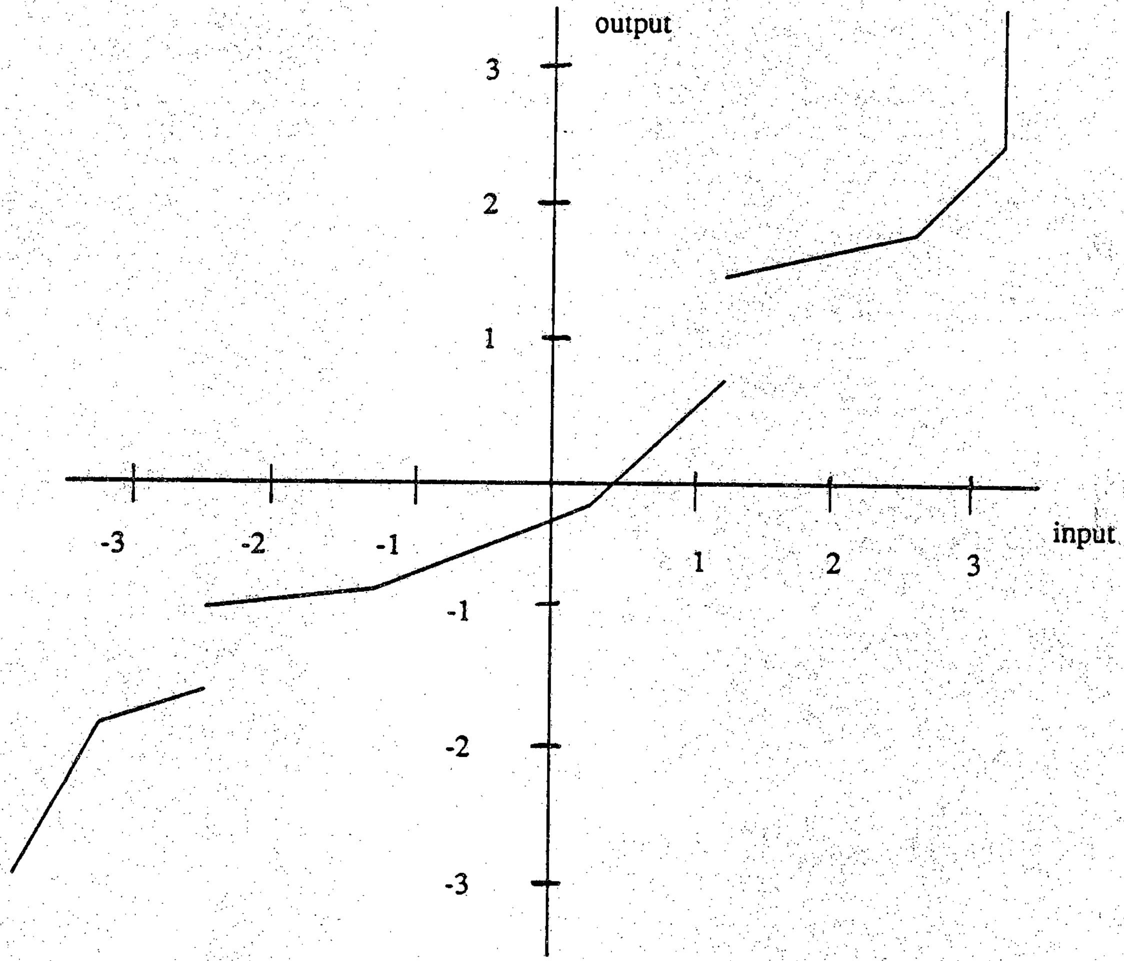


Figure 12. A Piecewise Linear Input Output Characteristic

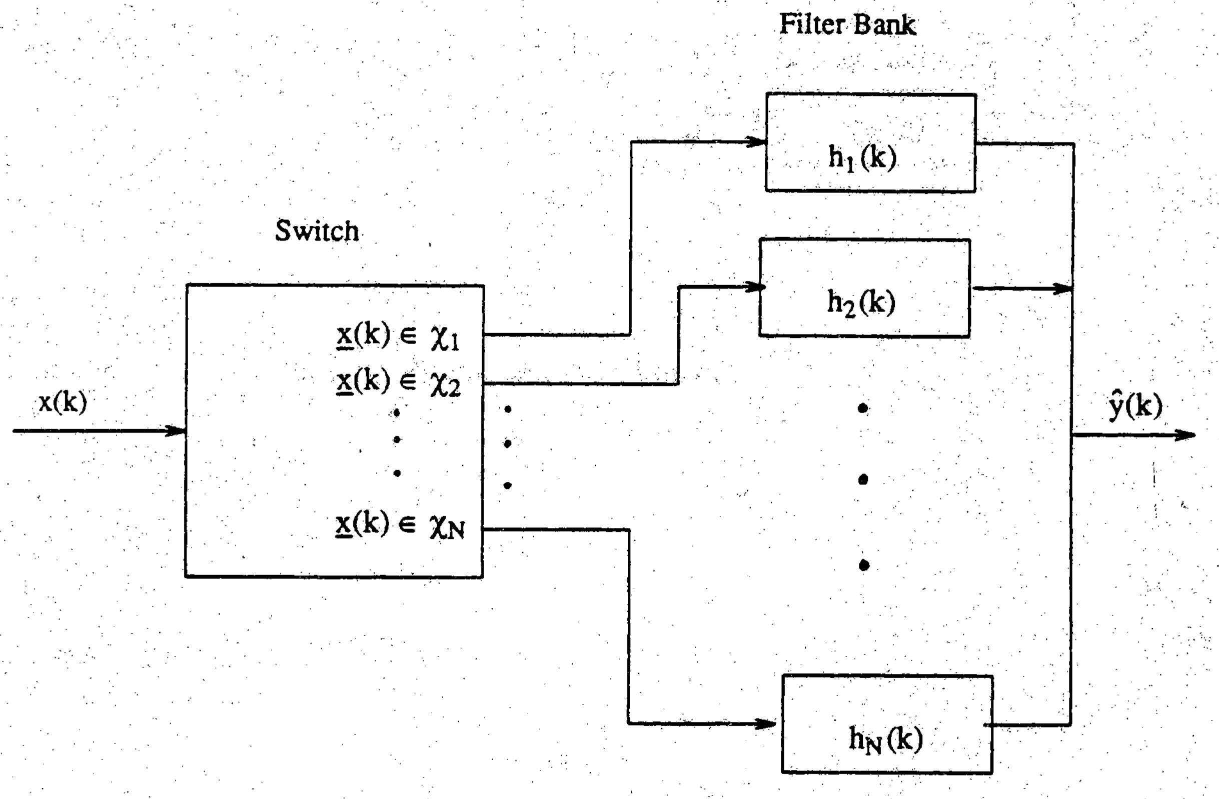

Filter Bank
Switch
x(k)
$\underline{x}(k) \in \chi_1$
$\underline{x}(k) \in \chi_2$
$\underline{x}(k) \in \chi_N$
$h_1(k)$
$h_2(k)$
$h_N(k)$
$\hat{y}(k)$

Figure 13. Structure of a Piecewise Linear Filter

where $\underline{x}(k) = [x(k),...,x(k-L+1)]'$ and $I(\cdot)$ is an indicator function.

In the adaptive version of the piecewise linear filter, the domains χ_i, the filters $h_i(k)$, and the number of domains and filters N keep changing as new samples arrive. In order to obtain the adaptation in an efficient manner, we adopted a tree-structured approach. We next describe in detail what a tree-structured piecewise linear filter is. We start by briefly discussing some basic definitions and terminologies associated with a binary tree graph (see [51] for more details about tree graphs).

A (binary) tree can be defined as a finite non-empty set T of positive integers and two functions left($\cdot$) and right($\cdot$) from T to $T \cup \{0\}$, which together satisfy the following two properties :

(i) For each $t \in T$, either left(t) = right(t) = 0 or left(t) > t and right(t) > t;

(ii) For each $t \in T$, other than the smallest integer in T, there is exactly one $s \in T$ such that either t = left(s) or t = right(s).

For each node $t \in T$, left(t) and right(t) simply indicate the left and right nodes which branch out from t (or are both zero if t is a terminal node).

The minimum element of a tree T is called the root of T, denoted by root(T). If $s, t \in T$ and t = left(s) or t = right(s), then s is called the parent of t and t is called an offspring of s. A node s is called an ancestor of a node t if s = parent(t) or s = parent(parent(t)) or $\cdots$. A node t is called a descendant of a node s if s is an ancestor of t. The depth of a node t is the number of ancestors of t. A node t is called a terminal node if it is not a parent, that is, if left(t) = right(t) = 0. Let $\tilde{T}$ denote the collection of terminal nodes of T. The elements in $T - \tilde{T}$ are called nonterminal nodes.

For a nonempty subset T_1 of T define $\mathrm{left}_1(\cdot)$ and $\mathrm{right}_1(\cdot)$ from T_1 to $T_1 \cup \{0\}$ by $\mathrm{left}_1(t) = \mathrm{left}(t)$ if $\mathrm{left}(t) \in T_1$, $\mathrm{right}_1(t) = \mathrm{right}(t)$ if $\mathrm{right}(t) \in T_1$ and $\mathrm{left}_1(t) = \mathrm{right}_1(t) = 0$ otherwise. T_1 is called a subtree of T if the triple T_1, $\mathrm{left}_1(\cdot)$, $\mathrm{right}_1(\cdot)$ forms a tree. Given $t \in T$, the collection T_t consisting of t and all its descendents is called the branch of T stemming from t. It is a subtree of T.

A subtree T_1 of T is called a pruned subtree of T if $\mathrm{root}(T_1) = \mathrm{root}(T)$, i.e., if T_1 and T share the same root node; this is denoted by $T_1 \leq T$. It is seen that $\leq$ defines a partial order on the pruned subtrees of a tree. A tree, a subtree and a pruned subtree are illustrated in Figure 14.

To construct a tree-structured piecewise linear filter (or more precisely a family of piecewise linear filters) we start with a fixed binary tree T_o, and for each node $t \in T_o$ we specify a tap weight vector $\underline{c_t} = [c_t(0), \ldots, c_t(L-1)]'$, an offset d_t and a threshold θ_t (actually, there is no threshold θ_t associated with terminal nodes $t \in \tilde{T}_o$, and in many problems there is no offset d_t for $t = \mathrm{root}(T_o)$). We then associate a piecewise linear filter with each pruned subtree $T \leq T_o$ as follows. Let $\underline{x}$ be an input vector, i.e., $\underline{x} \in \chi = \mathbb{R}^L$. Then each node $t \in T_o$ is associated with the linear filter

$$\hat{y}_t = \underline{c_t}' \, \underline{x} + d_t,$$

and each pruned subtree $T \leq T_o$ is associated with the piecewise linear filter

$$\hat{y}_T = \hat{y}_{t_*},$$

where t_* is the terminal node in T obtained by starting at the root node and using the rule

$$\hat{y}_t > \theta_t \quad \text{go to right(t)}$$

$$\hat{y}_t \leq \theta_t \quad \text{go to left(t)}$$

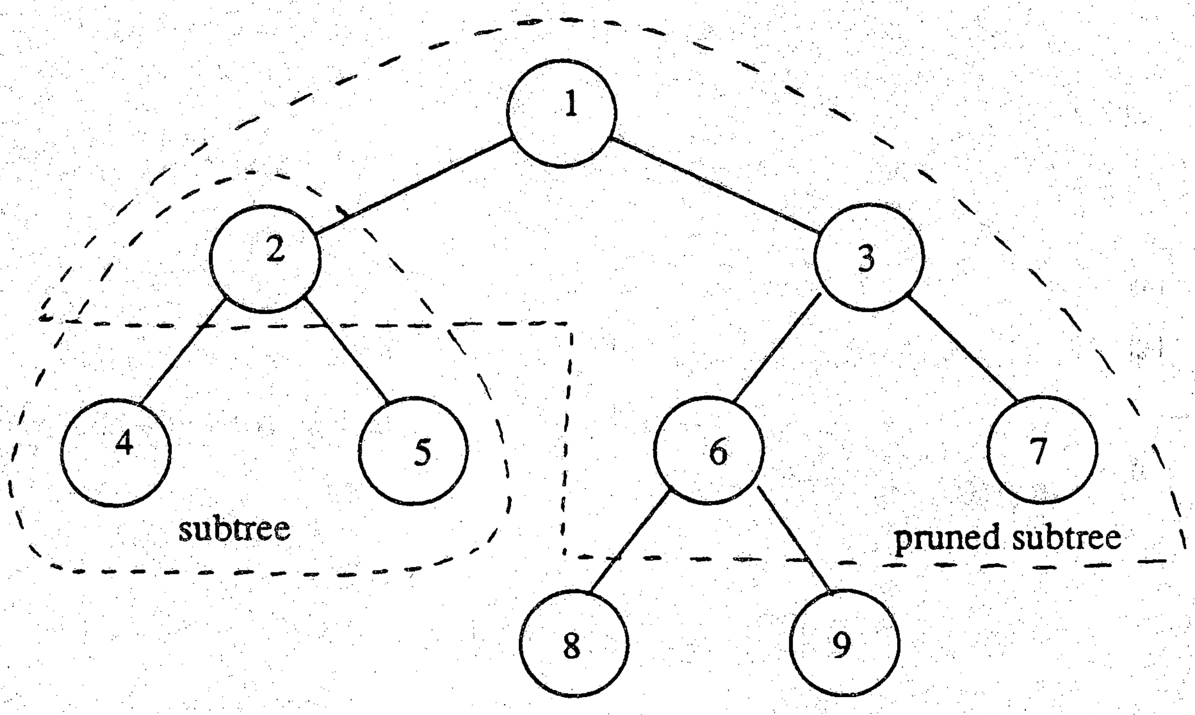


Figure 14. A Binary Tree, Subtree, Pruned Subtree

Observe that this procedure implicitly specifies the (polygonal) domain χ_t of the linear filter $\hat{y}_t$ at node t. We can write

$$\hat{y}_T = \hat{y}_t \quad \text{if } \underline{x} \in \chi_t \text{ and } t \in \tilde{T},$$

i.e., $\hat{y}_T$ is just the output of the linear filter at node t if t is a terminal node of T and the input vector $\underline{x}$ lands in t.

A basic tree T_o along with tap weights $\underline{c}_t$, offsets d_t, and thresholds θ_t for a family of tree-structured piecewise linear filters of length 2 is illustrated in Figure 15. The family of pruned subtrees generated by T_o and the associated partitioning of input space is shown in Figure 16. Note that there are three piecewise linear filters corresponding to the three possible pruned subtrees, namely T={1}, T={1,2,3} and T={1,2,3,4,5}. The root node pruned subtree T={1} corresponds to a linear filter :

$$\hat{y}_{\{1\}} = \hat{y}_1$$

The pruned subtree T={1,2,3} with terminal nodes $\tilde{T} = \{2,3\}$ corresponds to a piecewise linear filter comprised of two linear filters restricted to polygonal domains (one filter for each of the terminal nodes) :

$$\begin{aligned} \hat{y}_{\{1,2,3\}} &= \hat{y}_2 \quad \text{if } \underline{x} \in \chi_2 \\ &= \hat{y}_3 \quad \text{if } \underline{x} \in \chi_3 \end{aligned}$$

Here the domains χ_2 and χ_3 are given by $\chi_2 = \{\underline{x} : \hat{y}_1 \le -0.03\}$ and $\chi_3 = \{\underline{x} : \hat{y}_1 > -0.03\}$. The pruned subtree T={1,2,3,4,5} with terminal nodes $\tilde{T} = \{2,4,5\}$ corresponds to a piecewise linear filter comprised of three linear filters restricted to polygonal domains (again, one filter for each of the terminal nodes) :

$$\begin{aligned} \hat{y}_{\{1,2,3,4,5\}} &= \hat{y}_2 \quad \text{if } \underline{x} \in \chi_2 \\ &= \hat{y}_4 \quad \text{if } \underline{x} \in \chi_4 \end{aligned}$$

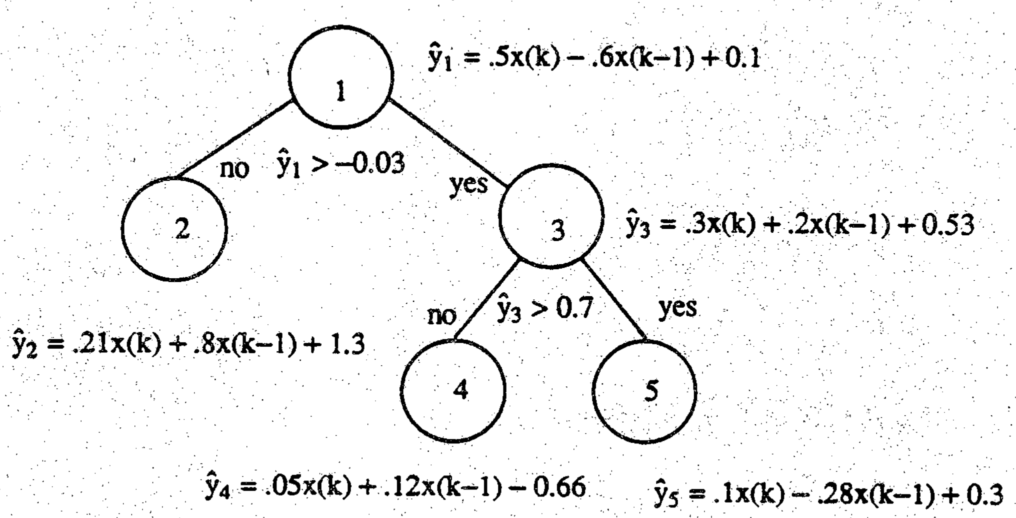

1
2
3
4
5
$\hat{y}_1 = .5x(k) - .6x(k-1) + 0.1$
no
$\hat{y}_1 > -0.03$
yes
$\hat{y}_3 = .3x(k) + .2x(k-1) + 0.53$
no
$\hat{y}_3 > 0.7$
yes
$\hat{y}_2 = .21x(k) + .8x(k-1) + 1.3$
$\hat{y}_4 = .05x(k) + .12x(k-1) - 0.66$
$\hat{y}_5 = .1x(k) - .28x(k-1) + 0.3$

Figure 15. A Tree-Structured Piecewise Linear Filter

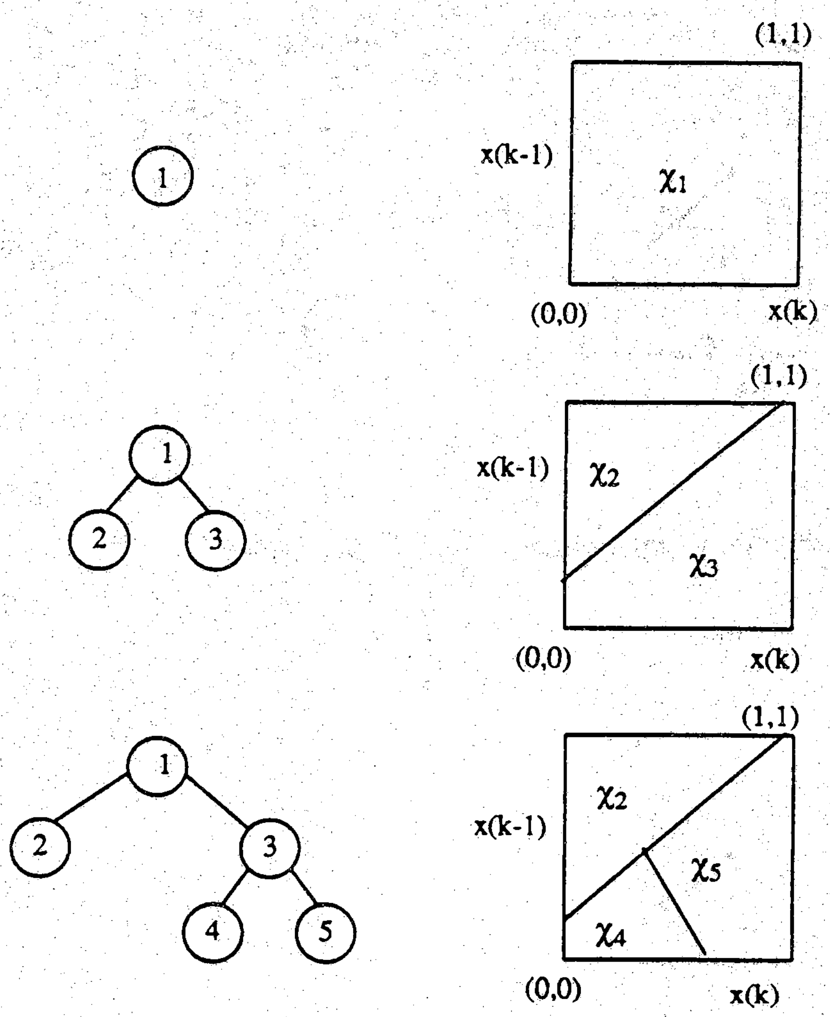


Figure 16. Pruned Subtrees and Associated Partitioning of Input Space

$$= \hat{y}_5 \text{ if } \underline{x} \in \chi_5$$

Here the domains χ_2, χ_4 and χ_5 are given by $\chi_2 = \{\underline{x} : \hat{y}_1 \leq -0.03\}$, $\chi_4 = \{\underline{x} : \hat{y}_1 > -0.03, \hat{y}_3 \leq 0.7\}$ and $\chi_5 = \{\underline{x} : \hat{y}_1 > -0.03, \hat{y}_3 > 0.7\}$. In general, it is seen that for any given pruned subtree T of T_o, the corresponding piecewise linear filter is determined by the linear filters $\hat{y}_t$ and polygonal domains χ_t at the terminal nodes $t \in T$. Furthermore the linear filter $\hat{y}_t$ is determined by the weights $\underline{c}_t$ and offsets d_t at the node t, whereas the domain χ_t is determined by the weights $\underline{c}_s$, offsets d_s and thresholds θ_s at the ancestor nodes s of node t.

In the above description of a tree-structured piecewise linear filter we specified two parameters d_t and θ_t at each node which we referred to as the offset and threshold, respectively. It is an important observation that we cannot combine these parameters into a single parameter without restricting the structure of the piecewise linear filters corresponding to the pruned subtrees. The point here is that the basic tree T_o along with tap weights $\underline{c}_t$, offsets d_t and thresholds θ_t actually corresponds to a family of piecewise linear filters, one for each pruned subtree T of T_o. The idea is to select the right-sized pruned subtree so as to avoid overfitting the data (this is analogous to selecting the right number of linear segments in Figure 12); a major advantage of the tree-structured approach is that the right-sized pruned subtree can be selected efficiently (see Chapter 4). In order to allow enough flexibility to determine appropriate values for $\underline{c}_t$, d_t and θ_t without knowing whether the node t will eventually be selected as a terminal node, we allow both d_t and θ_t to be nonzero. We proceed by choosing $\underline{c}_t$ and d_t to yield an appropriate filter assuming t will be selected as a terminal node, and then choose θ_t to yield an appropriate split assuming t will be selected as a nonterminal node; details are discussed in Chapter 4.

CHAPTER 4
TRAINING A TREE-STRUCTURED PIECEWISE LINEAR FILTER

In the previous chapter we discussed the structure of a tree-structured piecewise linear filter. In this chapter we specify training algorithms for adaptively updating the tap weight vectors, offsets and thresholds, and selecting a pruned subtree and the corresponding piecewise linear filter.

Recall that we are considering a family of piecewise linear filters which are in one-to-one correspondence with the pruned subtrees of a basic binary tree T_o. Let $\underline{x}$ be the input vector to the filter. Using the notation introduced in Chapter 3, each node $t \in T_o$ is associated with a linear filter

$$\hat{y}_t = \underline{c_t}'\underline{x} + d_t ,$$

and each pruned subtree $T \leq T_o$ is associated with a piecewise linear filter

$$\hat{y}_T = \hat{y}_{t_*} ,$$

where t_* is the terminal node in T which is arrived at starting from the root node and using the rule

$$\begin{aligned} \hat{y}_t > \theta_t &\quad \text{go to right(t)} \\ \hat{y}_t \leq \theta_t &\quad \text{go to left(t)} . \end{aligned} \tag{4.1}$$

Also recall that we write

$$\hat{y}_T = \hat{y}_t \quad \text{if } t \in \tilde{T} \text{ and } \underline{x} \in \chi_t$$

We shall describe adaptive algorithms for selecting suitable $\underline{c}_t$, d_t and θ_t for $t \in T_0$, and also a suitable $T \leq T_0$, based on a sequence of training samples $(\underline{x}(k), y(k))$, k=0,1,..., where $\underline{x}(k)$ is the input vector and y(k) is the desired signal at time instant k.

In the sequel, we shall derive our training algorithms under the assumption that $\underline{x}(k)$, y(k), k=0,1,..., are jointly stationary random sequences with $(\underline{x}(k), y(k))$ jointly distributed like $(\underline{x}, y)$. By appropriately selecting gain parameters, the tree-structured filter should be able to track sufficiently slowly varying nonstationary sequences.

When a training sample is presented to the tree-structured filter, it is used in two steps which we shall refer to as "tree growing" and "tree pruning". To start things off, the training sample is propagated down the tree. In the tree growing step the tap weight vectors, offsets and thresholds are updated at each node. In the tree pruning step, the selected pruned subtree and corresponding piecewise linear filter is updated.

More precisely, let $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$ be the selected values of $\underline{c}_t$, d_t, θ_t respectively, and let T(k) be the selected pruned subtree of T_0, based on the first k training samples $(\underline{x}(i), y(i))$, i=0, 1, ..., k-1. Now for $t \in T_0$ let

$$\hat{y}_t\,(k) = \underline{c}_t(k)'\underline{x}(k) + d_t(k)$$

The k+1-th training sample $(y(k), \underline{x}(k))$ is propagated down the tree T_0 from root node to terminal node according to the rule

$$\hat{y}_t(k) > \theta_t(k) \quad \text{go to right(t)}$$

$$\hat{y}_t(k) \leq \theta_t(k) \quad \text{go to left(t)}$$

As the training sample propagates down the tree, the values of $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$ are updated to $\underline{c}_t(k+1)$, $d_t(k+1)$, $\theta_t(k+1)$, respectively, during the tree growing step, and then the selected pruned subtree T(k) is updated to T(k+1) during the tree pruning step.

We shall specify optimization problems whose solutions are certain values of $\underline{c}_t$, d_t, and θ_t which shall be denoted as $\underline{c}_t^*$, d_t^* and θ_t^*, for $t \in T_o$. We shall also specify an optimization problem whose solution is a certain pruned subtree T^* of T_o. We then describe the tree growing and pruning algorithms which recursively generate estimates $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$ and $T(k)$ of $\underline{c}_t^*$, d_t^*, θ_t^* and T^* respectively.

We shall make use of the following notation. Let $\chi_t(k)$ and χ_t^* denote the (polygonal) domains corresponding to the decision rule (4.1) with $\underline{c}_t$, d_t, θ_t replaced by $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$ and $\underline{c}_t^*$, d_t^*, θ_t^*, respectively. Also, for $t \in T_o$ and $T \leq T_o$ let

$$\hat{y}_t(k) = \underline{c}_t(k)'\underline{x}(k) + d_t(k)$$

$$\hat{y}_T(k) = \hat{y}_t(k) \quad \text{if } t \in \tilde{T} \text{ and } \underline{x}(k) \in \chi_t(k)$$

$$\hat{y}_t^*(k) = \underline{c}_t^{*\prime}\underline{x}(k) + d_t^*,$$

$$\hat{y}_T^*(k) = \hat{y}_t^*(k) \quad \text{if } t \in \tilde{T} \text{ and } \underline{x}(k) \in \chi_t^*.$$

and

$$\hat{y}_t^* = \underline{c}_t^{*\prime}\underline{x} + d_t^*,$$

$$\hat{y}_T^* = \hat{y}_t^* \quad \text{if } t \in \tilde{T} \text{ and } \underline{x} \in \chi_t^*.$$

It will be convenient to denote $E\{\cdot \mid \underline{x} \in \chi_t^*\}$ by $E_t\{\cdot\}$, $P\{\cdot \mid \underline{x} \in \chi_t^*\}$ by $P_t\{\cdot\}$, and $Cov_t\{\cdot,\cdot\} = Cov\{\cdot,\cdot \mid \underline{x} \in \chi_t^*\}$, i.e., P_t, E_t and Cov_t denote the conditional probability, expectation and covariance given the input vector passes through node t, respectively.

We discuss tree growing and tree pruning in detail below. A summary of the overall algorithm is given in Figure 17.

```
Initialization :
        $p_t(0) = \frac{1}{2^{\mathrm{depth}(t)}}$,
        $\underline{c_t}(0) = \underline{0},\ d_t(0) = 0,\ \theta_t(0) = 0,$

Update :
    Let $(\underline{x}(k), y(k))$ be the k+1-th training sample.
    Let $\hat{y}_t(k) = \underline{c_t}(k)'\underline{x}(k) + d_t(k)$
    Propagate the training sample from root node to terminal node according to
        $\hat{y}_t(k) > \theta_t(k)$   go to right(t)
        $\hat{y}_t(k) \le \theta_t(k)$   go to left(t)
    If the training sample passes through node t
        $\underline{c_t}(k+1) = \underline{c_t}(k) + \mu(y(k) - \hat{y}_t(k))\underline{x}(k)$
        $d_t(k+1) = d_t(k) + \mu(y(k) - \hat{y}_t(k))$
        $\theta_t(k+1) = \theta_t(k) + \mu(y(k) - \theta_t(k))$
        $p_t(k+1) = p_t(k) + \mu(1 - p_t(k))$
        $\varepsilon_t(k+1) = \varepsilon_t(k) + \mu((y(k) - \hat{y}_t(k))^2 - \varepsilon_t(k))$
    If the training sample does not pass through node t
        $\underline{c_t}(k+1) = \underline{c_t}(k)$
        $d_t(k+1) = d_t(k)$
        $\theta_t(k+1) = \theta_t(k)$
        $p_t(k+1) = p_t(k) - \mu p_t(k)$
        $\varepsilon_t(k+1) = \varepsilon_t(k)$
    Generate T(k+1) using the pruning algorithm.
```

Figure 17. Summary of Training Algorithm.

4.1 Tree Growing

We discuss the weight and offset adaptation, and the threshold adaptation separately.

4.1.1 Weight and Offset Adaptation

Here we (inductively) define the optimal tap weight vectors $\underline{c_t}^*$ and offsets d_t^* for $t \in T_o$, and specify algorithms for estimating them. Suppose that $\underline{c_s}^*$, d_s^*, θ_s^* are defined for all ancestors s of a node $t \in T_o$ (and hence χ_t^* is defined). We choose $\underline{c_t}^*$ and d_t^* to minimize the mean square error (MSE) between the true conditional filter output $\hat{y}_t$ and the desired signal y, given that the channel output vector $\underline{x}$ passes through the node t, i.e., we choose $\underline{c_t}^*$ and d_t^* to minimize

$$\varepsilon_t = E_t\{ (\hat{y}_t - y)^2 \}$$

with respect to $\underline{c_t}$ and d_t.

We sequentially estimate the values $\underline{c_t}^*$ and d_t^* based on the training sequence using an approximate stochastic gradient algorithm :

$$\underline{c_t}(k+1) = \underline{c_t}(k) + \mu\,(y(k) - \hat{y}_t\,(k))\,\underline{x}(k)\,I(\underline{x}(k) \in \chi_t(k)), \tag{4.2}$$

$$d_t(k+1) = d_t(k) + \mu\,(y(k) - \hat{y}_t\,(k))\,I(\underline{x}(k) \in \chi_t(k)), \tag{4.3}$$

where the gain parameter μ is a small positive number and $\underline{c_t}(0) = 0$, $d_t(0) = 0$. Note that these are indeed approximate stochastic gradient algorithms for minimizing ε_t over $\underline{c_t}$ and d_t : the instantaneous gradient of ε_t with respect to $\underline{c_t}$ (for example) evaluated at $\underline{x} = \underline{x}(k)$, $y = y(k)$ and $\hat{y}_t = \hat{y}_t(k)$ is $-(2/p_t^*)\,(y(k) - \hat{y}_t(k))\,\underline{x}(k)\, I(\underline{x}(k) \in \chi_t^*)$. However for large k, $\chi_t(k)$ should presumably be close to χ_t^*. These approximations allow us to update the node parameters for all nodes in the tree with each training sample, i.e., it is not necessary to wait until the node parameters of all the ancestors of a given node have converged in order to update the given node parameters. Finally, note that in the fully adaptive mode, y(k) is replaced by $\tilde{y}(k) = \mathrm{dec}(\hat{y}_{T(k)}(k))$ in (4.2) and (4.3).

4.1.2 Threshold Adaptation

Here we (inductively) define the optimal thresholds θ_t^* for $t \in T_o$, and specify algorithms for estimating them. Suppose that $\underline{c}_s^*$, d_s^*, θ_s^* are defined for all ancestors s of a node $t \in T_o$ (and hence χ_t^* is defined) and also $\underline{c}_t^*$ and d_t^* are defined (and hence $\hat{y}_t^*$ is defined). Let $\ell(t) = \mathrm{left}(t)$ and $r(t) = \mathrm{right}(t)$. We proceed by showing that a partition $\chi^o_{\ell(t)}$ and $\chi^o_{r(t)}$ of χ_t^* is optimal in a certain sense, and then showing that $\chi^*_{\ell(t)}$ and $\chi^*_{r(t)}$ approximate $\chi^o_{\ell(t)}$ and $\chi^o_{r(t)}$, respectively, in a certain sense for suitable choice of θ_t^*.

Suppose we require that the probability that the input vector $\underline{x}$ passes through node $\ell(t)$ given it passes through node t is 1/2. This keeps the tree balanced. Suppose we also require that the filter outputs $\hat{y}_{\ell(t)}$ and $\hat{y}_{r(t)}$ be easy to fit to the data over their respective domains $\chi_{\ell(t)}$ and $\chi_{r(t)}$ in a suitable sense. A reasonable way to implement this is by choosing $\chi_{\ell(t)}$ and $\chi_{r(t)}$ to minimize the mean conditional variance of the desired signal y given only the knowledge of whether the channel output vector $\underline{x}$ passes through node $\ell(t)$ or r(t), given that it passes through node t. Let $\mathcal{G}_t$ be the σ-field generated by the events $\{\underline{x} \in \chi_{\ell(t)}\}$ and $\{\underline{x} \in \chi_{r(t)}\}$. To condition on $\mathscr{G}_t$ is to know whether $\underline{x}$ passes through $\ell(t)$ or r(t). Formally, for each node $t \in T_o - \tilde{T}_o$ we want to minimize

$$E_t\{\mathrm{Var}\{y \mid \mathscr{G}_t\}\} \tag{4.4}$$

over partition $\chi_{\ell(t)}$ and $\chi_{r(t)}$ of χ_t^* subject to the constraint that

$$P_t\{\underline{x} \in \chi_{\ell(t)}\} = P_t\{\underline{x} \in \chi_{r(t)}\} = \frac{1}{2}. \tag{4.5}$$

In the Appendix (Section 4.4) it is shown that the partition $\chi^o_{\ell(t)} = \{\underline{\xi} \in \chi_t^* : E\{y \mid \underline{x} = \underline{\xi}\} - E_t\{y\} \le 0\}$, $\chi^o_{r(t)} = \{\underline{\xi} \in \chi_t^* : E\{y \mid \underline{x} = \underline{\xi}\} - E_t\{y\} > 0\}$ in fact minimizes (4.4), provided it satisfies (4.5). Note that $E\{y \mid \underline{x}\}$ is a conditionally unbiased estimate of $E_t\{y\}$, i.e., $E_t\{E\{y \mid \underline{x}\}\} = E_t\{y\}$. Hence it is reasonable to believe that $\chi^o_{\ell(t)}$ and $\chi^o_{r(t)}$ will approximately satisfy (4.5). Now we know that $\hat{y}_t^*(\underline{x})$ is the optimal affine estimator of y based on $\underline{x}$ in the MSE sense and $E\{y \mid \underline{x}\}$ is the optimal nonlinear estimator of y based on $\underline{x}$ in the MSE sense. Put another way, $\hat{y}_t^*(\underline{x})$ is a wide sense version of $E\{y \mid \underline{x}\}$ (see [52]). Hence we can interpret $\chi^*_{\ell(t)}$ and $\chi^*_{r(t)}$ as wide sense versions of $\chi^o_{\ell(t)}$ and $\chi^o_{r(t)}$, respectively, if we take $\theta_t^* = E_t\{y\}$.

The conclusion of the above discussion is that it is reasonable to take $\theta_t^* = E_t\{y\}$. Now θ_t^* will (trivially) minimize the cost function

$$\alpha_t = E_t\{(\theta_t - y)^2\}$$

over θ_t. We sequentially estimate the value of θ_t^* based on the training sequence using an approximate stochastic gradient algorithm :

$$\theta_t(k+1) = \theta_t(k) + \mu\,(y(k) - \theta_t(k))\, I(\underline{x}(k) \in \chi_t(k)) \tag{4.6}$$

where the gain parameter μ is a small positive number and $\theta_t(0) = 0$. Note that this is indeed an approximate stochastic gradient algorithm for minimizing α_t over θ_t : the instantaneous gradient of α_t with respect to θ_t evaluated at $\underline{x} = \underline{x}(k)$, $y = y(k)$ and $\theta_t = \theta_t(k)$ is $-(2/p_t^*)\,(y(k) - \theta_t(k))\, I(\underline{x}(k) \in \chi_t^*)$. However for large k, $\chi_t(k)$ should presumably be close to χ_t^*. As with the tap weight and offset adaptation these approximations allows us to update the node parameters for all nodes in the tree with each training sample, i.e., it is not necessary to wait until the node parameters of all the ancestors of a given node have converged in order to update the given node parameters. The validity of these assumptions are verified by analysis (Chapter 5) and numerical experiments (Chapter 6). Also note that the offset adaptation (4.3) is different from threshold adaptation (4.6). This is consistent with the need for both of the parameters d_t and θ_t corresponding to the uncertainty whether node t will eventually be selected as a terminal node, as discussed in Chapter 3. Finally, note that in the fully adaptive mode, y(k) is replaced by $\tilde{y}(k) = \mathrm{dec}(\hat{y}_{T(k)}(k))$ in (4.6).

4.2 Tree Pruning

Here we define an optimally pruned subtree T^* of T_o and specify an algorithm for estimating it. We choose T^* to minimize the mean square error (MSE) between the optimal piecewise linear filter output $\hat{y}_T^*$ and the desired signal y, i.e., we choose T^* to minimize

$$\varepsilon_T^* = E\{(\hat{y}_T^* - y)^2\}$$

over all pruned subtrees T of T_o. Note that we can decompose ε_T^* as

$$\varepsilon_T^* = \sum_{t \in \tilde{T}} \varepsilon_t^* \, p_t^*$$

where

$$\varepsilon_t^* = E_t\{(\hat{y}_t^* - y)^2\}$$

and

$$p_t^* = P\{\underline{x} \in \chi_t^*\}.$$

as before. Let $\varepsilon_t(k)$, $p_t(k)$ denote estimates of ε_t^* and p_t^*, respectively, based on the first k training samples. We estimate T^* by the pruned subtree T(k) which minimizes the "estimated" MSE

$$\varepsilon_T(k) = \sum_{t \in \tilde{T}} \varepsilon_t(k)\, p_t(k)$$

over all pruned subtrees T of T_o. This approach avoids overfitting the finite amount of data used for training up to time k. For small k, T(k) will typically be the root node of T_o, corresponding to a linear filter being optimal for small data sets; for large k, T(k) will usually be a nontrivial pruned subtree of T_o, corresponding to a piecewise linear filter being optimal for larger data sets. We next derive the estimates of $p_t(k)$ and $\varepsilon_t(k)$.

First observe that p_t^* will (trivially) minimize the cost function

$$\beta_t = E\{(p_t - I(\underline{x} \in \chi_t^*))^2\}$$

over p_t. We sequentially estimate the value of p_t^* based on the training sequence using an approximate stochastic gradient algorithm

$$p_t(k+1) = p_t(k) + \mu(I(\underline{x}(k) \in \chi_t(k)) - p_t(k)) \qquad (4.7)$$

where the gain parameter μ is a small positive number and the $p_t(k)$'s are initialized to the probabilities of a balanced tree,

$$p_t(0) = \frac{1}{2^{\mathrm{depth}(t)}}.$$

Note that if $\mu \le 1$ then $p_t(k)$'s are valid probabilities, i.e., $0 \le p_t(k) \le 1$ for all $t \in T_o$ and $\sum_{t \in \tilde{T}} p_t(k) = 1$ for all $T \le T_o$. Also note that this is indeed an approximate stochastic gradient algorithm for minimizing β_t with respect to p_t: the instantaneous gradient of β_t with respect to p_t evaluated at $\underline{x} = \underline{x}(k)$ and $p_t = p_t(k)$ is $-2(I(\underline{x}(k) \in \chi_t^*) - p_t(k))$. However for large k, $\chi_t(k)$ should presumably be close to χ_t^*.

Now ε_t^* will (trivially) minimize the cost function

$$\gamma_t = E_t\{(\varepsilon_t - (\hat{y}_t^* - y)^2)^2\}$$

over ε_t. We sequentially estimate the value of ε_t^* based on the training sequence using an approximate stochastic gradient algorithm:

$$\varepsilon_t(k+1) = \varepsilon_t(k) + \mu\,((\hat{y}_t(k) - y(k))^2 - \varepsilon_t(k))\, I(\underline{x}(k) \in \chi_t(k)) \qquad (4.8)$$

where the gain parameter μ is a small positive number. Note that this is indeed an approximate stochastic gradient algorithm for minimizing γ_t over ε_t : the instantaneous gradient of γ_t with respect to ε_t evaluated at $\underline{x} = \underline{x}(k)$, $y = y(k)$, $\hat{y}_t^* = \hat{y}_t^*(k)$ and $\varepsilon_t = \varepsilon_t(k)$ is $-(2/p_t^*)\,((\hat{y}_t^*(k) - y(k))^2 - \varepsilon_t(k))\, I(\underline{x}(k) \in \chi_t^*)$. However for large k, $\hat{y}_t(k)$ and $\chi_t(k)$ should presumably be close to $\hat{y}_t^*(k)$ and χ_t^*, respectively. In the fully adaptive mode y(k) is replaced by $\tilde{y}(k) = \mathrm{dec}(\hat{y}_{T(k)}(k))$ in (4.8).

Given $p_t(k)$, $\varepsilon_t(k)$ for $t \in T_o$, the following simple and efficient algorithm generates T(k) from T_o (see Chapter 3 for tree notation).

4.2.1 Pruning Algorithm

Assume T_o has M nodes $t_1 < t_2 < ... < t_M$. Let $\ell(t) = \text{left}(t)$ and $r(t) = \text{right}(t)$ for $t \in T_o$.

```
$T = T_o$
for $t = t_M, ..., t_1$
    { if $t \in \tilde{T}$ then
        $\delta_t = \varepsilon_t(k)p_t(k)$
      if $t \in T - \tilde{T}$ then
        { $\delta_t = \delta_{\ell(t)} + \delta_{r(t)}$
          if $\varepsilon_t(k)p_t(k) \le \delta_t$ then
            { $T \leftarrow T - (T_{\ell(t)} \cup T_{r(t)})$
              $\ell(t) = 0$, $r(t) = 0$
              $\delta_t = \varepsilon_t(k)p_t(k)$ } } }
$T(k) = T$                                                    □
```

The pruning algorithm described above is essentially a bottom-up algorithm that starts from the terminal nodes and proceeds up the tree, pruning away branches. The pruning algorithm actually generates the smallest optimally pruned subtree, i.e., if T' also minimizes $\varepsilon_T(k)$ over $T \le T_o$ then $T(k) \le T'$. The existence and the properties of smallest optimally pruned subtrees are discussed in [51]. Various pruning algorithms (including the one mentioned above) have been used to design tree-structured classifiers[51, 53] and vector quantizers[54, 55].

4.3 Implementation Issues and Remarks

(1) Observe that $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$, $p_t(k)$ and $\varepsilon_t(k)$ are computed recursively and must be updated with each training sample. However, $T(k)$ is not computed recursively and in fact need not be computed in the training mode. In the fully adaptive mode $T(k)$ need only be computed when it is reasonably thought to have changed significantly.

(2) It makes some heuristic sense to scale the gain parameters μ in the update equations for $\underline{c}_t(k)$, $d_t(k)$, $\theta_t(k)$ and $\varepsilon_t(k)$ so as to compensate for the unequal probabilities that a training sample passes through a particular node. Note the factor $1/p_t^*$ multiplying the instantaneous gradient of the cost functions leading to these update equations in this regard. One choice of nonadaptive scaling would be to replace μ by $\mu/2^{-\text{depth}(t)}$; a choice of adaptive scaling would be to replace μ by $\mu/p_t(k+1)$ (we scale by $1/p_t(k+1)$ instead of $1/p_t(k)$ because if $\underline{x}(k) \in$ cstk then $p_t(k+1) > \mu$ and the gain $\mu/p_t(k+1)$ is bounded).

(3) We have assumed above that $x(k)$ and $y(k)$ are real valued. Suppose now that $x(k)$ and $y(k)$ are complex-valued. In this case it can be argued that the binary tree should be replaced by a quaternary tree, i.e., a tree in which each node has four offspring. Let left1(t), left2(t), right1(t) and right2(t) denote the four offspring of node t on such a tree. The training sample $(\underline{x}(k), y(k))$ is now propagated from root node to terminal node according to

$$\mathrm{Re}[\hat{y}_t(k)] \le \mathrm{Re}[\theta_t(k)],\ \mathrm{Im}[\hat{y}_t(k)] \le \mathrm{Im}[\theta_t(k)] \quad \text{go to left1(t)}$$

$$\mathrm{Re}[\hat{y}_t(k)] \le \mathrm{Re}[\theta_t(k)],\ \mathrm{Im}[\hat{y}_t(k)] > \mathrm{Im}[\theta_t(k)] \quad \text{go to left2(t)}$$

$$\mathrm{Re}[\hat{y}_t(k)] > \mathrm{Re}[\theta_t(k)],\ \mathrm{Im}[\hat{y}_t(k)] \le \mathrm{Im}[\theta_t(k)] \quad \text{go to right1(t)}$$

$$\mathrm{Re}[\hat{y}_t(k)] > \mathrm{Re}[\theta_t(k)],\ \mathrm{Im}[\hat{y}_t(k)] > \mathrm{Im}[\theta_t(k)] \quad \text{go to right2(t)}$$

where Re[·] and Im[·] denotes the real and imaginary parts of the argument, respectively. The remaining changes to the training algorithm are obvious and we omit the details.

(4) The tree-structured piecewise linear filter described above may be modified into a tree-structured piecewise nonlinear filter by replacing the linear filter at each node by a nonlinear filter. For example in the channel equalization application, a tree-structured piecewise linear equalizer may be easily modified into a tree-structured piecewise decision feedback equalizer by replacing the linear equalizer at each node by a decision feedback equalizer. Thus the input vector

$$\underline{x}(k) = [x(k),\ldots,x(k-L+1)]'$$

is replaced by

$$\underline{z}(k) = [x(k+L),\ldots,x(k),y(k-1),\ldots,y(k-L)]'$$

in the training mode, and by

$$\tilde{\underline{z}}(k) = [y(k+L),\ldots,y(k),\tilde{y}(k-1),\ldots,\tilde{y}(k-L)]'$$

in the fully adaptive mode where $\tilde{y}(k) = \mathrm{dec}(\hat{y}_{T(k)}(k))$.

(5) We point out that the basic tree T_o should be chosen as large as possible subject to hardware constraints. The pruning process will always select the right-sized subtree and corresponding piecewise linear filter. A dedicated hardware can be configured to implement the tree-structured filter. The choice of a large basic tree does not pose memory problems since the record associated with each node occupies only a few bytes. With additional memory and a separate processing unit allocated to each level of the tree it is possible to configure a parallel architecture which allows samples at different levels of the tree to be processed simultaneously. Due to the sequential manner in which samples traverse the

tree, only one of the nodes at a particular level of the tree is processing an input sample at any point in time. Hence, all that is required by this parallel configuration is to be able to process the input sample at just one node within a symbol interval. Thus the proposed tree-structured filter can potentially be used in high data rate applications.

(6) The tree-structured piecewise linear adaptive filter was motivated by the successful application of classification and regression trees to difficult nonlinear and nonparametric problems[51]. However, although our tree-structured filter bears some similarity to classification and regression trees, it is actually quite different from a classification and regression tree. Here the terminal nodes are not just assigned a class label or a regression value, but rather represent a linear filter with restricted domain. It is also different in that classification and regression trees are determined in a batch mode offline, whereas the tree structured filter is determined recursively in real-time. On the other hand, like classification and regression trees, our procedure does perform a sequential, hierarchical partitioning of the input space. Also a pruning algorithm is used to obtain a right-sized tree, i.e., a tree which neither underfits nor overfits the data.

4.4 Appendix

Let y and $\underline{x}$ be scalar and vector ($\mathbb{R}^L$) valued random variables, respectively. Let χ, χ_1 and χ_2 be Borel subsets of $\mathbb{R}^L$ with $\chi_1 \cup \chi_2 = \chi$ and $\chi_1 \cap \chi_2 = \phi$, and let $\mathcal{Y}$ be the σ-field generated by the events $\{\underline{x} \in \chi_1\}$, $\{\underline{x} \in \chi_2\}$. In the sequel, it will be assumed that all quantities are conditioned with respect to $\underline{x} \in \chi$. It is required to minimize $E\{\mathrm{Var}\{y|\mathcal{Y}\}\}$ over χ_1 and χ_2 subject to the constraint

$P\{\underline{x} \in \chi_1\} = P\{\underline{x} \in \chi_2\} = 0.5$. We shall show that $\chi_1{}^{\circ} = \{\underline{\xi} : E\{y \mid \underline{x} = \underline{\xi}\} > E\{y\}\}$, $\chi_2{}^{\circ} = \{\underline{\xi} : E\{y \mid \underline{x} = \underline{\xi}\} \leq E\{y\}\}$, is a solution (assuming that $P\{\underline{x} \in \chi_1{}^{\circ}\} = P\{\underline{x} \in \chi_2{}^{\circ}\} = 0.5$). Let $\mathcal{Y}^{\circ}$ be the σ-field generated by the events $\{\underline{x} \in \chi_1{}^{\circ}\}$, $\{\underline{x} \in \chi_2{}^{\circ}\}$.

We want to show $E\{Var\{y \mid \mathcal{Y}\}\} \geq E\{Var\{y \mid \mathcal{Y}^{\circ}\}\}$. From the identity

$$E\{Var\{y \mid \mathcal{Y}\}\} = Var\{y\} - Var\{E\{y \mid \mathcal{Y}\}\}$$

it is enough to show that $Var\{E\{y \mid \mathcal{Y}\}\} \leq Var\{E\{y \mid \mathcal{Y}^{\circ}\}\}$. Let $g(\underline{\xi}) = E\{y \mid \underline{x} = \underline{\xi}\} - E\{y\}$. Note that $\chi_1{}^{\circ} = \{\underline{\xi} : g(\underline{\xi}) > 0\}$, $\chi_2{}^{\circ} = \{\underline{\xi} : g(\underline{\xi}) \leq 0\}$. Hence

$$\begin{aligned}
Var\{E\{y \mid \mathcal{Y}\}\} &= E\{(E\{y \mid \mathcal{Y}\} - E\{y\})^2\} \\
&= E\{(E\{E\{y \mid \underline{x}\} - E\{y\} \mid \mathcal{Y}\})^2\} \\
&= E\{(E\{g(\underline{x}) \mid \mathcal{Y}\})^2\} \\
&= \sum_{i=1}^{2} (E\{g(\underline{x}) \mid \underline{x} \in \chi_i\})^2 P\{\underline{x} \in \chi_i\} \\
&= \sum_{i=1}^{2} (E\{g(\underline{x}) I(\underline{x} \in \chi_i)\})^2 P\{\underline{x} \in \chi_i\}^{-1} \\
&= 2 \sum_{i=1}^{2} (E\{g(\underline{x}) I(\underline{x} \in \chi_i)\})^2
\end{aligned}$$

since $P\{\underline{x} \in \chi_i\} = 0.5 \quad i = 1, 2$.

It is enough to consider the following two cases (the other two cases are similar)

(i) $E\{g(\underline{x}) I(\underline{x} \in \chi_1)\} > 0$, $E\{g(\underline{x}) I(\underline{x} \in \chi_2)\} > 0$

(ii) $E\{g(\underline{x}) I(\underline{x} \in \chi_1)\} > 0$, $E\{g(\underline{x}) I(\underline{x} \in \chi_2)\} \leq 0$

In the first case we have

$$\begin{aligned}
\mathrm{Var}\{E\{y|\mathcal{Y}\}\} &= 2\sum_{i=1}^{2} (E\{g(\underline{x})\, I(\underline{x}\in\chi_i)\})^2\\
&\le 2\sum_{i=1}^{2} (E\{g(\underline{x})\, I(\underline{x}\in\chi_i,\ g(\underline{x})>0)\})^2\\
&\le 2\,(\sum_{i=1}^{2} E\{g(\underline{x})\, I(\underline{x}\in\chi_i,\ g(\underline{x})>0)\})^2\\
&= 2\,(E\{g(\underline{x})\, I(g(\underline{x})>0)\})^2\\
&= 2\,(E\{g(\underline{x})\, I(\underline{x}\in\chi_1^{\circ})\})^2\\
&\le \mathrm{Var}\,\{E\{y|\mathcal{Y}^{\circ}\}\}
\end{aligned}$$

In the second case we have

$$\begin{aligned}
\mathrm{Var}\{E\{y|\mathcal{Y}\}\} &= 2\sum_{i=1}^{2} (E\{g(\underline{x})\, I(\underline{x}\in\chi_i)\})^2\\
&\le 2\,[(E\{g(\underline{x})\, I(\underline{x}\in\chi_1,\ g(\underline{x})>0)\})^2\\
&\quad + (E\{g(\underline{x})\, I(\underline{x}\in\chi_2,\ g(\underline{x})\le 0)\})^2]\\
&\le 2\,[(E\{g(\underline{x})\, I(g(\underline{x})>0)\})^2 + (E\{g(\underline{x})\, I(g(\underline{x})\le 0)\})^2]\\
&= 2\sum_{i=1}^{2} (E\{g(\underline{x})\, I(\underline{x}\in\chi_i^{\circ})\})^2\\
&= \mathrm{Var}\,\{E\{y|\mathcal{Y}^{\circ}\}\}
\end{aligned}$$

(in the above cases we have used the fact that if $E\{Z\}\ge 0$ then $(E\{Z\})^2 \le (E\{Z\, I(Z\ge 0)\})^2$, and if $E\{Z\}\le 0$, then $(E\{Z\})^2 \le (E\{Z\, I(Z\le 0)\})^2$). Hence χ_1°, χ_2° do in fact have the asserted optimality property.

CHAPTER 5
ANALYSIS OF THE TREE-STRUCTURED PIECEWISE LINEAR ADAPTIVE FILTER

In this chapter we analyze the asymptotic (large time) behavior of the training algorithm for the tree-structured piecewise linear filter described in Chapter 4. A fundamental difficulty arises because the training data at a non-root node depends on the unconverged parameters at its ancestor nodes. Hence the training data at a non-root node filter has a complex nonstationary and dependent character, even with the assumption of i.i.d training data to the overall tree filter. A further difficulty in analyzing the tree filter is the interaction between the tree growing and tree pruning steps.

To see more precisely the difficulty in analyzing the training algorithm for the tree-structured filter, consider the update equation for $\underline{c}_t(k)$ (see (4.2)). This is a standard LMS algorithm with training data $\{\underline{x}(k)I(\underline{x}(k) \in \chi_t(k)), y(k)\}$. Note that the domain $\chi_t(k)$ depends on the parameters $\underline{c}_s(k)$, $d_s(k)$ and $\theta_s(k)$ at the ancestor nodes s of node t, and hence on $\{\underline{x}(i), y(i) : i < k\}$. Thus the training data for a non-root node filter $\{\underline{x}(k)I(\underline{x}(k) \in \chi_t(k)), y(k)\}$ is indeed nonstationary and dependent even under the assumption that the training data for the overall tree filter $\{\underline{x}(k), y(k)\}$ is i.i.d.

There is a large body of literature on the convergence analysis of LMS and LMS-type algorithms with fixed gains and dependent data [56-60]. Consider the standard LMS algorithm

$$\underline{c}(k+1) = \underline{c}(k) + \mu(y(k) - \hat{y}(k))\underline{x}(k)$$

For data with finite memory (i.e. M-dependent data) quasiconvergence of the weights and outputs to the optimal Wiener solution in quadratic mean has been shown assuming only the existence of certain moments of the inputs and outputs[59]. For data with infinite memory (e.g., general strong mixing or asymptotically uncorrelated data) similar convergence results have been obtained but require bounded weights[60]. The actual algorithm that is analyzed in the latter case is a projected version of the LMS algorithm which takes the form

$$\underline{c}(k+1) = P(\underline{c}(k) + \mu(y(k) - \hat{y}(k))\underline{x}(k))$$

Here the operator $P : \mathbb{R}^L \rightarrow S$ projects the weight vector into a bounded region S whenever it attempts to leave S.

Now consider the tree-structured filter with i.i.d training data. As mentioned above, even with this assumption the training data at a non-root node filter is both nonstationary and dependent. Furthermore, it has infinite memory and does not obviously satisfy any of the standard dependence models (like strong mixing). The actual algorithm that shall be analyzed here is derived from a penalized version of the LMS algorithm which takes the form

$$\underline{c}(k+1) = (1-\mu^{v})\underline{c}(k) + \mu(y(k) - \hat{y}(k))\underline{x}(k)$$

The parameter μ^{v} can be viewed as arising from the penalty cost function $E\{(y-\hat{y})^2\} + \mu^{v-1}\ |\underline{c}|^2$. The advantage of the penalty versus the projection approach for our problem is that it is much easier to implement and analyze, and does not require prior knowledge of what region the optimal solution lies in. We are not aware of any literature on penalized LMS algorithms of this type.

Hence the actual training algorithm for the tree-structured piecewise linear filter that we shall analyze takes the form (compare with (4.2), (4.3), (4.6), (4.7), (4.8))

$$\underline{c}_t(k+1) = (1-\mu^{v})\underline{c}_t(k) + \mu(y(k) - \hat{y}_t(k))\underline{x}(k)I(\underline{x}(k) \in \chi_t(k))$$

$$d_t(k+1) = (1-\mu^{v})d_t(k) + \mu(y(k) - \hat{y}_t(k))I(\underline{x}(k) \in \chi_t(k))$$

$$\theta_t(k+1) = (1-\mu^{\nu})\theta_t(k) + \mu(y(k) - \theta_t(k))I(\underline{x}(k) \in \chi_t(k))$$

$$\varepsilon_t(k+1) = (1-\mu^{\nu})\varepsilon_t(k) + \mu((y(k) - \hat{y}_t(k))^2 - \varepsilon_t(k))I(\underline{x}(k) \in \chi_t(k))$$

$$p_t(k+1) = p_t(k) + \mu(I(\underline{x}(k) \in \chi_t(k)) - p_t(k))$$

Otherwise tree growing and pruning is exactly as described in Chapter 4. Note that the update equation for $p_t(k)$ is unchanged here since $p_t(k)$ is bounded a.s.

We shall make the following assumptions :

(A1) $\{\underline{x}(k), y(k)\}$ is an i.i.d random process with $(\underline{x}(k), y(k))$ jointly distributed like $(\underline{x}, y)$

(A2) $\underline{a}'\underline{x}$ has a bounded density function for all $\underline{a} \neq \underline{0}$

(A3) $\underline{x}$ and y have finite eighth order moments

Remark : To avoid trivialities we shall also assume $p_t^* > 0$, $\mathrm{Cov}_t\{\underline{x}\} > \underline{0}$, $\mathrm{Cov}_t\{\underline{x},y\} \neq \underline{0}$ for all $t \in T_o$, so that all of the tree parameters are well defined. In view of (A2) this is equivalent to just assuming

$$\mathrm{Cov}_t\{\underline{x},y\} \neq \underline{0} \text{ for all } t \in T_o.$$

This can be proved by induction on the levels of the tree. Indeed, suppose for node t that $p_t^* > 0$ and $\mathrm{Cov}_t\{\underline{x}\} > \underline{0}$. Then $\underline{c}_t^* = \mathrm{Cov}_t\{\underline{x}\}^{-1}\,\mathrm{Cov}_t\{\underline{x},y\} \neq \underline{0}$ and so $\hat{y}_t^* = \underline{c}_t^{*\prime}\underline{x} + d_t^*$ is a continuous random variable. Also $E_t\{\hat{y}_t^*\} = E_t\{y\} = \theta_t^*$. It follows that $P_t\{\hat{y}_t^* < \theta_t^*\} \in (0,1)$ and so $p_s^* > 0$ for $s = \ell(t), r(t)$. Since $\underline{a}'\underline{x}$ is continuous for all $\underline{a} \neq \underline{0}$, $\mathrm{Cov}_s\{\underline{x}\} > \underline{0}$ for $s = \ell(t), r(t)$.

We shall prove the following theorem.

Theorem 1 : Assume (A1) – (A3) and $1 < \nu < 1 + (1/5)\,4^{-\mathrm{depth}(T_o)}$. Then

$$\limsup_{k\to\infty} E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\} = O(\mu^{\alpha}) \text{ as } \mu \to 0,$$

where

$$\alpha = \frac{5}{8}\,(1 - \nu + \frac{1}{5}\,4^{-\mathrm{depth}(T_o)}).$$

The theorem asserts that there is quasiconvergence of the output of the selected tree filter $\hat{y}_{T(k)}$ to the optimal tree filter $\hat{y}^*_{T^*}$ in quadratic mean. Note that the theorem specifies that the output MSE is asymptotically upper bounded by a const.μ^{α} where $\alpha < 1$ and depends on the depth of the basic tree T_o; this is in contrast to linear filtering which yields an upper bound const.μ[2]. Also note that the theorem does not specify a stability bound on μ; this is again in contrast to linear filtering where stability bounds on μ (involving the eigenvalues of the input autocorrelation matrix) can be readily determined[2].

The training algorithm for the tree-structured piecewise linear filter consists of a collection of coupled difference equations at the nodes of a binary tree. Our approach to proving Theorem 1 is to perform an order analysis of these difference equations at the successive levels of the tree. As will be seen in the sequel, there is an interesting interplay in the analysis between boundedness and convergence.

The proof of Theorem 1 is based on the following two theorems.

Theorem 2 : Assume conditions of Theorem 1. Then for $T \leq T_o$

$$\limsup_{k \to \infty} E\{(\hat{y}_T(k))^4\} = O(\mu^{1-\nu}) \text{ as } \mu \to 0 \tag{5.1}$$

$$\limsup_{k \to \infty} E\{(\hat{y}_T(k) - \hat{y}^*_T(k))^2\} = O(\mu^{\beta}) \text{ as } \mu \to 0 \tag{5.2}$$

where

$$\beta = \frac{1}{2}\,(1 - \nu + 4^{-\operatorname{depth}(T_o)})$$

Theorem 3 : Assume conditions of Theorem 1. Then for $T \leq T_o$

$$\limsup_{k \to \infty} P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\} = O(\mu^{\beta/2}) \text{ as } \mu \to 0$$

Theorem 2 asserts bounds and convergence for the tree-structured filter $\hat{y}_T$ for a fixed pruned subtree T of T_o. This theorem is concerned with tree growing and is proved in Section 5.1. Theorem 3 asserts the convergence of the MSE $\varepsilon^*_{T(k)}$ of the selected tree filter $\hat{y}^*_{T(k)}$ (with T(k) considered as fixed†) to the MSE $\varepsilon^*_{T^*}$ of the optimal tree filter $\hat{y}^*_{T^*}$. This theorem is concerned with tree pruning and is proved in Section 5.2.

Assuming Theorems 2 and 3 are true, we prove Theorem 1 as follows. Here and in the sequel $K_1, K_2, \dots$ denote positive constants whose values can change from one proof to another.

Proof of Theorem 1 : We have

$$E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\} = E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\, I(\varepsilon^*_{T(k)} = \varepsilon^*_{T^*})\} \tag{5.3}$$

$$+ E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\, I(\varepsilon^*_{T(k)} > \varepsilon^*_{T^*})\}$$

Now if $\varepsilon^*_{T(k)} = \varepsilon^*_{T^*}$ then it is not hard to see that $\hat{y}^*_{T(k)}(k) = \hat{y}^*_{T^*}(k)$. Hence using Theorem 2

$$E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\, I(\varepsilon^*_{T(k)} = \varepsilon^*_{T^*})\} \le K_1 \max_{T \le T_o} E\{(\hat{y}_T(k) - \hat{y}^*_T(k))^2\}$$

$$= O(\mu^\beta) \tag{5.4}$$

as $k \to \infty$ and $\mu \to 0$. Also using Holder's inequality and Theorems 2 and 3

$$E\{(\hat{y}_{T(k)}(k) - \hat{y}^*_{T^*}(k))^2\, I(\varepsilon^*_{T(k)} > \varepsilon^*_{T^*})\}$$

$$\le K_2\, (E\{(\hat{y}_{T(k)}(k))^4\}^{1/2} + E\{(\hat{y}^*_{T^*}(k))^4\}^{1/2})\, P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\}^{1/2}$$

$$\le K_3\, (\max_{T \le T_o} E\{(\hat{y}_T(k))^4\}^{1/2} + 1)\, P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\}^{1/2}$$

$$= O(\mu^{(1-\nu)/2})\, O(\mu^{\beta/4})$$

† i.e. $\varepsilon^*_{T(k)} = E\{(\hat{y}^*_{T(k)}(k) - y(k))^2 \mid \mathfrak{F}_k\}$ where $\mathfrak{F}_k$ is the σ-field generated by $\{\underline{x}(i), y(i) : i < k\}$

$$= O(\mu^{\alpha}) \tag{5.5}$$

as $k \to \infty$ and $\mu \to 0$. Combining (5.3)-(5.5) completes the proof. $\square$

The proofs of Theorem 2 and 3 in the sequel will be simplified by assuming $d_t(k) = d_t^* = 0$. We can do this without loss of generality by considering the augmented vectors $\tilde{\underline{c}}_t(k) = (\underline{c}_t(k)', d_t(k))'$, $\tilde{\underline{c}}_t^* = (\underline{c}_t^{*\prime}, d_t^*)'$. There are a few details to be checked here but we omit further discussion for brevity.

Also we shall frequently refer to the following fact about linear difference inequalities which we state here for convenience and without proof.

Fact : Let $\{\xi_k\}$ be a sequence of non-negative numbers which satisfies

$$\xi_{k+1} \le (1 - \alpha(\mu) + \beta_k(\mu))\xi_k + \gamma_k(\mu), \;\; k = 0, 1, \ldots$$

for some functions $\alpha(\mu)$, $\beta_k(\mu)$, $\gamma_k(\mu)$, $\mu > 0$. Let

$$\beta(\mu) = \limsup_{k \to \infty} \beta_k(\mu), \tag{5.6}$$

$$\gamma(\mu) = \limsup_{k \to \infty} \gamma_k(\mu). \tag{5.7}$$

If $\liminf_{\mu \to 0} \alpha(\mu) > 0$, $\limsup_{\mu \to 0} \alpha(\mu) < 1$ and $\beta(\mu) = o(\alpha(\mu))$ as $\mu \to 0$, then

$$\limsup_{k \to \infty} \xi_k = O(\frac{\gamma(\mu)}{\alpha(\mu)}) \;\; \text{as } \mu \to 0. \tag{5.8}$$

Remark : If the convergence in (5.6), (5.7) is uniform (in μ) then so is the convergence in (5.8). Furthermore, if limsup in (5.6), (5.7) can be replaced by sup, then

$$\sup_k \xi_k = O(\frac{\gamma(\mu)}{\alpha(\mu)} + 1) \;\; \text{as } \mu \to 0.$$

5.1 Analysis of Tree Growing : Proof of Theorem 2

To establish Theorem 2 it will be seen that it is necessary to obtain good estimates on the deviation between the true and desired domains where the linear filters are active at each of the nodes. This will require good estimates of the absolute as well as quadratic weight errors. This will in turn require good estimates on the second and fourth order moments of the weights.

We will make use of the following Lemmas. It will be convenient to define

$$1_t(k) = I(\underline{x}(k) \in \chi_t(k)),$$

$$1_t^*(k) = I(\underline{x}(k) \in \chi_t^*).$$

and

$$z_t(k) = 1_t(k) - 1_t^*(k).$$

Lemma 1 : For $t \in T_o$

$$\limsup_{k \to \infty} E\{ \, |\underline{c}_t(k)|^4 + (\theta_t(k))^4 \} = O(\mu^{1-\nu}) \quad \text{as } \mu \to 0 \tag{5.9}$$

Lemma 2 : For $t \in T_o$

$$\limsup_{k \to \infty} E\{ \, |\underline{c}_t(k) - \underline{c}_t^*|^2 + (\theta_t(k) - \theta_t^*)^2 \} = O(\mu^{\beta_t}) \quad \text{as } \mu \to 0 \tag{5.10}$$

where

$$\beta_t = \frac{1}{2}\,(1 - \nu + 4^{-\,\mathrm{depth}(t)})$$

Lemma 3 : For $t \in T_o$

$$\limsup_{k \to \infty} E\{ \, |\underline{c}_t(k) - \underline{c}_t^*| + |\theta_t(k) - \theta_t^*| \} = O(\mu^{\gamma_t}) \quad \text{as } \mu \to 0 \tag{5.11}$$

where

$$\gamma_t = \frac{1}{2}\,(4^{-\,\mathrm{depth}(t)})$$

Lemma 4 : For $t \in T_o$

$$\limsup_{k\to\infty} E\{\,|z_t(k)|\,\} = O(\mu^{2\gamma_t}) \quad \text{as } \mu \to 0 \tag{5.12}$$

Remark : Note that $|z_t(k)| = I(\underline{x}(k) \in \chi_t(k)\,\Delta\,\chi_t^*)$ where Δ is the symmetric difference operator. Hence $E\{\,|z_t(k)|\,\} = P\{\underline{x}(k) \in \chi_t(k)\,\Delta\,\chi_t^*\}$ is the domain error mentioned above.

Lemmas 1-4 are proved below. Assuming these lemmas hold, we prove Theorem 2 as follows.

First consider (5.1). We have using Lemma 1

$$\begin{aligned}
E\{(\hat{y}_T(k))^4\} &= E\{\,(\sum_{t\in\tilde{T}} \hat{y}_t(k)\,1_t(k))^4\}\\
&\le K_1 \max_{t\in T_o} E\{(\hat{y}_t(k))^4\}\\
&\le K_1 \max_{t\in T_o} E\{\,|\underline{x}|^4\}\,E\{\,|\underline{c}_t(k)|^4\}\\
&= O(\mu^{1-\nu})
\end{aligned}$$

as $k \to \infty$ and $\mu \to 0$, as required.

Next consider (5.2). We have using Holder's inequality and Lemmas 2 and 4

$$\begin{aligned}
E\{(\hat{y}_T(k) - \hat{y}^*_T(k))^2\} &= E\{\,(\sum_{t\in\tilde{T}} (\hat{y}_t(k)\,1_t(k) - \hat{y}^*_t(k)\,1^*_t(k)))^2\,\}\\
&\le K_1 \max_{t\in T_o} E\{(\hat{y}_t(k)\,1_t(k) - \hat{y}^*_t(k)\,1^*_t(k))^2\}\\
&\le K_2 \max_{t\in T_o} (E\{(\hat{y}_t(k) - \hat{y}^*_t(k))^2\} + E\{(\hat{y}^*_t(k) z_t(k))^2\})\\
&\le K_2 \max_{t\in T_o} (E\{\,|\underline{x}|^2\}\,E\{\,|\underline{c}_t(k) - \underline{c}^*_t|^2\}\\
&\qquad + |\underline{c}^*_t|^2\,E\{\,|\underline{x}|^4\}^{1/2}\,E\{\,|z_t(k)|^4\}^{1/2})\\
&\le K_3 \max_{t\in T_o} (E\{\,|\underline{c}_t(k) - \underline{c}^*_t|^2\} + E\{\,|z_t(k)|\,\}^{1/2})
\end{aligned}$$

$$= O(\mu^{\beta}) + O(\mu^{\gamma}) = O(\mu^{\beta})$$

as $k \to \infty$ and $\mu \to 0$, as required.

It remains to prove Lemmas 1-4. Note that Lemma 3 was not directly used in the proof of Theorem 2 above but is necessary to establish the other lemmas and in fact will be proved simultaneously with the other lemmas in an induction proof on the levels of the tree T_o.

The proof of Lemmas 1-4 by induction proceeds as follows. We first note that (5.9)-(5.12) are true for the root node t_o. To see this observe that $\underline{c}_{t_o}(k)$ (and similarly $\theta_{t_o}(k)$) are generated by (essentially[†]) the standard LMS algorithm with i.i.d data $\{\underline{x}(k), y(k)\}$ and $E\{|\underline{x}|^8\} < \infty$, $E\{y^8\} < \infty$. Also since $\underline{a}'\underline{x}$ is a continuous random variable for all $\underline{a} \neq$ bf0 we must have $E\{\underline{x}\underline{x}'\} > \underline{0}$. It follows by similar analysis to[61] but considering fourth means instead of quadratic means that

$$\limsup_{k \to \infty} E\{|\underline{c}_{t_o}(k) - \underline{c}_{t_o}^*|^4 + (\theta_{t_o}(k) - \theta_{t_o}^*)^4\} = O(\mu^2) \quad \text{as } \mu \to 0 \qquad (5.13)$$

(5.9)-(5.11) (and trivially (5.12)) at the root node t_o follow from (5.13) and Holder's inequality.

Next suppose (5.9)-(5.12) are true for some nonterminal node $t \in T_o$. Let $s = \text{left}(t)$ or $s = \text{right}(t)$. We will establish the following sequence of propositions.

Proposition 1 :

$$\limsup_{k \to \infty} E\{|z_s(k)|\} = O(\mu^{2\gamma_o}) \quad \text{as } \mu \to 0 \qquad (5.14)$$

Proposition 2 :

$$\sup_{k} E\{|\underline{c}_s(k)|^2 + (\theta_s(k))^2\} = O(\mu^{1-v}) \quad \text{as } \mu \to 0 \qquad (5.15)$$

† the only difference is the μ^{v} term and this does not affect the analysis since $v > 1$

$$\sup_k E\{|\underline{c}_s(k)|^4 + (\theta_s(k))^4\} = O(\mu^{2(1-\nu)}) \text{ as } \mu \to 0 \qquad (5.16)$$

Proposition 3 :

$$\limsup_{k \to \infty} E\{|\underline{c}_s(k) - \underline{c}_s^*|^2 + (\theta_s(k) - \theta_s^*)^2\} = o(1) \text{ as } \mu \to 0 \qquad (5.17)$$

Proposition 4 :

$$\limsup_{k \to \infty} E\{|\underline{c}_s(k)|^2 + (\theta_s(k))^2\} = O(1) \text{ as } \mu \to 0 \qquad (5.18)$$

$$\limsup_{k \to \infty} E\{|\underline{c}_s(k)|^4 + (\theta_s(k))^4\} = O(\mu^{1-\nu}) \text{ as } \mu \to 0 \qquad (5.19)$$

Proposition 5 :

$$\limsup_{k \to \infty} E\{|\underline{c}_s(k) - \underline{c}_s^*|^2 + (\theta_s(k) - \theta_s^*)^2\} = O(\mu^{\beta_s}) \text{ as } \mu \to 0 \qquad (5.20)$$

Proposition 6 :

$$\limsup_{k \to \infty} E\{|\underline{c}_s(k) - \underline{c}_s^*| + |\theta_s(k) - \theta_s^*|\} = O(\mu^{\gamma_s}) \text{ as } \mu \to 0 \qquad (5.21)$$

The idea behind Propositions 1-6 is as follows. First, the estimate of the domain error (5.12) and the absolute weight and threshold errors (5.11) at the parent node t of node s is used to obtain the estimates of the domain error at node s (5.14). Next an estimate of the second moments of the weights and thresholds (5.15) is derived and then used to obtain an estimate of the fourth moments of the weights and thresholds (5.16). Next, the estimates of the domain error (5.14) and the fourth moments of the weights and thresholds (5.16) are used to obtain the convergence in quadratic mean of the weights and thresholds (5.17). Next the convergence in quadratic mean of the weights and thresholds (5.17) is used to obtain a better estimate of the second moments of the weights and thresholds (5.18), and consequently a better estimate of the fourth moments of weights and thresholds (5.19). Finally the estimate of the domain error (5.14) and the refined estimates of the second and fourth moments of the weights and thresholds, (5.18) and (5.19), respectively, are used to obtain the final estimates of the quadratic and absolute weight and threshold errors, (5.20) and (5.21), respectively. It should be noted that the estimates of the absolute weight and threshold errors (5.21) is not simply obtained from the estimate of the quadratic weight and threshold errors (5.20) by, say, Holder's inequality; observe that β_s and not γ_s depends on v in this regard. The above procedure is fairly well optimized with respect to obtaining the weakest conditions and best estimates in Theorem 2 and ultimately Theorem 1.

The proofs of Propositions 1-6 are given below. Assuming that the propositions hold, Lemmas 1-4 follow by induction.

Proof of Proposition 1 :

First observe that $|z_s(k)| = I(\underline{x}(k) \in \chi_s(k) \Delta \chi_s^*)$ where Δ denotes symmetric difference. Hence for any $\delta > 0$

$$\begin{aligned}
E\{|z_s(k)|\} &= P\{\underline{x}(k) \in \chi_s(k) \Delta \chi_s^*\} \\
&\le P\{\underline{x}(k) \in \chi_t(k) \Delta \chi_t^*\} \\
&\quad + P\{(\underline{x}(k) \in \chi_t(k) \cap \chi_t^* \cap (\chi_s(k) \Delta \chi_s^*))\} \\
&\le E\{|z_t(k)|\} + P\{\{\hat{y}_t(k) \le \theta_t(k)\} \cap \{\hat{y}_t^*(k) > \theta_t^*\}\} \\
&\quad + P\{\{\hat{y}_t^*(k) \le \theta_t^*\} \cap \{\hat{y}_t(k) > \theta_t(k)\}\} \\
&\le E\{|z_t(k)|\} + P\{|\hat{y}_t(k) - \theta_t(k) - (\hat{y}_t^*(k) - \theta_t^*)| \ge \delta\} \\
&\quad + P\{|\hat{y}_t(k) - \theta_t(k)| \le \delta\} + P\{|\hat{y}_t^*(k) - \theta_t^*| \le \delta\} \\
&\le E\{|z_t(k)|\} + 2P\{|\hat{y}_t(k) - \theta_t(k) - (\hat{y}_t^*(k) - \theta_t^*)| \ge \delta\}
\end{aligned}$$

$$+ 2P\{\,|\hat{y}^*_t(k) - \theta_t^*\,| \le 2\delta\} \tag{5.22}$$

Now as discussed in the Remark following Assumptions (A1)-(A3) we have $\underline{c}_t^* \neq 0$. Hence $\underline{c}_t^{*\prime}\underline{x}$ has a bounded density and so

$$P\{\,|\hat{y}^*_t(k) - \theta_t^*\,| \le 2\delta\} = O(1)\,\delta \tag{5.23}$$

Also by Markov's inequality

$$P\{\,|\hat{y}_t(k) - \theta_t(k) - (\hat{y}^*_t(k) - \theta_t^*)| > \delta\}$$

$$\le \frac{1}{\delta}\,(E\{\,|\hat{y}_t(k) - \hat{y}^*_t(k)|\,\} + E\{\,|\theta_t(k) - \theta_t^*\,|\})$$

$$\le \frac{1}{\delta}\,(E\{\,|\underline{x}|\,\}E\{\,|\underline{c}_t(k) - \underline{c}_t^*\,|\,\} + E\{\,|\theta_t(k) - \theta_t^*\,|\}).$$

Hence by (5.11)

$$P\{\,|\hat{y}_t(k) - \theta_t(k) - (\hat{y}^*_t(k) - \theta_t^*)| > \delta\} = \frac{O(\mu^{\gamma_t})}{\delta} \tag{5.24}$$

as $k \to \infty$ and $\mu \to 0$.

Substituting (5.12), (5.23), (5.24) into (5.22) gives

$$E\{\,|z_s(k)|\,\} = O(\mu^{2\gamma_t}) + \frac{O(\mu^{\gamma_t})}{\delta} + O(1)\,\delta$$

as $k \to \infty$ and $\mu \to 0$. Choose $\delta = \mu^{\gamma_t/2}$ (this choice will minimize the right hand side of the above equation). Then

$$E\{\,|z_s(k)|\,\} = O(\mu^{\gamma_t/2}) = O(\mu^{2\gamma_s})$$

as $k \to \infty$ and $\mu \to 0$. □

Proof of Proposition 2 :

$\underline{c}_s(k)$ and $\theta_s(k)$ can be treated similarly; we only consider $\underline{c}_s(k)$. We have

$$\underline{c}_s(k{+}1) = (1 - \mu^{v} - \mu\underline{x}(k)\underline{x}(k)'1_s(k))\,\underline{c}_s(k) + \mu y(k)\underline{x}(k)1_s(k)$$

We first claim that for $n = 0, 1, \ldots,$

$$|\underline{c}_s(k+1)|^{2n} \le (1 - 2n\mu^v + o(\mu^v)u(k)^{4n})\, |\underline{c}_s(k)|^{2n} + o(\mu^v)u(k)^{4n} + 2n\mu v(k) \qquad (5.25)$$

as $\mu \to 0$, where

$$u(k) = |\underline{x}(k)| + |y(k)| + 1$$

and

$$v(k) = (-|\underline{x}(k)'\underline{c}_s(k)| + |y(k)|)\, \underline{x}(k)'\underline{c}_s(k) 1_s(k)$$

The details of these expansions are omitted.

Next observe that for $n = 0, 1, \ldots,$

$$\begin{aligned} &E\{v(k)|\underline{c}_s(k)|^n\} \\ &= E\{(-|\underline{x}(k)'\underline{c}_s(k)| + |y(k)|)\, |\underline{x}(k)'\underline{c}_s(k)|\, |\underline{c}_s(k)|^n 1_s(k)\} \\ &\le E\{\, [(-|\underline{x}(k)'\underline{c}_s(k)| + |y(k)|)\, |\underline{x}(k)'\underline{c}_s(k)|\, |\underline{c}_s(k)|^n 1_s(k)]^+ \,\} \\ &\le E\{\, |y(k)|\, |\underline{x}(k)'\underline{c}_s(k)|\, |\underline{c}_s(k)|^n\, I(|\underline{x}(k)'\underline{c}_s(k)| < |y(k)|)\} \\ &\le E\{\, |y(k)|^2\, |\underline{c}_s(k)|^n\} \\ &= E\{|y|^2\}\, E\{|\underline{c}_s(k)|^n\} \end{aligned} \qquad (5.26)$$

Now taking expected values in (5.25) and using (5.26) gives

$$\begin{aligned} E\{|\underline{c}_s(k+1)|^2\} &\le (1 - 2\mu^v + o(\mu^v)E\{u^4\})\, E\{|\underline{c}_s(k)|^2\} \\ &\quad + o(\mu^v)E\{u^4\} + 2\mu E\{y^2\} \\ &= (1 - 2\mu^v + o(\mu^v))\, E\{|\underline{c}_s(k)|^2\} + O(\mu) \end{aligned}$$

as $\mu \to 0$. Here we have used the independence of $u(k)$ and $\underline{c}_s(k)$, and also $E\{u(k)^4\} = E\{u^4\} \le K_1\, (E\{|\underline{x}|^4\} + E\{y^4\}) < \infty$. It follows from the Remark following the Fact that

$$\sup_k E\{|\underline{c}_s(k)|^2\} = O(\mu^{1-\nu}) \quad \text{as } \mu \to 0. \tag{5.27}$$

Now taking expected values in (5.25) and using (5.26) and (5.27) gives

$$\begin{aligned} E\{|\underline{c}_s(k+1)|^4\} &\le (1 - 4\mu^\nu + o(\mu^\nu)E\{u^8\})\, E\{|\underline{c}_s(k)|^4\} \\ &\quad + o(\mu^\nu)E\{u^8\} + 4\mu E\{y^2\}E\{|\underline{c}_s(k)|^2\} \\ &= (1 - 4\mu^\nu + o(\mu^\nu))\, E\{|\underline{c}_s(k)|^4\} + O(\mu^{2-\nu}) \end{aligned} \tag{5.28}$$

as $\mu \to 0$. Here we have again used the independence of u(k) and $\underline{c}_s(k)$, and also $E\{u(k)^8\} = E\{u^8\} \le K_2\,(E\{|\underline{x}|^8\} + E\{y^8\}) < \infty$. It now follows from the Remark following the Fact that

$$\sup_k E\{|\underline{c}_s(k)|^4\} = O(\frac{\mu^{2-\nu}}{\mu^\nu}) = O(\mu^{2(1-\nu)}) \text{ as } \mu \to 0$$

□

Proof of Proposition 3 :

$\underline{c}_s(k)$ and $\theta_s(k)$ can be treated separately and similarly; we only consider $\underline{c}_s(k)$. We have

$$\underline{c}_s(k+1) = (1 - \mu^\nu - \mu\underline{x}(k)\underline{x}(k)'1_s(k))\, \underline{c}_s(k) + \mu y(k)\underline{x}(k)1_s(k) \tag{5.29}$$

Let

$$\underline{c}_s^*(k+1) = (1 - \mu^\nu - \mu\underline{x}(k)\underline{x}(k)'1_s^*(k))\, \underline{c}_s^*(k) + \mu y(k)\underline{x}(k)1_s^*(k) \tag{5.30}$$

and define

$$\underline{e}_s(k) = \underline{c}_s(k) - \underline{c}_s^*(k) \tag{5.31}$$

$$\underline{e}_s^*(k) = \underline{c}_s^*(k) - \underline{c}_s^* \tag{5.32}$$

We first consider $\underline{e}_s^*(k)$. Observe that $\underline{c}_s^*(k)$ is generated by (essentially) the standard LMS algorithm with i.i.d data $\{\underline{x}(k)\,1_s^*(k), y(k)\} = \{\text{input}, \text{output}\}$, and $E\{|\underline{x}|^4\, 1_s^*\} < \infty$, $E\{y^4\} < \infty$. Also, since $\underline{a}'\underline{x}$ is a continuous random variable for all $\underline{a} \neq \underline{0}$ and $p_s^* > 0$ we must have $E\{\underline{x}\underline{x}'1_s^*\} > \underline{0}$. It follows by a similar analysis to[61] that

$$\limsup_{k\to\infty} E\{|\underline{e}_s^*(k)|^2\} = O(\mu) \quad \text{as } \mu \to 0 \tag{5.33}$$

We next consider $\underline{e}_s(k)$. We have

$$\underline{e}_s(k+1) = (1 - \mu^v - \mu\underline{x}(k)\underline{x}(k)'1_s^*(k))\,\underline{e}_s(k) \tag{5.34}$$

$$- \mu\underline{x}(k)\underline{x}(k)'z_s(k)\underline{c}_s(k) + \mu y(k)\underline{x}(k)z_s(k)$$

Hence

$$|\underline{e}_s(k+1)|^2 \leq (1 + o(\mu)u(k)^4)\,|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'\underline{x}(k)\underline{x}(k)'\underline{e}_s(k)1_s^*(k)$$

$$+ O(\mu)u(k)^4\ |z_s(k)|\ (|\underline{c}_s(k)|^2 + 1) \tag{5.35}$$

as $\mu \to 0$, where

$$u(k) = 1 + |\underline{x}(k)| + |y(k)|$$

Let λ_s be the smallest eigenvalue of $E\{\underline{x}\underline{x}'1_s^*\} > \underline{0}$. Since $\underline{e}_s(k)$ is $\mathcal{F}_k$ measurable and $\underline{x}(k)$, $y(k)$, $1_s^*(k)$ are independent of $\mathcal{F}_k$

$$E\{(1 + o(\mu)u(k)^4\,|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'\underline{x}(k)\underline{x}(k)'\underline{e}_s(k)1_s^*(k)\}$$

$$= (1 + o(\mu)E\{u^4\})\,E\{|\underline{e}_s(k)|^2\} - 2\mu E\{\underline{e}_s(k)'E\{\underline{x}\underline{x}'1_s^*\}\underline{e}_s(k)\}$$

$$\leq (1 - 2\mu\lambda_s + o(\mu))\,E\{|\underline{e}_s(k)|^2\} \tag{5.36}$$

Hence taking expected values in (5.35) and using (5.36) and Holder's inequality gives

$$E\{|\underline{e}_s(k+1)|^2\} \leq (1 - 2\mu\lambda_s + o(\mu))\,E\{|\underline{e}_s(k)|^2\}$$

$$+ O(\mu)\,E\{u^8\}^{1/2}\,E\{|z_s(k)|^2\}^{1/2}\,(E\{|\underline{c}_s(k)|^4\} + 1)^{1/2}$$

$$\leq (1 - 2\mu\lambda_s + o(\mu))\, E\{ |\underline{e}_s(k)|^2 \}$$

$$+ O(\mu)\, E\{ |z_s(k)| \}^{1/2} (E\{ |\underline{c}_s(k)|^4 \}^{1/2} + 1) \qquad (5.37)$$

as $\mu \to 0$. Now using Propositions 1 and 2 we have

$$E\{ |\underline{e}_s(k+1)|^2 \} \leq (1 - 2\mu\lambda_s + o(\mu))\, E\{ |\underline{e}_s(k)|^2 \} + O(\mu^{2-\nu+\gamma_s})$$

as $k \to \infty$ and $\mu \to 0$. It follows from the Fact that

$$\limsup_{k \to \infty} E\{ |\underline{e}_s(k)|^2 \} = O(\frac{\mu^{2-\nu+\gamma_s}}{\mu}) = O(\mu^{1-\nu+\gamma_s}) \text{ as } \mu \to 0$$

Since $\nu \leq 1 + (1/5)\, 4^{-\text{depth}(T_s)}$, we have $1 - \nu + \gamma_s > 0$ and so

$$\limsup_{k \to \infty} E\{ |\underline{e}_s(k)|^2 \} = o(1) \quad \text{as } \mu \to 0 \qquad (5.38)$$

Finally, combining (5.33) and (5.38) gives

$$\limsup_{k \to \infty} E\{ |\underline{c}_s(k) - \underline{c}_s^*|^2 \} \leq K_1 \limsup_{k \to \infty} (E\{ |\underline{e}_s(k)|^2 \} + E\{ |\underline{e}_s^*(k)|^2 \})$$

$$= o(1) \quad \text{as } \mu \to 0$$

□

Proof of Proposition 4 :

(5.18) follows immediately from Proposition 3.

(5.19) can be established by treating $\underline{c}_s(k)$ and $\theta_s(k)$ separately and similarly; we only consider $\underline{c}_s(k)$. Combining (5.18) and (5.28) gives

$$E\{ |\underline{c}_s(k+1)|^4 \} = (1 - 4\mu^\nu + o(\mu^\nu))\, E\{ |\underline{c}_s(k)|^4 \} + O(\mu)$$

as $k \to \infty$ and $\mu \to 0$. It follows from the Fact that

$$\limsup_{k \to \infty} E\{ |\underline{c}_s(k)|^4 \} = O(\frac{\mu}{\mu^\nu}) = O(\mu^{1-\nu}) \text{ as } \mu \to 0$$

□

Proof of Proposition 5 :

$\underline{c}_s(k)$ and $\theta_s(k)$ can be treated separately and similarly; we only consider $\underline{c}_s(k)$. Define $\underline{e}_s(k)$ and $\underline{e}_s^*(k)$ as in (5.31) and (5.32). Combining Proposition 1, Proposition 4 and (5.37) gives

$$E\{|\underline{e}_s(k+1)|^2\} \le (1-2\mu\lambda_s + o(\mu))\, E\{|\underline{e}_s(k)|^2\} + O(\mu^{(3-v+2\gamma_s)/2})$$

as $k \to \infty$ and $\mu \to 0$. It follows from the Fact that

$$\limsup_{k\to\infty} E\{|\underline{e}_s(k)|^2\} = O(\frac{\mu^{(3-v+2\gamma_s)/2}}{\mu}) = O(\mu^{\beta_s}) \text{ as } \mu \to 0 \tag{5.39}$$

Hence combining (5.33) and (5.39) gives

$$\limsup_{k\to\infty} E\{|\underline{c}_s(k) - \underline{c}_s^*|^2\} \le K_1 \limsup_{k\to\infty} (E\{|\underline{e}_s(k)|^2\} + E\{|\underline{e}_s^*(k)|^2\})$$

$$= O(\mu^{\beta_s}) \text{ as } \mu \to 0$$

since $\beta_s < 1$. □

Proof of Proposition 6 :

$\underline{c}_s(k)$ and $\theta_s(k)$ can be treated separately and similarly; we only consider $\underline{c}_s(k)$. Define $\underline{e}_s(k)$ and $\underline{e}_s^*(k)$ as in (5.31) and (5.32).

From (5.34) we have

$$|\underline{e}_s(k+1)| \le [(1+o(\mu)u(k)^4)\,|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'\underline{x}(k)\underline{x}(k)'\underline{e}_s(k)1_s^*(k)]^{1/2}$$

$$+ O(\mu)u(k)^2\,|z_s(k)|(|\underline{c}_s(k)|+1) \tag{5.40}$$

as $\mu \to 0$, where

$$u(k) = 1 + |\underline{x}(k)| + |y(k)|$$

Let $\lambda_s > 0$ be the smallest eigenvalue of $E\{\underline{x}\underline{x}'1_s^*\} > \underline{0}$. Since $\underline{e}_s(k)$ is $\mathfrak{F}_k$ measurable and $\underline{x}(k)$, $y(k)$, $1_s^*(k)$ are independent of $\mathfrak{F}_k$ we can apply the conditional Jensen's inequality to get

$$E\{[(1+o(\mu)u(k)^4)|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'\underline{x}(k)\underline{x}(k)'\underline{e}_s(k)1_s^*(k)]^{1/2}\}$$

$$\leq E\{[E\{(1+o(\mu)u(k)^4)|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'\underline{x}(k)\underline{x}(k)'\underline{e}_s(k)1_s^*(k) \mid \mathcal{F}_k\}]^{1/2}\}$$

$$= E\{[(1+o(\mu)E\{u^4\})|\underline{e}_s(k)|^2 - 2\mu\underline{e}_s(k)'E\{\underline{x}\underline{x}'1_s^*\}\underline{e}_s(k)]^{1/2}\}$$

$$\leq (1-2\mu\lambda_s + o(\mu))^{1/2}\, E\{|\underline{e}_s(k)|\}$$

$$= (1-\mu\lambda_s + o(\mu))\, E\{|\underline{e}_s(k)|\} \tag{5.41}$$

as $\mu \to 0$ (in the last step we use $(1+\eta)^{1/2} = 1+\eta/2+o(\eta)$ as $\eta \to 0$). Hence taking expected values in (5.40) and using (5.41) and Holder's inequality gives

$$E\{|\underline{e}_s(k+1)|\} \leq (1-\mu\lambda_s + o(\mu))\, E\{|\underline{e}_s(k)|\}$$

$$+ O(\mu)E\{u^4\}^{1/2}\, E\{|z_s(k)|^2\}^{1/2}\, (E\{|\underline{c}_s(k)|^2\}+1)^{1/2}$$

$$\leq (1-\mu\lambda_s + o(\mu))\, E\{|\underline{e}_s(k)|\}$$

$$+ O(\mu)\, E\{|z_s(k)|\}^{1/2}\, (E\{|\underline{c}_s(k)|^2\}^{1/2}+1)$$

as $\mu \to 0$. Now using Proposition 1 and Proposition 4 we have

$$E\{|\underline{e}_s(k+1)|\} \leq (1-\mu\lambda_s + o(\mu))\, E\{|\underline{e}_s(k)|\} + O(\mu^{1+\gamma_s})$$

as $k \to \infty$ and $\mu \to 0$. It follows from the Fact that

$$\limsup_{k\to\infty} E\{|\underline{e}_s(k)|\} = O(\frac{\mu^{1+\gamma_s}}{\mu}) = O(\mu^{\gamma_s}) \text{ as } \mu \to 0 \tag{5.42}$$

Finally, combining (5.33) and (5.42) gives

$$\limsup_{k\to\infty} E\{|\underline{c}_t(k) - \underline{c}_t^*|\} \leq \limsup_{k\to\infty} (E\{|\underline{e}_s(k)|\} + E\{|\underline{e}_s^*(k)|^2\}^{1/2})$$

$$= O(\mu^{\gamma_s}) \text{ as } \mu \to 0$$

since $\gamma_s \leq 1/2$. □

5.2 Analysis of Tree Pruning : Proof of Theorem 3

To establish Theorem 3 we will make use of the following lemmas.

Lemma 5 : For $t \in T_o$

$$\limsup_{k \to \infty} E\{\,|\varepsilon_t(k)|^2 + |p_t(k)|^2\} = O(\mu^{2(1-\nu)}) \text{ as } \mu \to 0$$

Lemma 6 : For $t \in T_o$

$$\limsup_{k \to \infty} E\{\,|\varepsilon_t(k) - \varepsilon_t^*| + |p_t(k) - p_t^*|\,\} = O(\mu^{\beta_1/2}) \text{ as } \mu \to 0$$

Lemmas 5 and 6 are proved below. Assuming these lemmas hold we prove Theorem 3 as follows.

Observe that since the number of pruned subtrees of T_o is finite, there exists a $\delta > 0$ such that if $\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}$ then $\varepsilon^*_{T(k)} > \varepsilon^*_{T^*} + \delta$. Hence

$$P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\} = P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*} + \delta\}$$

$$= P\{\varepsilon^*_{T(k)} - \varepsilon_{T(k)}(k) > \varepsilon^*_{T^*} - \varepsilon_{T(k)}(k) + \delta\}$$

$$\le P\{\varepsilon^*_{T(k)} - \varepsilon_{T(k)}(k) > \varepsilon^*_{T^*} - \varepsilon_{T^*}(k) + \delta\}$$

$$\le P\{\,|\varepsilon^*_{T(k)} - \varepsilon_{T(k)}(k)| + |\varepsilon^*_{T^*} - \varepsilon_{T^*}(k)| > \delta\}$$

$$\le P\{\sum_{t \in T_o} |\varepsilon_t^* p_t^* - \varepsilon_t(k) p_t(k)| > \frac{\delta}{2}\}$$

$$\le K_1 \max_{t \in T_o} P\{\,|\varepsilon_t^* p_t^* - \varepsilon_t(k) p_t(k)| > \frac{\delta}{2|T_o|}\}$$

$$\le K_1 \max_{t \in T_o} P\{\,|\varepsilon_t(k) - \varepsilon_t^*| + |p_t(k) - p_t^*| > \frac{\delta}{2|T_o|(1 + |\varepsilon_t^*|)}\}$$

since $|p_t(k)| \le 1$. Hence by Markov's inequality

$$P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\} \le \frac{K_2}{\delta} \max_{t \in T_o} E\{\,|\varepsilon_t(k) - \varepsilon_t^*| + |p_t(k) - p_t^*|\,\}$$

as $k \to \infty$ and $\mu \to 0$. Finally by Lemma 6

$$\limsup_{k \to \infty} P\{\varepsilon^*_{T(k)} > \varepsilon^*_{T^*}\} = O(\mu^{\beta/2}) \text{ as } \mu \to 0$$

It remains to prove Lemmas 5 and 6. Note that Lemma 5 was not directly used in the proof of Theorem 3, but will be required to establish Lemma 6.

Proof of Lemma 5 :

Since $|p_t(k)| \le 1$ we need only consider $\varepsilon_t(k)$. We have

$$\varepsilon_t(k+1) = (1 - \mu^v - \mu 1_t(k))\, \varepsilon_t(k) + \mu \tilde{y}_t(k)^2\, 1_t(k)$$

where $\tilde{y}_t(k) = y(k) - \hat{y}_t(k)$.

We first claim that

$$\varepsilon_t(k+1)^2 \le (1 - 2\mu^v + o(\mu))\, \varepsilon_t(k)^2 + o(\mu^v)\, (\tilde{y}_t(k)^4 + 1) + 2\mu v(k) \tag{5.43}$$

as $\mu \to 0$, where

$$v(k) = (-|\varepsilon_t(k)| + \tilde{y}_t(k)^2)\, |\varepsilon_t(k)|\, 1_t(k).$$

The details of this expansion are omitted.

Next observe that

$$\begin{aligned} E\{v(k)\} &= E\{(-|\varepsilon_t(k)| + \tilde{y}_t(k)^2)\, |\varepsilon_t(k)|\, 1_t(k)\} \\ &\le E\{[(-|\varepsilon_t(k)| + \tilde{y}_t(k)^2)\, |\varepsilon_t(k)|\, 1_t(k)]^+\} \\ &\le E\{\tilde{y}_t(k)^2\, |\varepsilon_t(k)|\, I(|\varepsilon_t(k)| < \tilde{y}_t(k)^2)\} \\ &\le E\{\tilde{y}_t(k)^4\} \\ &\le K_1(E\{y^4\} + E\{\hat{y}_t(k)^4\}) \\ &\le K_2\, (1 + E\{|\underline{c}_t(k)|^4\}) \end{aligned}$$

Hence by Lemma 1

$$E\{v(k)\} = O(\mu^{1-v}) \tag{5.44}$$

as $k \to \infty$ and $\mu \to 0$.

Now taking expected values in (5.43) and using (5.44) gives

$$E\{\varepsilon_t(k+1)^2\} \le (1 - 2\mu^v + o(\mu))\, E\{\varepsilon_t(k)^2\} + O(\mu^{2-v})$$

as $k \to \infty$ and $\mu \to 0$. It follows from the Fact that

$$\limsup_{k \to \infty} E\{\varepsilon_t(k)^2\} = O(\frac{\mu^{2-v}}{\mu^v}) = O(\mu^{2(1-v)}) \quad \text{as } \mu \to 0$$

□

Proof of Lemma 6 :

$\varepsilon_t(k)$ and $p_t(k)$ can be treated separately and similarly; we only consider $\varepsilon_t(k)$†.

We have

$$\varepsilon_t(k+1) = (1 - \mu^v - \mu 1_t(k))\, \varepsilon_t(k) + \mu \tilde{y}_t(k)^2\, 1_t(k)$$

where $\tilde{y}_t(k) = y(k) - \hat{y}_t(k)$. Let

$$\varepsilon_t^*(k+1) = (1 - \mu^v - \mu 1_t^*(k))\, \varepsilon_t^*(k) + \mu \tilde{y}_t^*(k)^2\, 1_t^*(k)$$

where $\tilde{y}_t^*(k) = y(k) - \hat{y}_t^*(k)$. Define

$$e_t(k) = \varepsilon_t(k) - \varepsilon_t^*(k)$$

$$e_t^*(k) = \varepsilon_t^*(k) - \varepsilon_t^*$$

We first consider $e_t^*(k)$. Observe that $\varepsilon_t^*(k)$ is generated by (essentially) the standard LMS algorithm with i.i.d data $\{1_t^*(k), \tilde{y}_t^*(k)^2\} = \{\text{input, output}\}$ and $E\{|\tilde{y}_t^*|^8\} \le K_1 E\{|\underline{x}|^8\} < \infty$. Also $E\{1_t^*\} = p_t^* > 0$. It follows by similar analysis to[61] that

$$\limsup_{k \to \infty} E\{|e_t^*(k)|^2\} = O(\mu) \quad \text{as } \mu \to 0 \tag{5.45}$$

We next consider $e_t(k)$. We have

† In fact, since $p_t(k)$ is uniformly bounded a.s., the analysis of $p_t(k)$ is easier

$$e_t(k+1) = (1 - \mu^v - \mu 1_t^*(k))\, e_t(k) - \mu z_t(k)\varepsilon_t(k)$$
$$+ \mu \tilde{y}_t^*(k)^2 z_t(k) + \mu(\tilde{y}_t(k)^2 - \tilde{y}_t^*(k)^2) 1_t(k)$$

Hence

$$|e_t(k+1)| \le (1 - 2\mu 1_t^*(k) + o(\mu))^{1/2}\ |e_t(k)|$$
$$+ O(\mu)u(k)^2 |z_t(k)| (|\varepsilon_t(k)| + 1) + O(\mu) |\tilde{y}_t(k)^2 - \tilde{y}_t^*(k)^2| \qquad (5.46)$$

as $\mu \to 0$, where

$$u(k) = 1 + |\underline{x}(k)| + |y(k)|$$

By Jensen's inequality

$$E\{ (1 - 2\mu 1_t^*(k) + o(\mu))^{1/2} \} \le E\{(1 - 2\mu 1_t^*(k) + o(\mu))\}^{1/2}$$
$$= (1 - 2\mu p_t^* + o(\mu))^{1/2}$$
$$= 1 - \mu p_t^* + o(\mu) \qquad (5.47)$$

as $\mu \to 0$ (in the last step we use $(1+\eta)^{1/2} = 1 + \eta/2 + o(\eta)$ as $\eta \to 0$). Hence taking expected values in (5.46) and using (5.47) and Holder's inequality gives

$$E\{|e_t(k+1)|\} \le (1 - \mu p_t^* + o(\mu))\, E\{|e_t(k)|\}$$
$$+ O(\mu) E\{u^4\}^{1/2} E\{|z_t(k)|^2\}^{1/2} (E\{\varepsilon_t(k)^2\} + 1)^{1/2}$$
$$+ O(\mu)\, E\{|\tilde{y}_t(k)^2 - \tilde{y}_t^*(k)^2|\}$$
$$\le (1 - \mu p_t^* + o(\mu))\, E\{|e_t(k)|\}$$
$$+ O(\mu)\, E\{|z_t(k)|\}^{1/2} (E\{\varepsilon_t(k)^2\}^{1/2} + 1)$$
$$+ O(\mu)\, E\{|\tilde{y}_t(k)^2 - \tilde{y}_t^*(k)^2|\} \qquad (5.48)$$

as $\mu \to 0$. But by Holder's inequality and Lemma 2

$$E\{|\tilde{y}_t(k)^2 - \tilde{y}_t^*(k)^2|\}$$
$$\le E\{|\tilde{y}_t(k) + \tilde{y}_t^*(k)||\tilde{y}_t(k) - \tilde{y}_t^*(k)|\}$$

$$\le K_1\,(E\{y^2\}^{1/2} + E\{|\hat{y}_t(k)|^2\}^{1/2} + E\{|\hat{y}^*_t(k)|^2\}^{1/2})\,E\{(\hat{y}_t(k) - \hat{y}^*_t(k))^2\}^{1/2}$$

$$\le K_2\,(1 + E\{|\underline{c}_t(k)|^2\}^{1/2})\,E\{|\underline{c}_t(k) - \underline{c}_t^*|^2\}^{1/2}$$

$$= O(\mu^{\beta_t/2}) \tag{5.49}$$

as $k \to \infty$ and $\mu \to 0$. Hence combining (5.48) and (5.49) and using Lemmas 4 and 5 we get

$$E\{|e_t(k+1)|\} \le (1 - \mu p_t^* + o(\mu))\,E\{|e_t(k)|\}$$
$$+ O(\mu^{2-\nu+\gamma_t}) + O(\mu^{1+\beta_t/2})$$

as $k \to \infty$ and $\mu \to 0$. Since $\nu < 1 + (1/5)\,4^{-\mathrm{depth}(T_o)}$ it is easily verified that $1 - \nu + \gamma_t > \beta_t/2$. Hence

$$E\{|e_t(k+1)|\} \le (1 - \mu p_t^* + o(\mu))\,E\{|e_t(k)|\} + O(\mu^{1+\beta_t/2})$$

as $k \to \infty$ and $\mu \to 0$. It follows by the Fact that

$$\limsup_{k\to\infty} E\{|e_t(k)|\} = O(\frac{\mu^{1+\beta_t/2}}{\mu}) = O(\mu^{\beta_t/2}) \quad \text{as } \mu \to 0 \tag{5.50}$$

Finally, combining (5.45) and (5.50) gives

$$\limsup_{k\to\infty} E\{|\varepsilon_t(k) - \varepsilon_t^*|\} \le \limsup_{k\to\infty} (E\{|e_t(k)|\} + E\{|e_t^*(k)|^2\}^{1/2})$$
$$= O(\mu^{\beta_t/2}) \quad \text{as } \mu \to 0$$

since $\beta_t < 1$. □

CHAPTER 6
EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter we present results of computer simulations which compare the performance of the proposed tree-structured piecewise linear adaptive filter with that of existing linear and polynomial (truncated Volterra series) adaptive filters in the applications of channel equalization, echo cancellation and speech coding. The tree-structured filters were trained using the the proposed algorithm in Chapter 4, and the linear and polynomial filters were trained using the LMS-type algorithms described in Chapter 2. We also compare the computational complexity and convergence rate of the tree-structured filter and the polynomial filters.

6.1 Channel Equalization Application

In this section we present results of computer simulations which compare the performance of a tree-structured piecewise linear equalizer with that of linear and polynomial equalizers, and also the performance of a tree-structured piecewise decision feedback equalizer with that of a standard decision feedback equalizer.

In our simulations[62], we considered binary PAM signaling through an equivalent discrete-time channel with z-transform $G(z) = 0.227 + 0.460z^{-1} + 0.688z^{-2} + 0.460z^{-3} + 0.227z^{-4}$. This is a reasonable model for a channel encountered in communication systems with severe ISI ($|G(e^{j\omega})|$ has deep nulls) [7]. The channel is corrupted by additive white gaussian noise, which is independent of the information sequence. Each information symbol is ± 1 (independently) with probability 0.5.

A filter of length $L = 11$ was used for all equalizers. For the polynomial equalizers, polynomials of order $q \leq 3$ were considered. For the tree equalizers, a full tree of depth $d = 5$ was chosen as the basic tree T_o (these choices will be discussed further). All of the simulation results to be presented were obtained by averaging over an ensemble of 200 equalizers with independent data.

In Figures 18 and 19 we show the output MSE as a function of time (up to 5000 iterations) for the linear, second order polynomial, third order polynomial and tree-structured piecewise linear equalizers for SNRs of 10 dB and 20 dB. For each type of equalizer and SNR, the gain μ was chosen to be about one-half the value where unstable behavior first occurred. In Figure 20 we show the probability of error as a function of SNR after about 40,000 iterations when all of the equalizers have converged. We make several observations. First, it is clear that a nonlinear equalizer is necessary to achieve a small asymptotic probability of error. Second, it is also clear that higher order polynomial equalizers have lower asymptotic probability of error but converge more slowly (more precisely, higher order polynomial equalizers require much smaller gains μ for stability which limits the rate of convergence). Indeed, the second and third order polynomial equalizers have a higher output MSE than the linear equalizer even after 5000 iterations, although their asymptotic error rates are much smaller. Next, it is seen that the tree-structured equalizer initially converges about as fast as the linear equalizer and from then on converges much faster and to a much lower probability of error. Finally, it is seen that the tree-structured piecewise linear equalizer converges much faster and to a significantly lower probability of error than the second or third order polynomial equalizers. In fact, it is verified that no

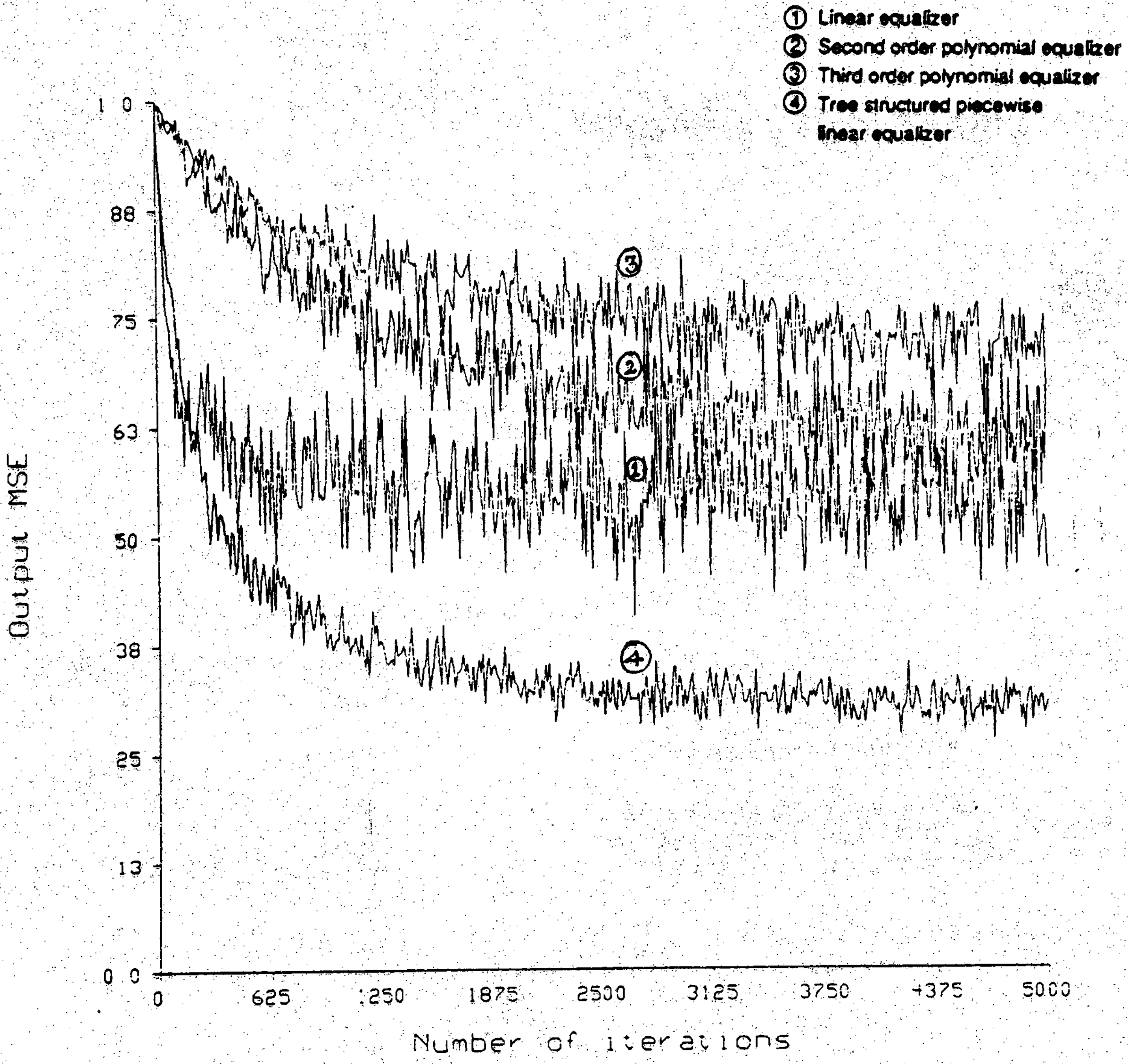


Figure 18. Learning Curves for SNR of 10dB

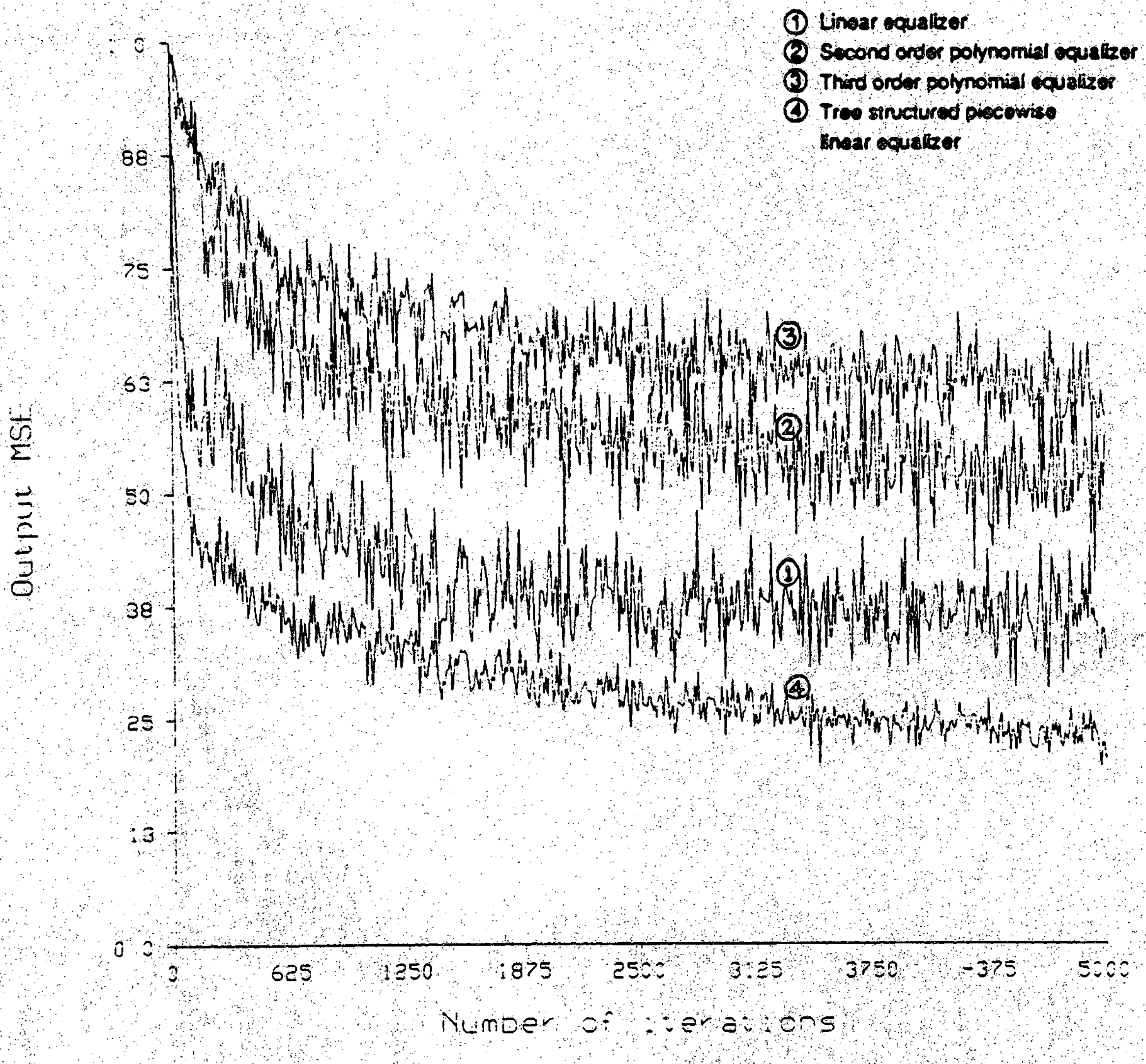


Figure 19. Learning Curves for SNR of 20dB

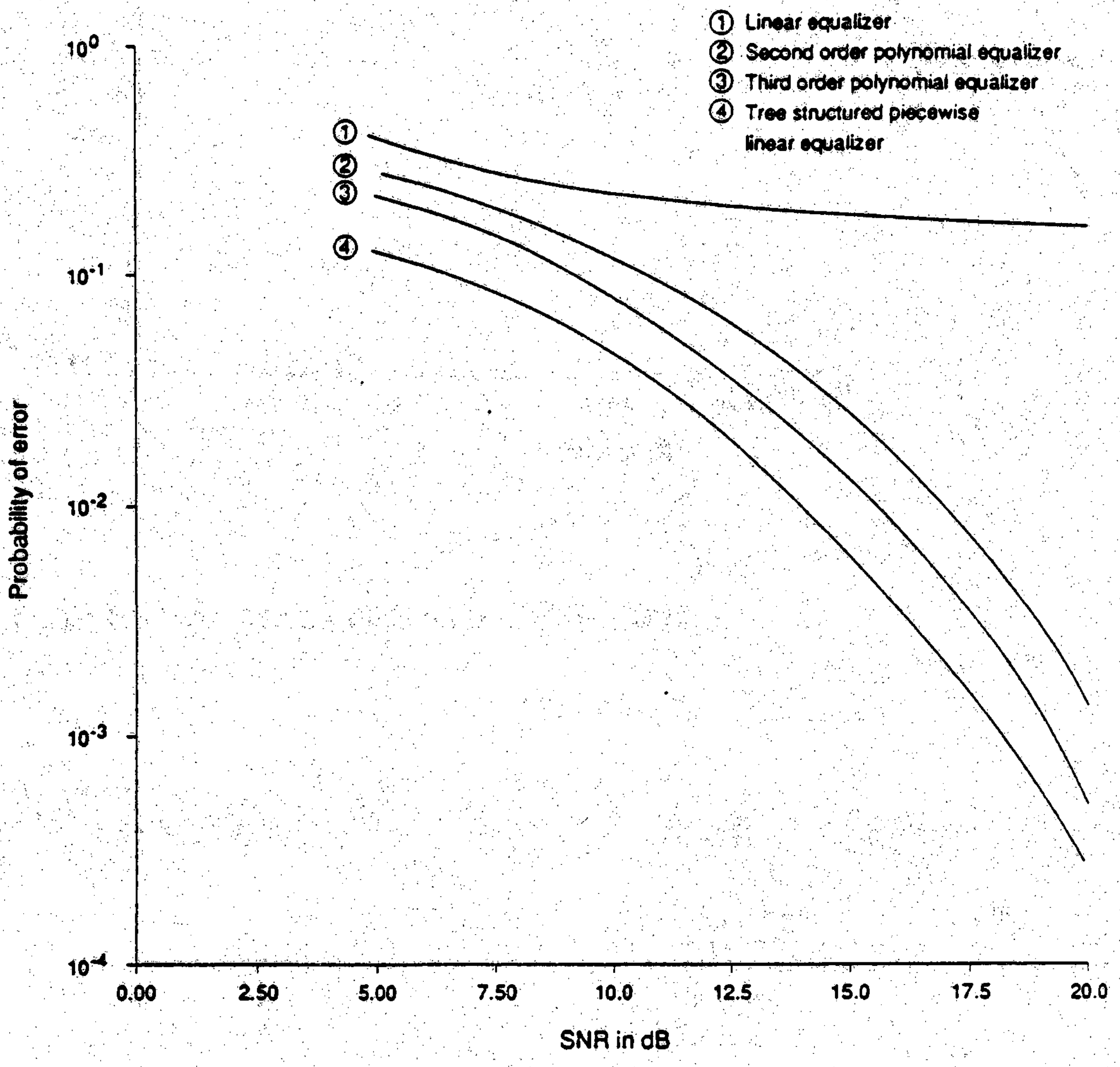


Figure 20. Asymptotic Probability of Error Curves

combination of the linear, second and third order polynomial equalizers performs as well as the tree-structured piecewise linear equalizer, either in terms of rate of convergence or asymptotic probability of error. Fourth and higher order polynomials exhibited extremely high computational complexity and low rates of convergence. A detailed discussion comparing computational complexity and convergence rates of the tree-structured piecewise linear filter versus polynomial filters is given in Section 6.4.

In Figures 21 and 22 we show the output MSE as a function of time (up to 5000 iterations) for the decision feedback and tree-structured piecewise decision feedback equalizers for SNRs of 10 dB and 20 dB. For each equalizer and SNR, the gain μ was again chosen to be about one-half the value where unstable behavior first occurred. In Figure 23 we show the probability of error as a function of SNR after about 40,000 iterations when both of the equalizers have converged. It is seen that the tree-structured piecewise decision feedback equalizer initially converges about as fast as the decision feedback equalizer and from then on converges much faster and to a significantly lower probability of error (about 6dB for an error probability of 10^{-1} and about 2dB for an error probability of 10^{-2}).

6.2 Echo Cancellation Application

In this section we present results of computer simulations which compare the performance of a tree-structured piecewise linear adaptive echo canceler with that of linear, second-order and third-order polynomial types of adaptive echo cancelers [12, 17, 63].

In our simulations [64] we considered an echo path which consisted of a memoryless nonlinear system followed by a linear and another memoryless nonlinear

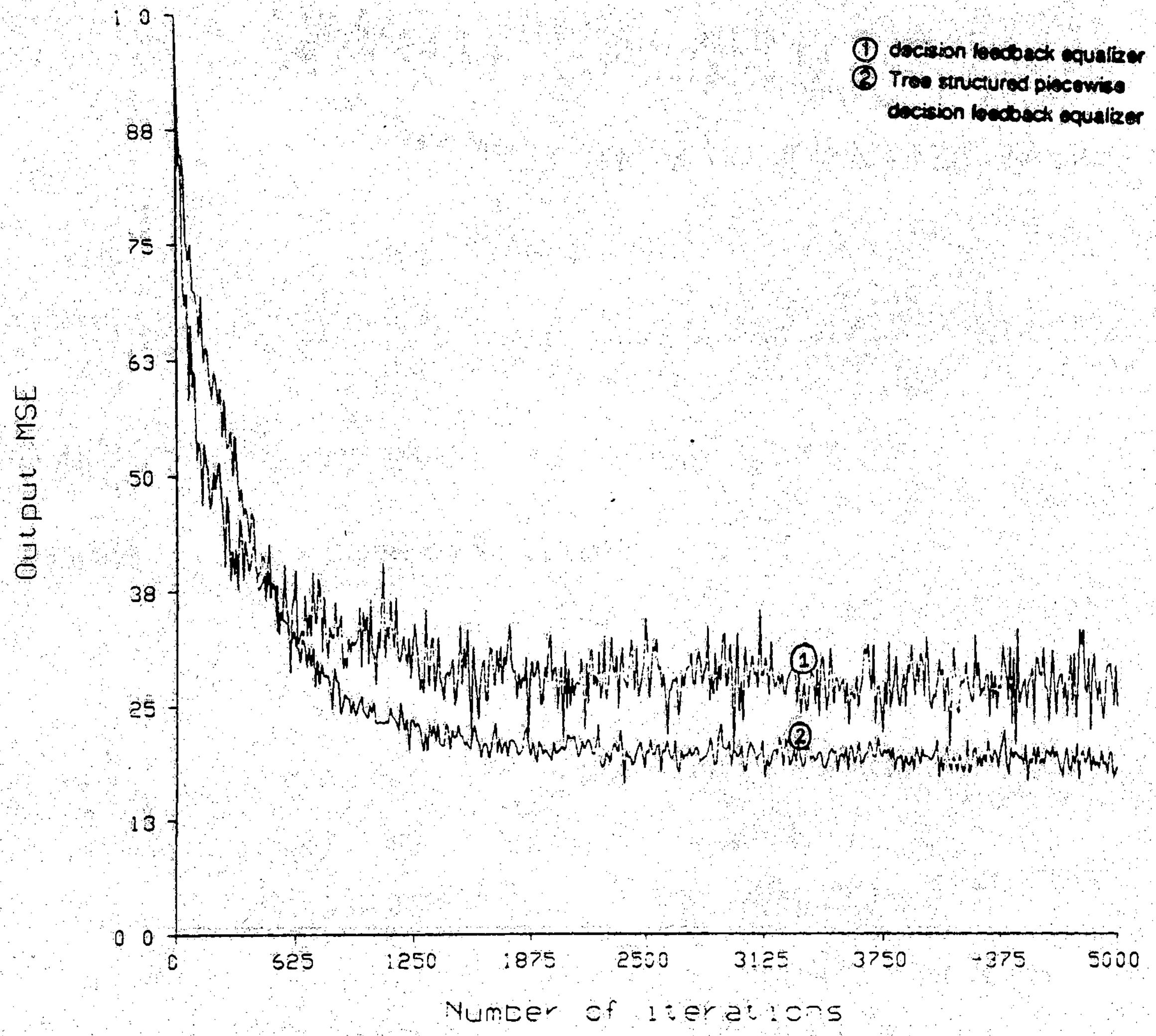


Figure 21. Learning Curves for SNR of 10dB

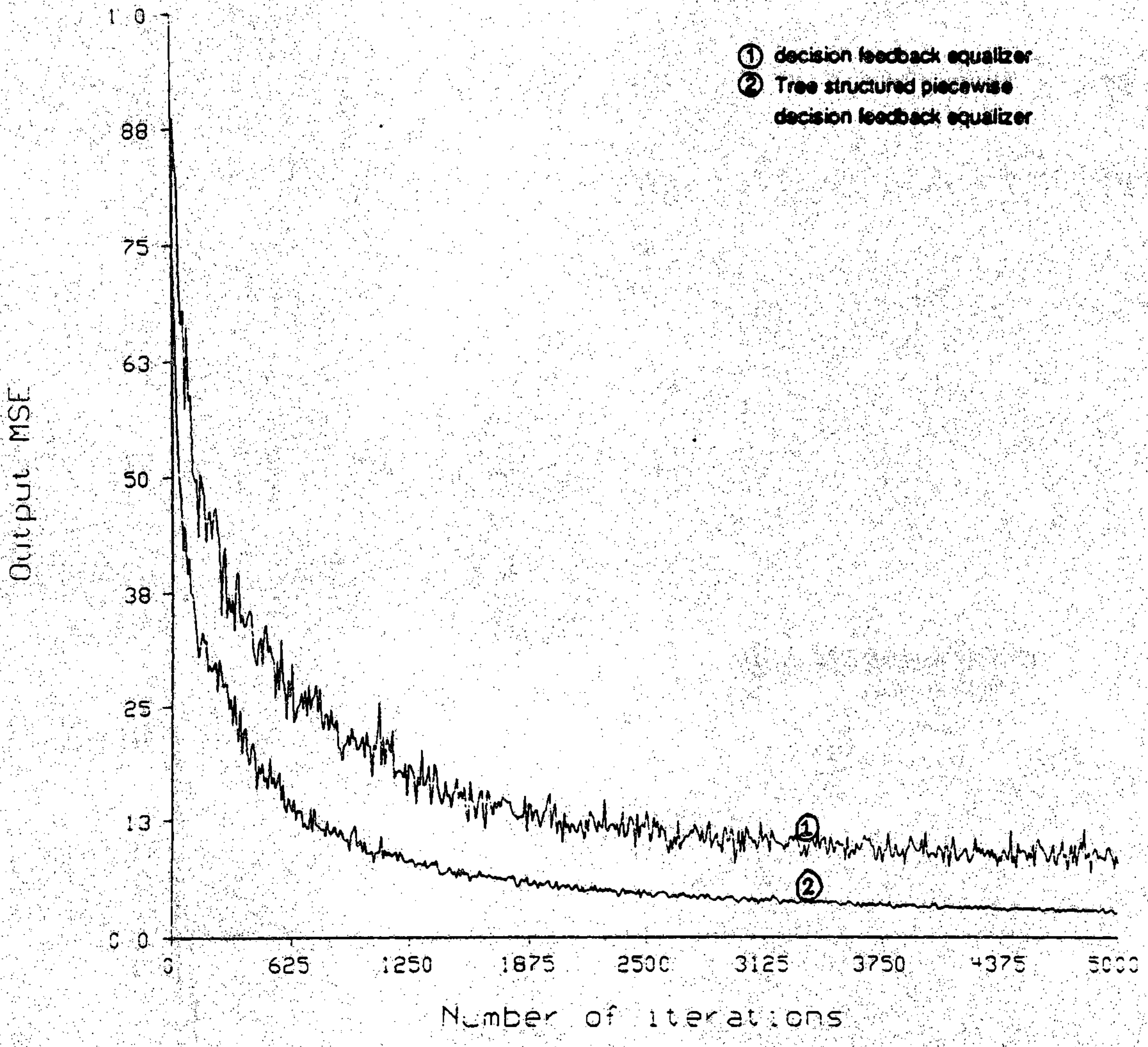


Figure 22. Learning Curves for SNR of 20dB

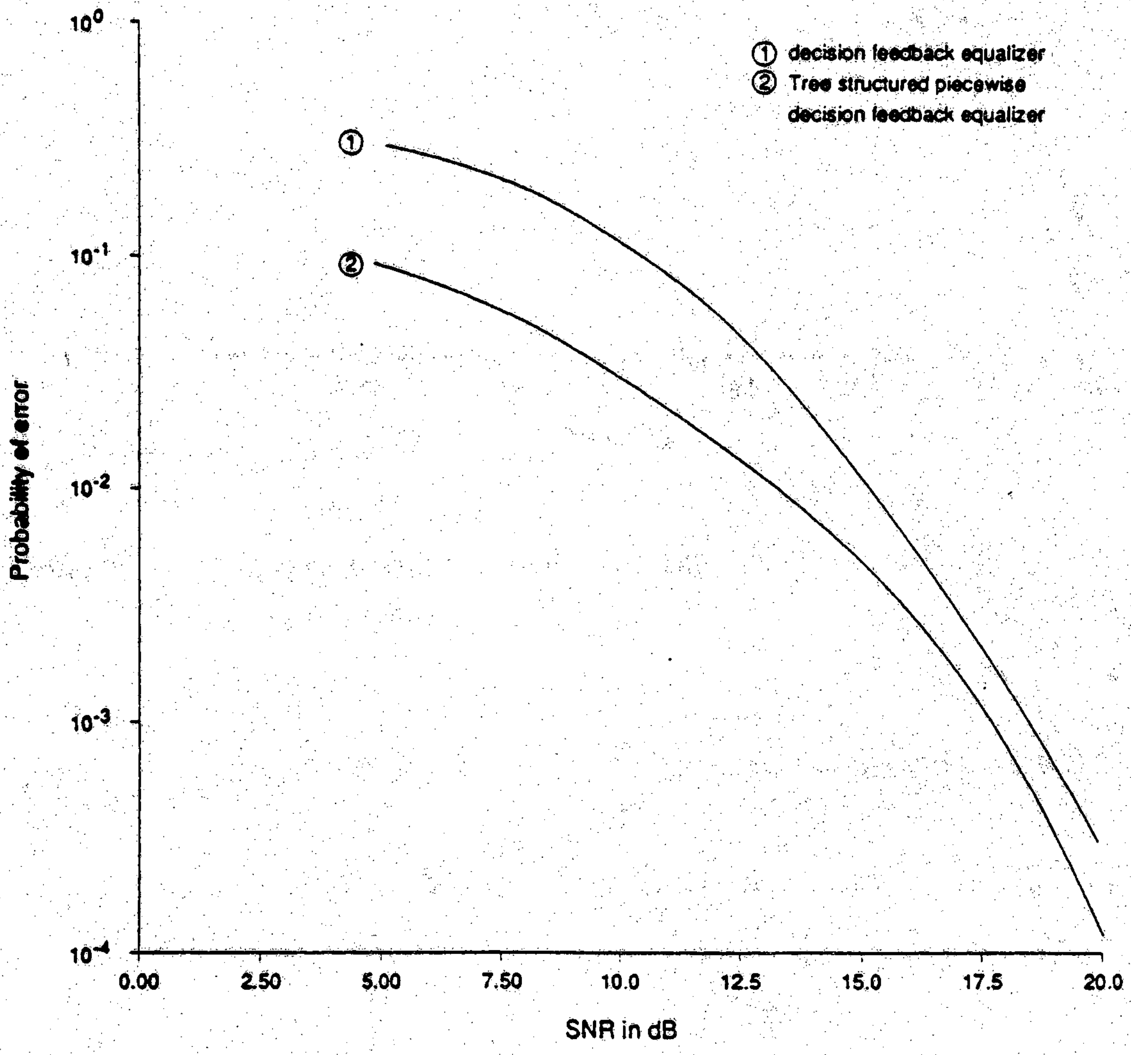


Figure 23. Asymptotic Probability of Error Curves

system as shown in Figure 24. The two nonlinear systems correspond to the compressor and expander shown in Figure 10 and the linear system corresponds to the hybrid and other associated circuitry in the telephone network [30]. A typical mu-law compressor and expander characteristic with mu = 255 was chosen for simulation and the linear system is assumed to have an impulse response[10]

$$g(k) = e^{-0.8k}, \ k \geq 0.$$

This is a reasonable model[30] which takes into account the presence of compander for nonuniform quantization of speech signals in digital telephone networks. As stated in introduction (also see [13]), the volume dependent gain characteristics of these companders is a major source of nonlinearity in the echo path.

In Figure 24, x(k) is the speech signal from the far end speaker, n(k) constitutes the speech signal of the near end speaker and/or noise generated in the hybrid, and y(k) is the output signal of the echo path in the telephone network. n(k) is generated independently of the input x(k) to the echo path. x(k) is also fed as input to the adaptive filter and the adaptive filter is trained to minimize the mean square error(MSE) between the outputs of the echo path and the adaptive filter.

A filter of length L=10 was used for all echo cancelers. For the tree-structured echo canceler, a full tree of depth $d = 5$ was chosen as the basic tree T_0. All of the simulation results to be presented were obtained by averaging over an ensemble of 200 echo cancelers with independent data.

In Figure 25 we show the output MSE as a function of time (up to 4000 iterations) for linear, second order polynomial, third order polynomial and tree-structured piecewise linear echo canceler for an SNR of 10 dB. For each type of echo canceler, the step size μ was chosen about one half the value where unstable behavior first occurred. In Figure 26 we show the asymptotic MSE as a function of SNR after

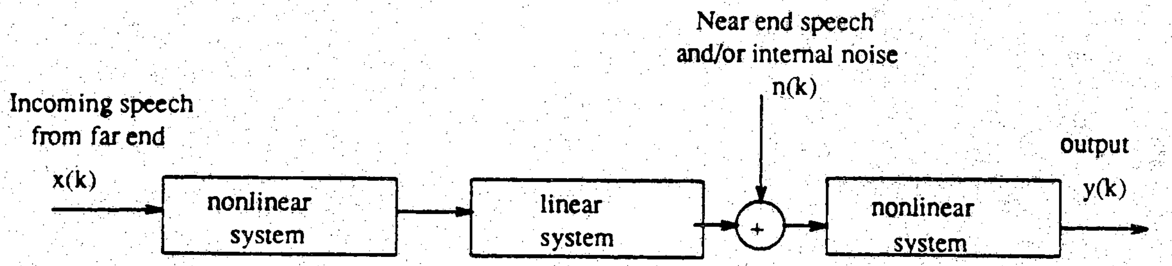


Figure 24. Simplified Model of an Echo Path

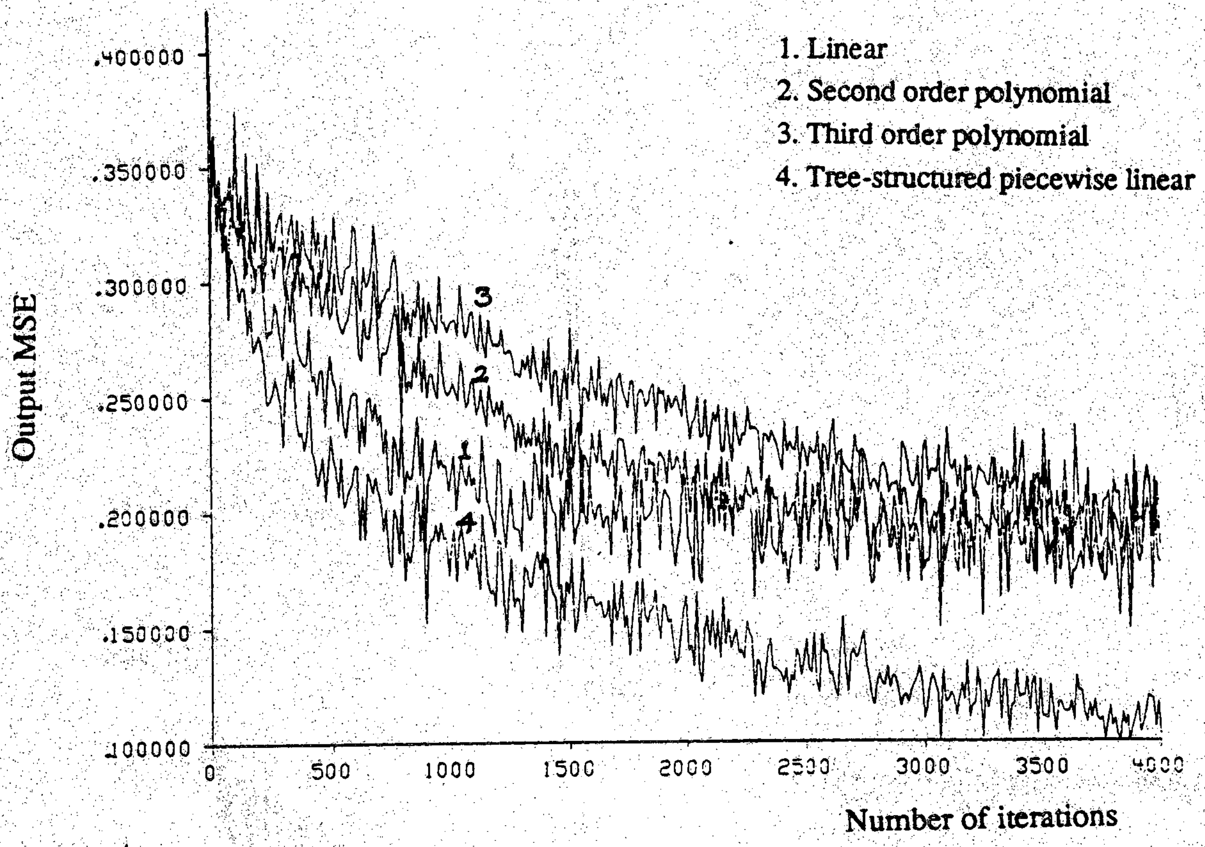


Figure 25. Learning Curves for SNR of 10dB

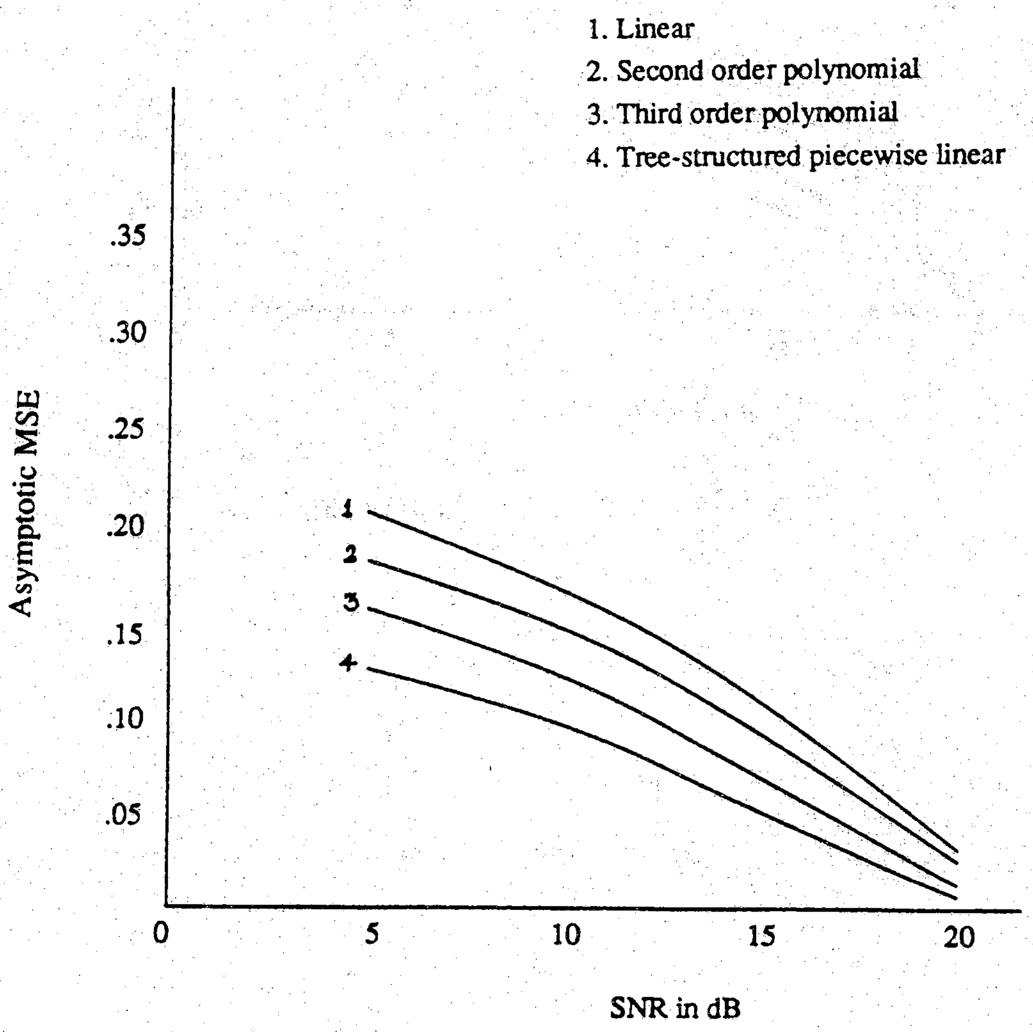


Figure 26. Asymptotic MSE Curves

about 20,000 iterations when all the echo cancelers have converged. We make several observations. First, it is clear that nonlinear adaptive cancelers are necessary to cancel echos on nonlinear echo paths. Second, it is also clear that higher order polynomial echo cancelers have lower asymptotic MSE but converge more slowly. Indeed the second and third order polynomial echo cancelers have a higher output MSE than the linear echo canceler even after 1500 iterations, although their asymptotic MSE is smaller. Next, it is seen that the tree-structured echo canceler initially converges about as fast as the linear echo canceler and from then on converges much faster to a lower asymptotic MSE. It is also seen that the tree-structured echo canceler converges much faster and to a significantly lower asymptotic MSE than the second or third order polynomial echo canceler.

6.3 Speech Coding Application

In this section we present results of computer simulations which compare the performance of a tree-structured piecewise linear adaptive predictor with that of linear, second-order and third-order polynomial types of adaptive predictors for speech-type signals.

Numerical experiments were performed on real as well as synthesized speech data[65]. Real speech data consisted of a speaker uttering the vowels a, e, i and u. Synthetic speech was generated using the simplified model of Figure 11(b). An impulsive train with a period of 10 milli-seconds was used for voiced speech and white Gaussian noise was used for unvoiced segments of the speech signal. A nonlinearity of the form $a_1 \exp(-|a_2 x|)$ was introduced in the system to represent the nonlinear glottal impedance in the speech production model. The sequences $\{a_1\}$ and

$\{a_2\}$ were dynamically generated every 50 milli-seconds using a first order auto-regressive (AR) model driven by two independent white noise sources $w_1(k)$ and $w_2(k)$, i.e.,

$$a_1(k) = 0.4a_1(k-1) + w_1(k)$$
$$a_2(k) = 0.4a_2(k-1) + w_2(k).$$

A filter of length L=12 was used for all predictors. For the tree-structured predictor a full tree of depth d=6 was chosen as the basic tree T_o.

In Figures 27 and 28 we show the learning curves (upto 4000 iterations) for linear, third order polynomial and tree-structured piecewise linear adaptive predictors for real and synthetic speech data, respectively (the learning curve for second order polynomial predictor has not been included since it overlaps considerably with the learning curve of the linear predictor). For each type of predictor, the gain μ was chosen to be about one-half the value at which unstable behavior first occured. In Figures 29 and 30 we show the SNR (defined as the ratio of power in input signal to the power in prediction error signal) in dB versus the filter length for linear, second order polynomial, third order polynomial and tree-structured piecewise linear adaptive predictors for real and synthetic speech data, respectively. We make several observations. First, it is clear that nonlinear predictors perform better than linear predictors for prediction of speech-type signals. Second, although higher order polynomial predictors yield higher SNRs, they also converge very slowly. Indeed the second and third order polynomial predictors have higher MSE than the linear predictor even after 4000 iterations although their asymptotic MSE is smaller. Third, the tree-structured predictor initially converges about as fast as the linear predictor and then converges to a much smaller asymptotic MSE (higher SNR). It is also seen that the tree-structured predictor converges much faster and to a lower asymptotic

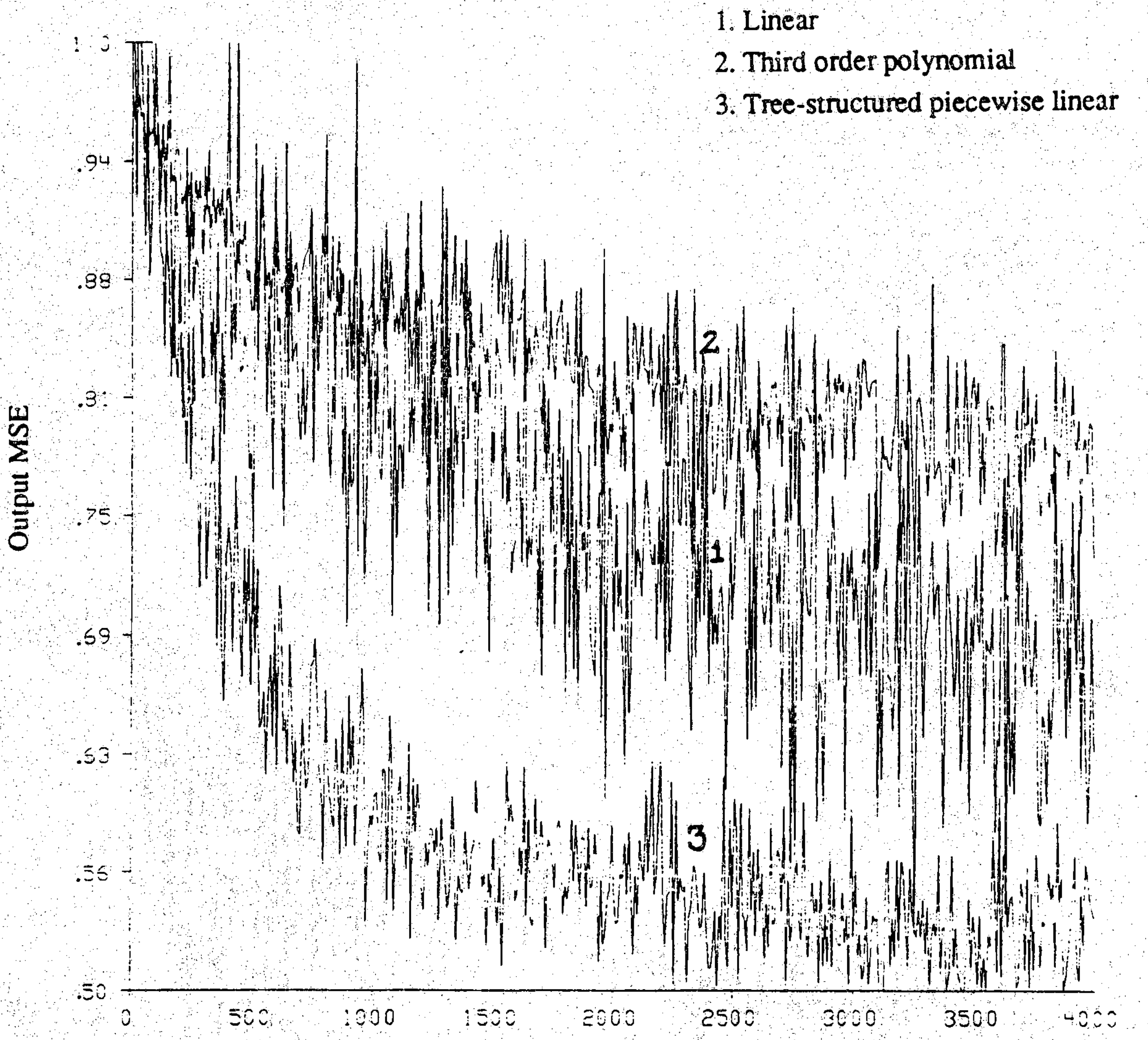


Figure 27. Learning Curves for Real Data

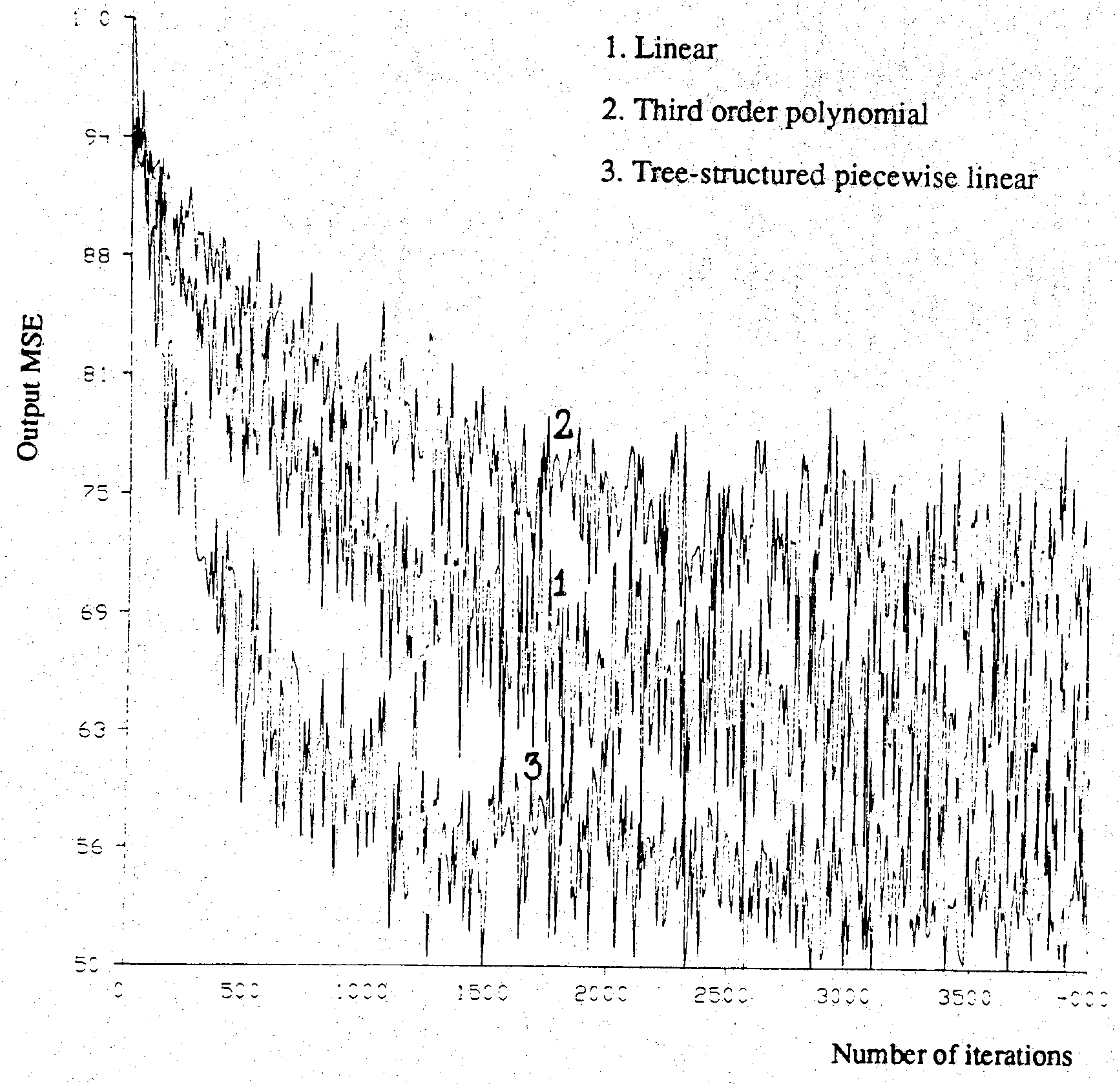


Figure 28. Learning Curves for Synthetic Data

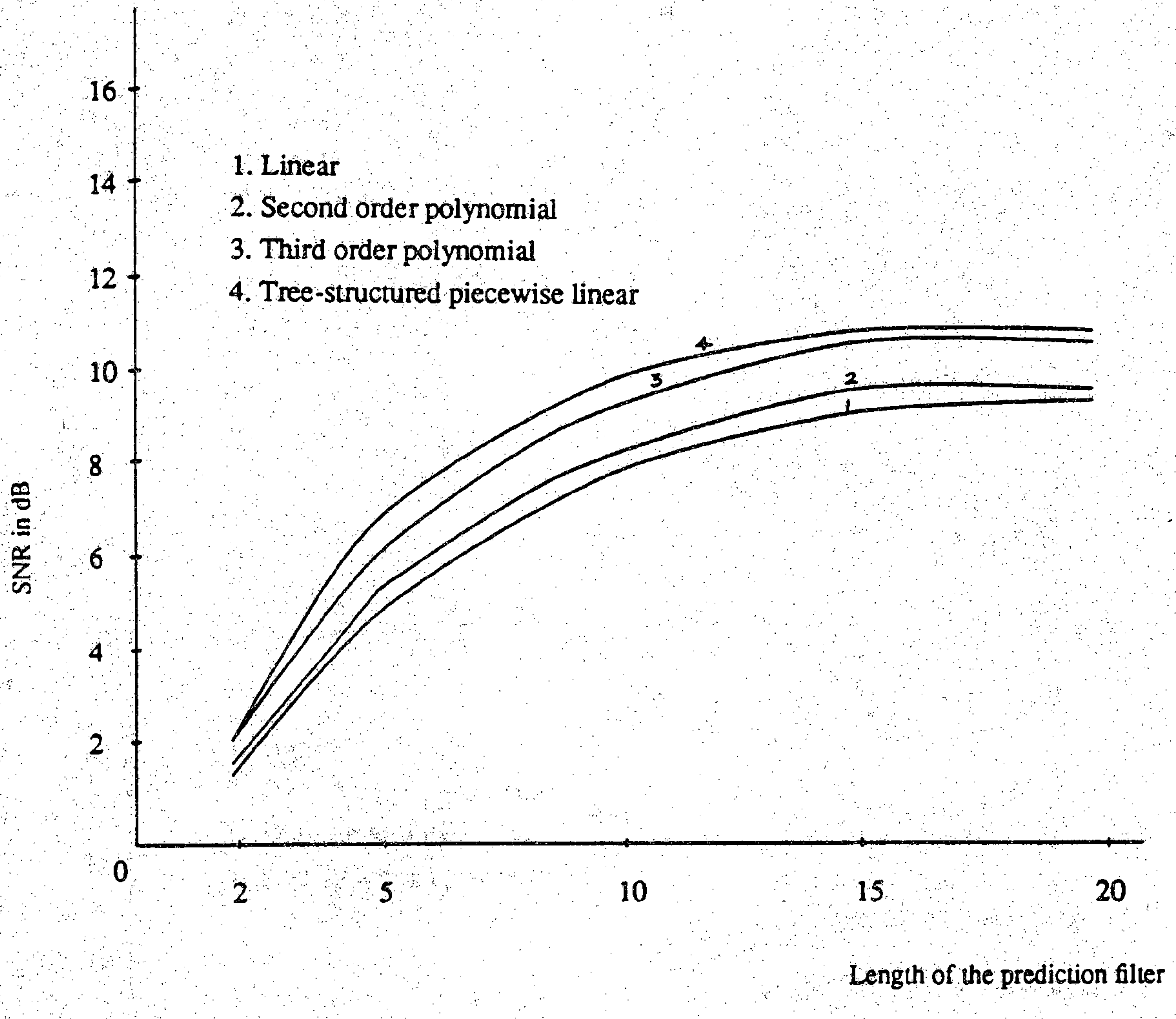


Figure 29. Performance Curves for Different Predictors (real data)

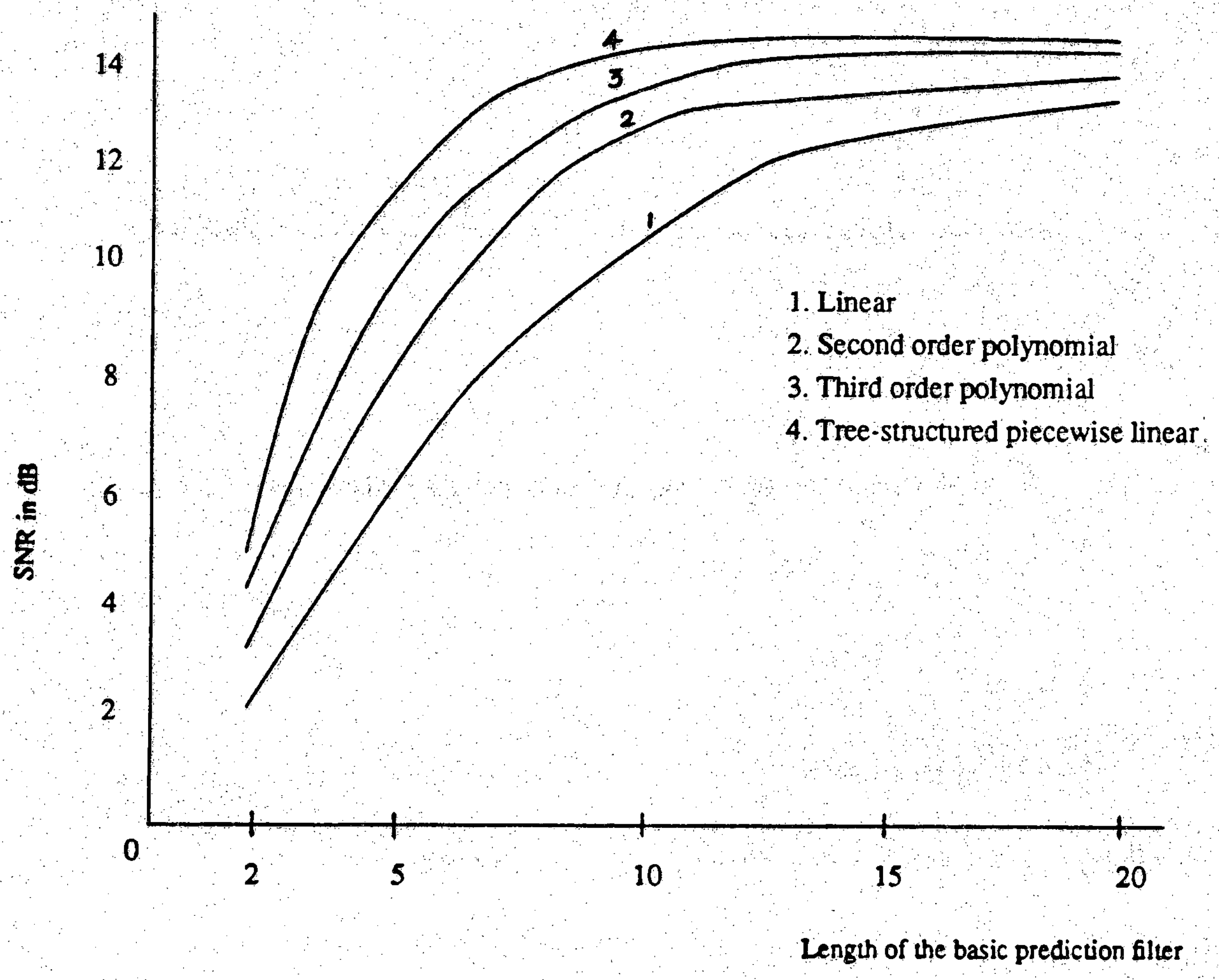


Figure 30. Performance Curves for Different Predictors (synthetic data)

MSE than the second order or third order polynomial predictors. Finally it can be seen from Figures 29 and 30 that the performance of different predictors does not improve significantly when the filter length is increased beyond a certain value. In our simulations, this value is approximately 12.

Apart from the SNR criterion discussed above to compare different types of predictors, another important criterion that is used in the choice of a predictor is the amount of compression that the predictor can achieve for a specified amount of fidelity or SNR. The compression is inversely proportional to the number of coefficients of the predictor that needs to be transmitted to the receiver. In Figure 31, we plot the the number of parameters that needs to be transmitted vs SNR in dB for the different predictors for real speech data. As seen from the figure, for SNR's above 9dB, the tree-structured piecewise linear predictor requires fewer parameters to be transmitted than the linear, second and third order polynomial predictors and hence achieves a better compression. It is also true that the tree predictor converges much faster at these SNR's. Hence the tree-structured predictors are extremely useful to encode speech signals for toll and broadcast quality transmissions in communication systems where such SNR's are required[32].

6.4 Comparison of Tree-Structured Piecewise Linear and Polynomial (truncated Volterra series) Filters

We first consider the computational complexity and rate of convergence of an unpruned tree-structured piecewise linear filter of length L and depth d versus a polynomial filter of length L and order q.

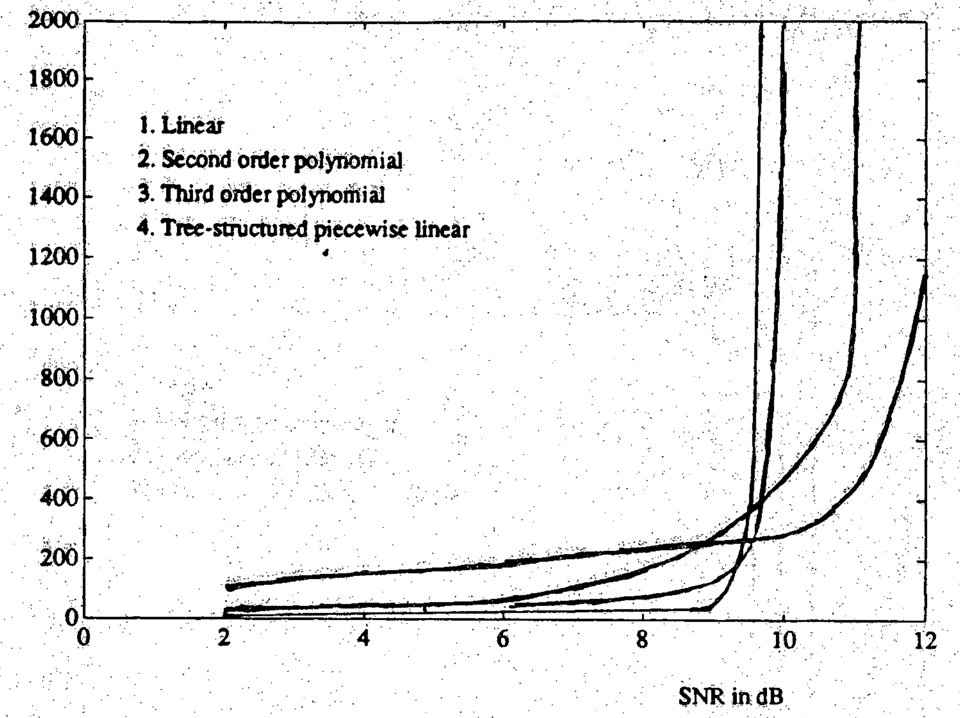


Figure 31. Number of Parameters to be Transmitted vs. SNR (real data)

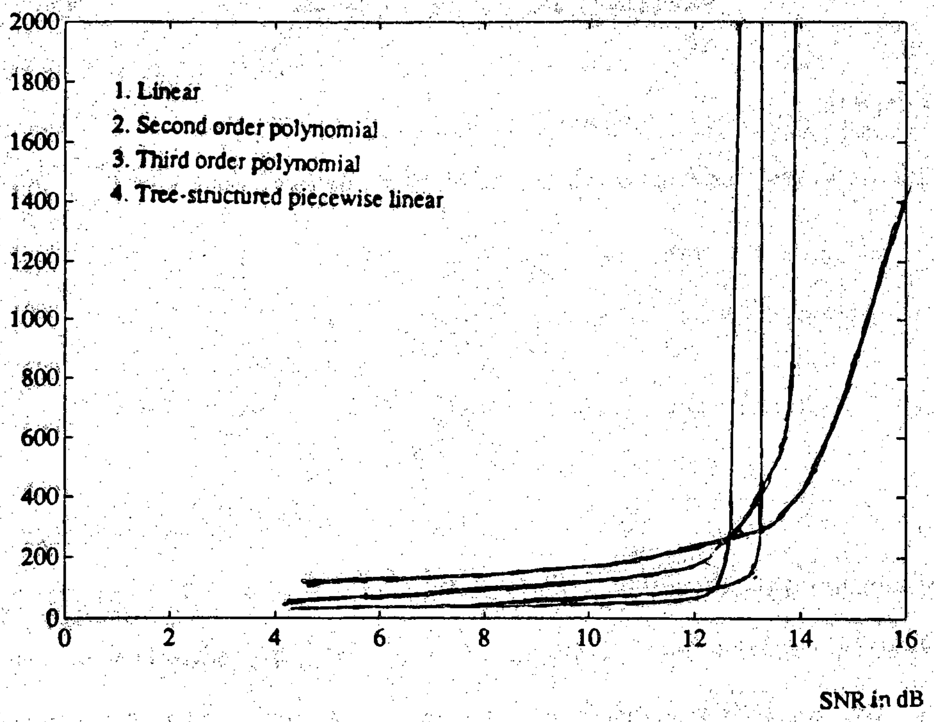


Figure 32. Number of Parameters to be Transmitted vs. SNR (synthetic data)

We can estimate the computational complexity of training the tree-structured and polynomial filters as follows. By computational complexity we mean here the number of parameter updates required for each training sample. For the polynomial filter, the complexity is just the number of tap weights $\underline{C}$ which is[†]

$$L_P = \begin{pmatrix} L+q \\ q \end{pmatrix} - 1$$

For the tree filter the complexity is the combined number of tap weights $\underline{c}_t$, offsets d_t, and thresholds θ_t along a path from the root node to a terminal node, and also all of the probabilities p_t, which totals

$$L_T = (d+1)(L+2) + 2^{d+1} - 2.$$

Denote by $L_T(L, d)$, $L_P(L, q)$ the explicit dependence of L_T and L_P on L, d, q. Then $L_T(11,5) = 140$ while $L_P(11,1) = 11$, $L_P(11,2) = 77$, $L_P(11,3) = 363$, $L_P(11,4) = 1364$ (hence the complexity of the third order polynomial filter is already much larger than the tree filter for the example considered above). It is seen that the computational complexity sets severe limits on the filter length and order of a polynomial filter compared with the filter length and depth of a tree filter; in particular for fixed q and d, the complexity of the polynomial filter $L_P \propto L^q$ whereas the complexity of the tree filter $L_T \propto L$. Also, recall that the training of the tree filter can be carried out in a highly parallel fashion (see Section 4.3), which does not seem possible for a polynomial filter.

† The number of terms of the form $c(j_1, \ldots, j_L)$ with $j_1 + \cdots + j_L = p$ is $\begin{pmatrix} L+p-1 \\ p \end{pmatrix}$ and $\sum_{p=1}^{q} \begin{pmatrix} L+p-1 \\ p \end{pmatrix} = \begin{pmatrix} L+q \\ q \end{pmatrix} - 1$ (see [52]).

We can also get some rough bounds on the asymptotic convergence rates of the tree-structured and polynomial filters as follows. Suppose we assume that the range and distribution of the eigenvalues of the (extended) input autocorrelation matrix $E\{\underline{X}\,\underline{X}'\}$ for the polynomial filters and the (conditional) input autocorrelation matrices $E_t\{\underline{x}\,\underline{x}'\}$ of the tree filter are the same (with λ_{min} the minimum eigenvalue and λ_{av} the average eigenvalue). Suppose we also assume that for the tree filter we have $\chi_t(k) = \chi_t^*$ and $p_t(k) = p_t^*$ and furthermore $p_t^* = 1/2^{depth(t)}$. Then (under suitable independence assumptions) the convergence of the MSE's will occur if and only if the gain parameter μ is upper bounded by

$$\mu_P = \frac{2}{L_P \lambda_{av}} \quad \text{for polynomial filter,}$$

$$\mu_T = \frac{2}{2^d L \lambda_{av}} \quad \text{for tree filter,}$$

and hence the corresponding largest time constant will be lower bounded by

$$\tau_P = -\frac{2}{\ln(1 - \mu_P \lambda_{min})} \quad \text{for polynomial filter,}$$

$$\tau_T = -\frac{2}{\ln(1 - \mu_T \lambda_{min})} \quad \text{for tree filter.}$$

These bounds can be derived using standard arguments (see [2]) and we do not go through the details here. It is seen that the (asymptotic) rate of convergence, like the computational complexity, sets severe limits on the filter length and order of a polynomial filter compared with the filter length and depth of a tree filter; in particular for fixed q and d and also L_o large enough such that μ_P, $\mu_T \ll 1/\lambda_{min}$, we can use a Taylor series expansion to show (approximately) that the time constant for the polynomial filter $\tau_P \propto L^q$ whereas the time constant for the tree filter $\tau_T \propto L$.

Now in the above discussion we have compared the computational complexity and rate of convergence of an unpruned tree-structured filter and a polynomial filter. However, an important advantage of the tree-structured approach is that it is possible to efficiently construct and select a pruned subtree of appropriate size. Indeed, note that the tree growing phase generates all the pruned subtrees, while the tree pruning phase selects a particular pruned subtree with very little additional computation (the complexity increases by the number of conditional MSE estimates ε_t along a particular path from root node to terminal node plus the pruning algorithm itself). The effect of selecting the pruned subtree is to avoid overfitting the data and to speed the initial rate of convergence. Now one could conceive of an approach which adaptively selects a polynomial filter of appropriate order from a sufficiently large bank of polynomial filters. However it is not clear how to efficiently construct and select from such a bank of polynomial filters since polynomial filters of different orders are not clearly related. Furthermore, the family of polynomial filters of different order will in general be so coarse that no combination of polynomial filters of different order will perform as well as the pruned tree filter, as was the case in the examples considered above.

CHAPTER 7
AUTOMATIC INSPECTION OF FUEL INJECTOR GEOMETRY

7.1 Introduction

This research is part of an ongoing atomization project investigating the characteristics of the spray process of fuel injectors. This project is sponsored by the Engineering Research Center. Previous research has shown that the geometry of the fuel injector nozzle affects the efficiency of the spray process of the injector and in turn the fuel efficiency of the engine[66]. A typical fuel injector cup is shown in Figure 33.

Geometrical features of the nozzle such as circularity of the hole, inlet and outlet diameters, droop and projection angles, entrance height, taper, and radius of curvature at the inlet of the injector hole contribute significantly to the performance of the fuel injector (see Figure 34). However the small size of the hole of the nozzle prevents mechanical means of accurately measuring the geometrical features mentioned above. The goal of this work is to acquire an image of the injector hole optically and use image processing and analysis techniques to accurately measure the geometrical features of the nozzle. The first stage of the research was focused on obtaining information about the circularity and size of the inlet (bottom) and outlet (top) of the injector hole. The holes were imaged using a microscope, vidicon camera, image digitizer and composite translation stages for holding and positioning the fuel injector. This is shown in Figure 33. The circularity information at any given depth was

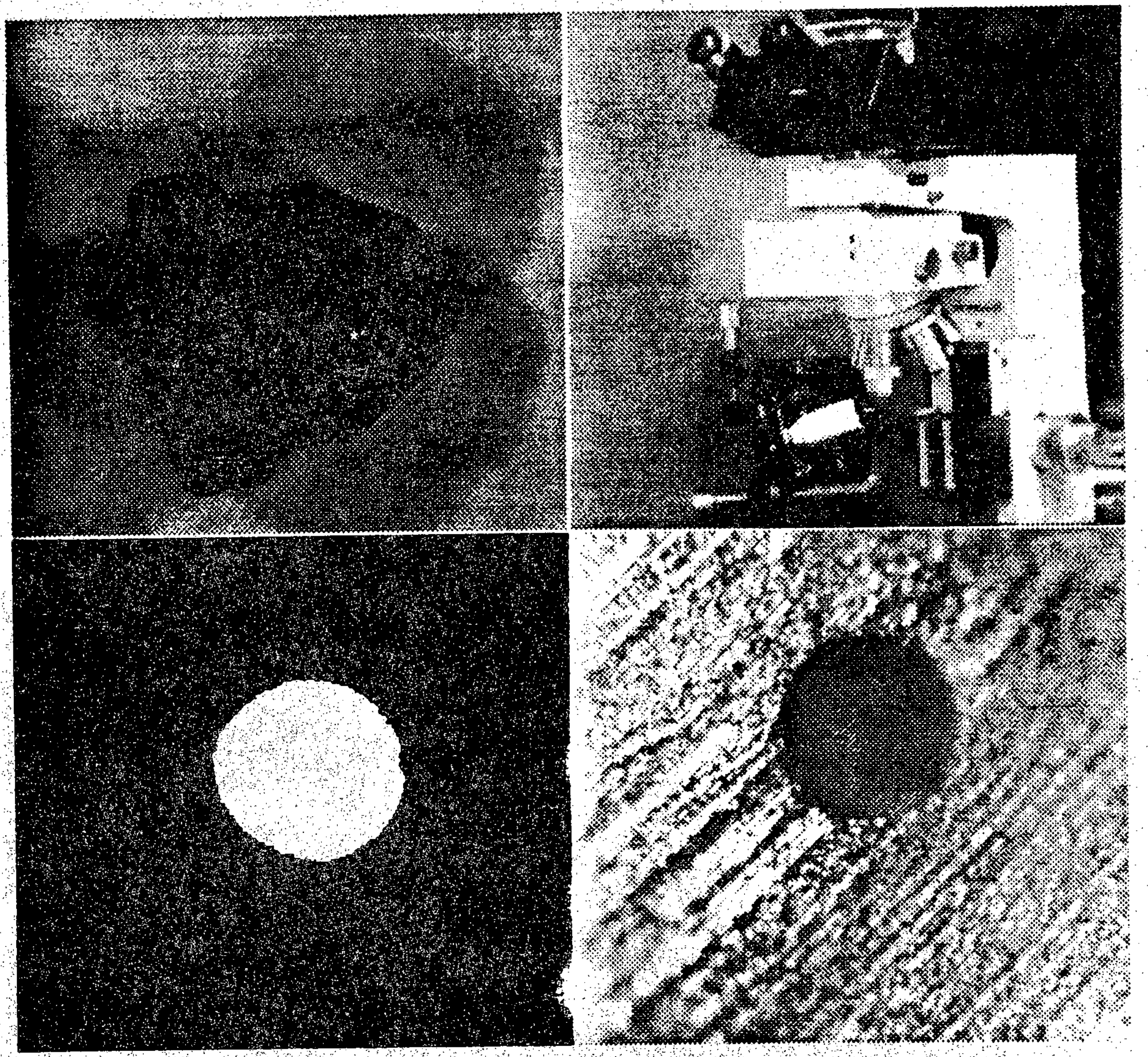


Figure 33. Fuel Injector Cup, Image Acquisition Set-up, Original Image of Bottom Lit Injector Hole, Original Image of the Top Lit Injector Hole.

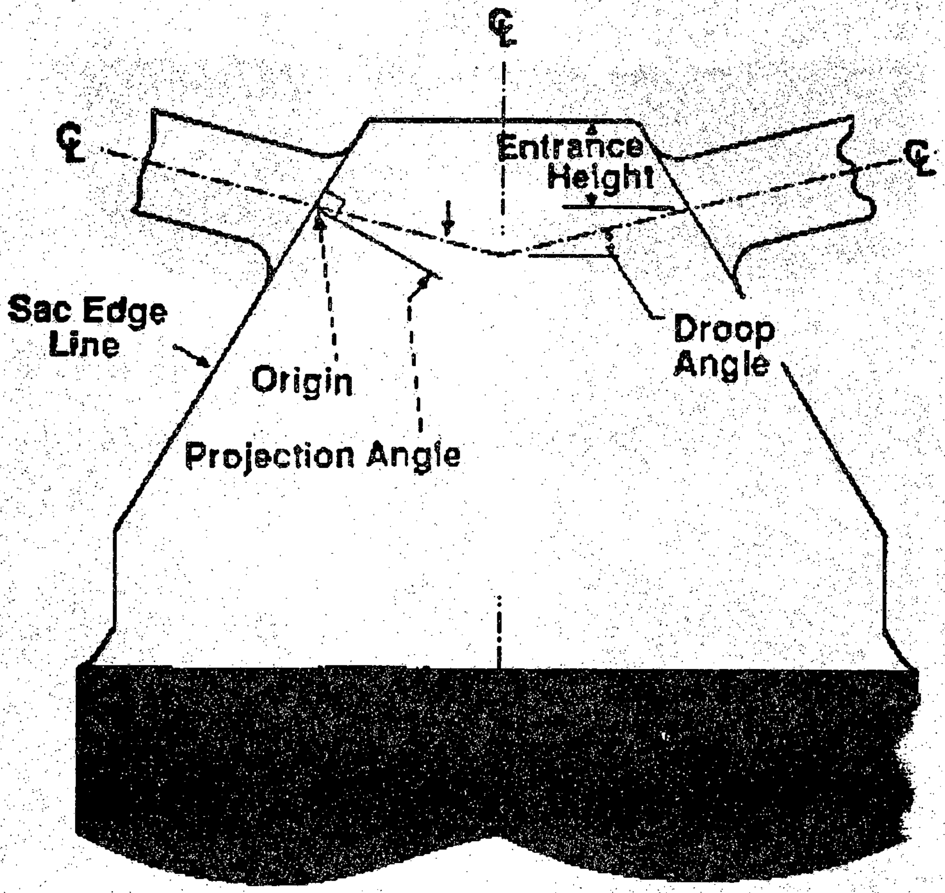


Figure 34. Schematic of the Cross Section of the Mold of the Fuel Injector Cup

obtained by constructing three dimensional wireframe and three dimensional solid versions of the hole[67, 68]. Model fitting approaches and mathematical morphology was used to obtain information about the circularity and size of the hole near the outlet. The design of measures that indicate the relative geometry at the outlet hole with respect to an "ideal" nozzle hole was investigated[67, 68].

The second stage of the research was focused on obtaining geometrical information near the inlet of the hole[69]. Since the injector inlet is at the bottom of the hole as seen from the outside and since there is a smoothly rounded bevel at the inlet of the hole, it is not possible to capture an image near the inlet by focusing the microscope at the corresponding depth. Instead of directly imaging the hole, a mold of the hole was prepared and image acquisition was performed on the mold rather than the hole itself (see Figures 35 and 36). We refer to the process of directly imaging the hole as "direct imaging" and the process of imaging the mold as "indirect imaging". Examples of direct imaging are shown in Figure 33. Detailed photographic studies and image processing techniques showed very little discrepancy between the images obtained from the actual hole and the mold representing it.

The images obtained were slightly blurred due to background noise. Several image processing techniques were used to obtain accurate information relative to the inlet diameter, the equivalent area, and the radius of curvature of the bevel at the inlet. Image segmentation was performed using a histogram oriented approach[70]. The filtering process used to clean up the segmented image utilizes a morphological filter for two reasons: removing noise and preserving the geometrical structure of the nozzle[72]. The discrepancy between the shapes of the real and ideal nozzles was then obtained. All the relevant geometrical features such as circularity of the hole, inlet and outlet diameters, droop and projection angles, entrance height and taper were

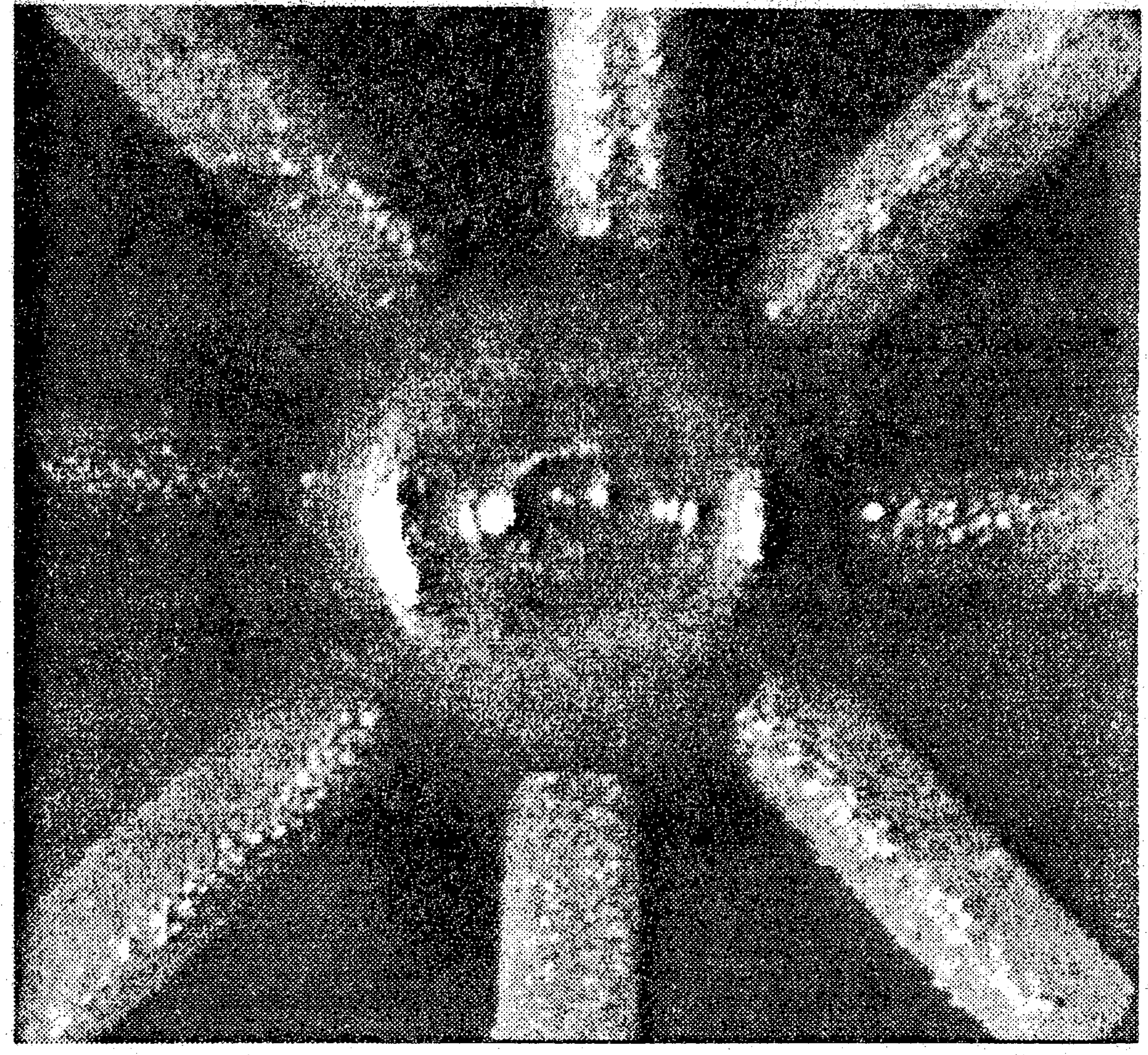


Figure 35. Mold Image of the Fuel Injector Cup (Top View)

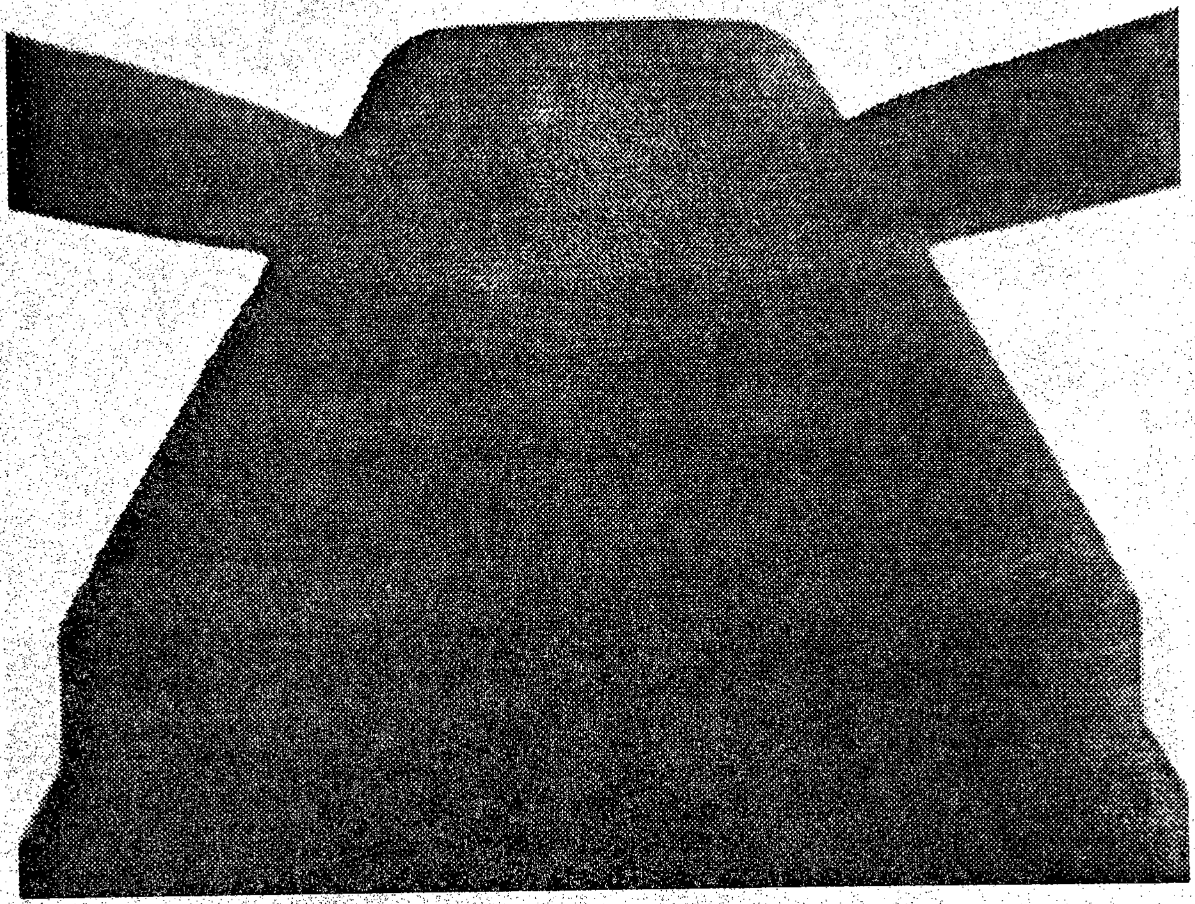


Figure 36. Original Image of the Mold of the Nozzle

automatically obtained by the inspection algorithm. After obtaining the diameters along the finger of the mold, spline fitting techniques were used to determine the effective radius of the bevel[72].

7.2 Image Acquisition

Image acquisition is performed using a microscope, vidicon camera, time base corrector, image digitizer, a composite translation stage and a light source. For image acquisition using direct imaging, the fuel injector cup is mounted on the composite translation stage which holds and positions the fuel injector (see Figure 33). The hole of the injector is illuminated from the open end of the injector cup. Thus the hole gets lit from the bottom. We refer to such a scheme of illumination as "back-lighting". Another scheme of illuminating the hole is to directly illuminate top of the hole, which we refer to as "top-lighting" (see Figure 33). Since the size of the hole is extremely small (in the range of ten-thousands of an inch in diameter), the illuminated hole is optically viewed through a microscope with a large enough magnification of the order of 200. The microscope is focused at different depths along the nozzle hole to obtain cross-sectional information. A vidicon camera, attached to the microscope, captures the image, and a time base corrector is used to synchronize the video signal output of the camera. The video signal is then quantized to 256 gray levels using a digitizer and the digitized output is stored as a two-dimensional array on the computer. This two-dimensional array is treated as the original image for further processing and analysis.

Acquisition using indirect imaging is performed by illuminating the mold from the bottom. Thus the image acquired will have a high contrast with the background pixels having gray scale values near 255 and the mold itself having gray scale values near zero.

7.3 Direct vs. Indirect Imaging

The direct imaging technique described above can be used to obtain accurate estimates of the outlet diameter and three dimensional wireframes of the hole, but the technique fails to give provide estimates of the geometrical information near the inlet portions of the hole. This is because the images obtained at depths near the inlet of the hole are blurred due to imperfect focusing and internal scattering of the light. Moreover, an image of the bottom of the hole has a false edge due to the direction of illumination of the light through the hole.

Indirect imaging using the molds has several advantages. (1) Since the outer surface of the finger of the mold represents the inner wall of the actual hole it is possible to obtain geometrical information near the inlet of the hole (see Figure 36). (2) By using image processing techniques it is possible to obtain the actual diameter of the hole "at any given depth" and hence the amount and nature of the taper (linear/nonlinear) can also be determined. It should be noted that in order to achieve the same information using direct imaging one has to use interpolation techniques which inherently introduces errors. (3) It is possible to measure the droop angles, projection angles and entrance height for multiple orifice nozzles, which is impossible using direct imaging due to the inaccessibility of the inlet portions of the hole.

7.4 Image Analysis

We discuss analysis on images obtained by direct and indirect imaging separately. The inspection algorithm for analyzing images obtained by direct imaging involves several stages including filtering, segmentation, binary image cleaning, area measurement, center location, and the discrepancy measurement [72]. The output of the algorithm contains the effective area and the equivalent diameter of the hole under study. The discrepancy between the shapes of the real and ideal nozzles is also contained. A block diagram of the algorithm is shown in Figure 37. A complete description of the analysis of injector images obtained using direct imaging is in [72].

7.4.1 Indirect Imaging

As mentioned in the previous section, indirect imaging offers several advantages over direct imaging. In terms of processing and analysis, the back-lit images obtained by indirect imaging offers an additional advantage in the sense that the complicated filtering process used for direct imaging is not necessary since the image has a reasonably high contrast with a clear background. A block diagram of the inspection algorithm for images obtained by indirect imaging is shown in Figure 38. The input is the mold of the nozzle and the output consists of geometrical features such as the droop and projection angles, entrance height, sac edge lines of the mold, center lines and diameter plots for the fingers of the mold, and the radius of curvature of the inlet portion of the hole.

The image of the mold of the nozzle is captured using the image acquisition technique discussed in Section 7.2. A typical image of the mold of the nozzle

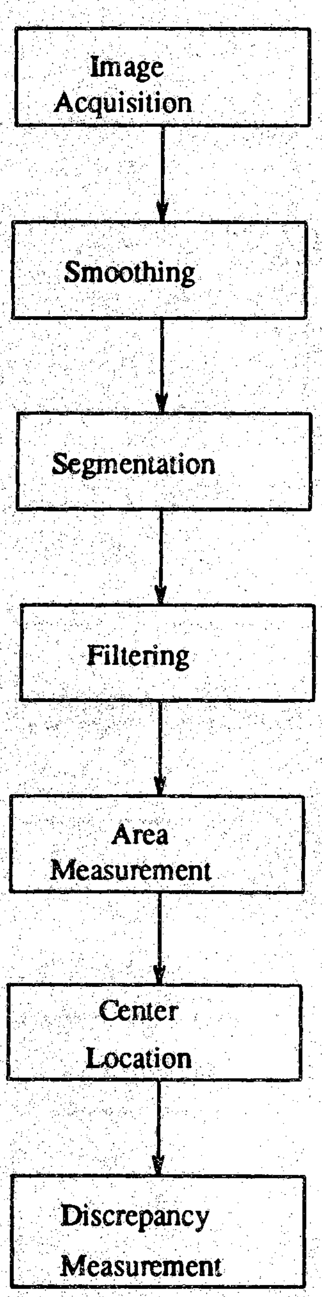


Figure 37. Block Diagram of Inspection Algorithm for Direct Imaging

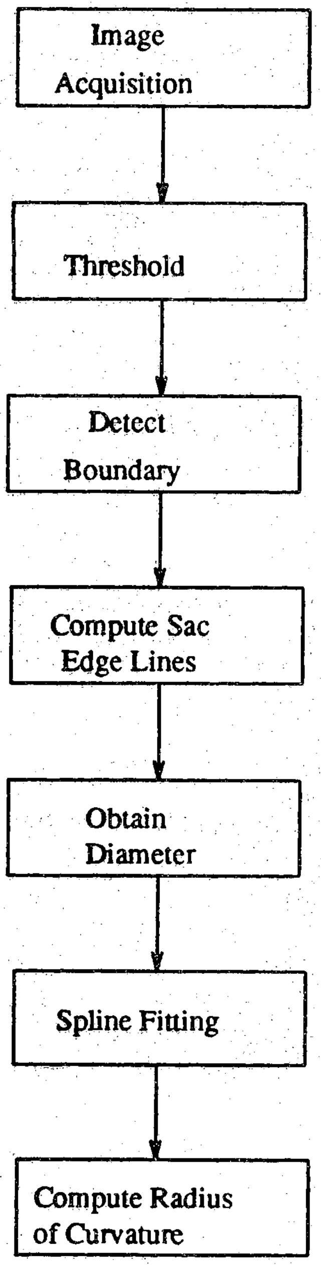


Figure 38. Block Diagram of Inspection Algorithm for Indirect Imaging

obtained by such a technique is shown in Figure 36. This image is first converted to a binary image by selecting a suitable threshold and assigning all pixels below the threshold to a gray level value of 0 (black) and all pixels above the threshold a gray level value of 255 (white) (see Figure 39). The choice of threshold is made as follows. The histogram of the original mold image is obtained which is expected to consist of two peaks widely separated from each other due to the mold and the background. Since it is clear that the background pixels have a gray scale value close to 255, and that the mold is not completely transparent to the light, the threshold value is selected just before the second peak of the histogram curve. The thresholded image of the original image is shown in Figure 39. The boundary of the thresholded image is detected using a gradient technique. The center lines of the fingers of the mold are obtained by first scanning the thresholded image vertically and obtaining the center points of the fingers and then finding the average slope of the line formed by the center points. The sac edge lines are obtained in a similar manner by searching for a slanted edge immediately below the finger and computing the average slope of the detected edge. The center lines and sac edge lines for the image in Figure 36 is shown in Figure 40. The slope of the center line represents the droop angle and projection angle is simply obtained using the equation $\gamma = 90 - (\alpha + \beta)$, where α and β are the angles of the center lines and sac edge lines with respect to the horizontal, respectively. The diameter versus the distance from the base of the finger is obtained by scanning the finger at an angle perpendicular to the center line and counting the number of dark pixels in the finger of the thresholded image. Figure 41 shows one such plot for the image of Figure 36. The same plot also indicates the amount (if any) and nature of taper in the actual hole of the fuel injector nozzle.

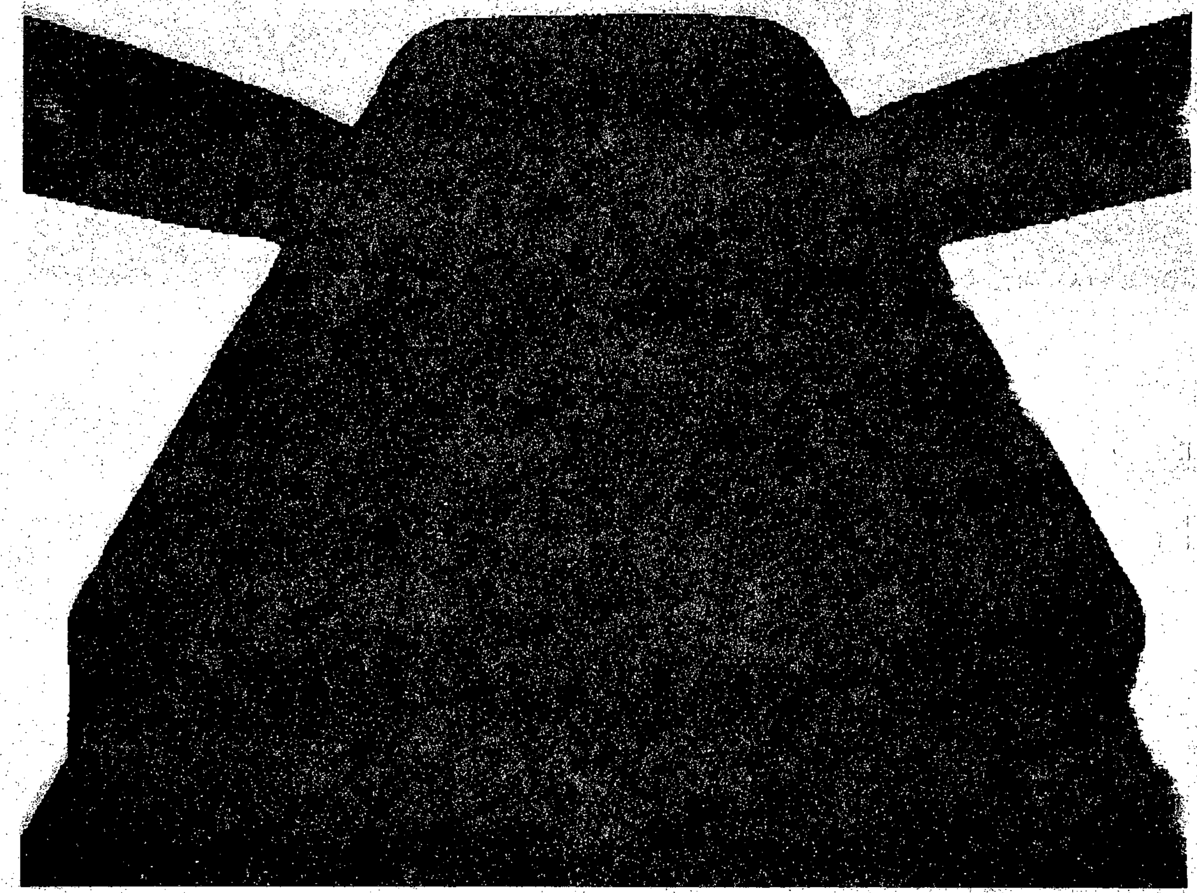


Figure 39. Thresholded Image of the Image in Figure 36

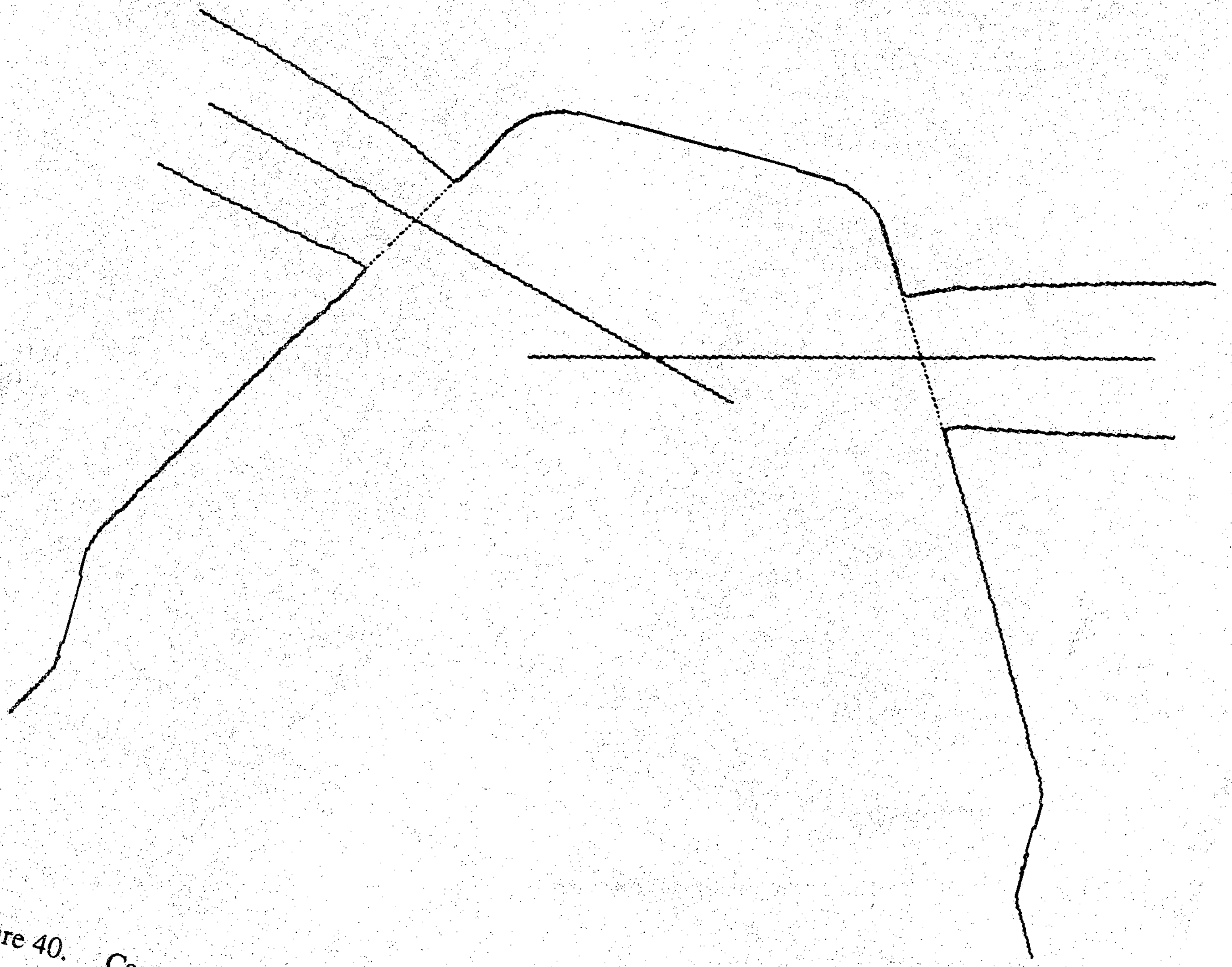


Figure 40. Center Lines and Sac Edge Lines for the Image in Figure 36

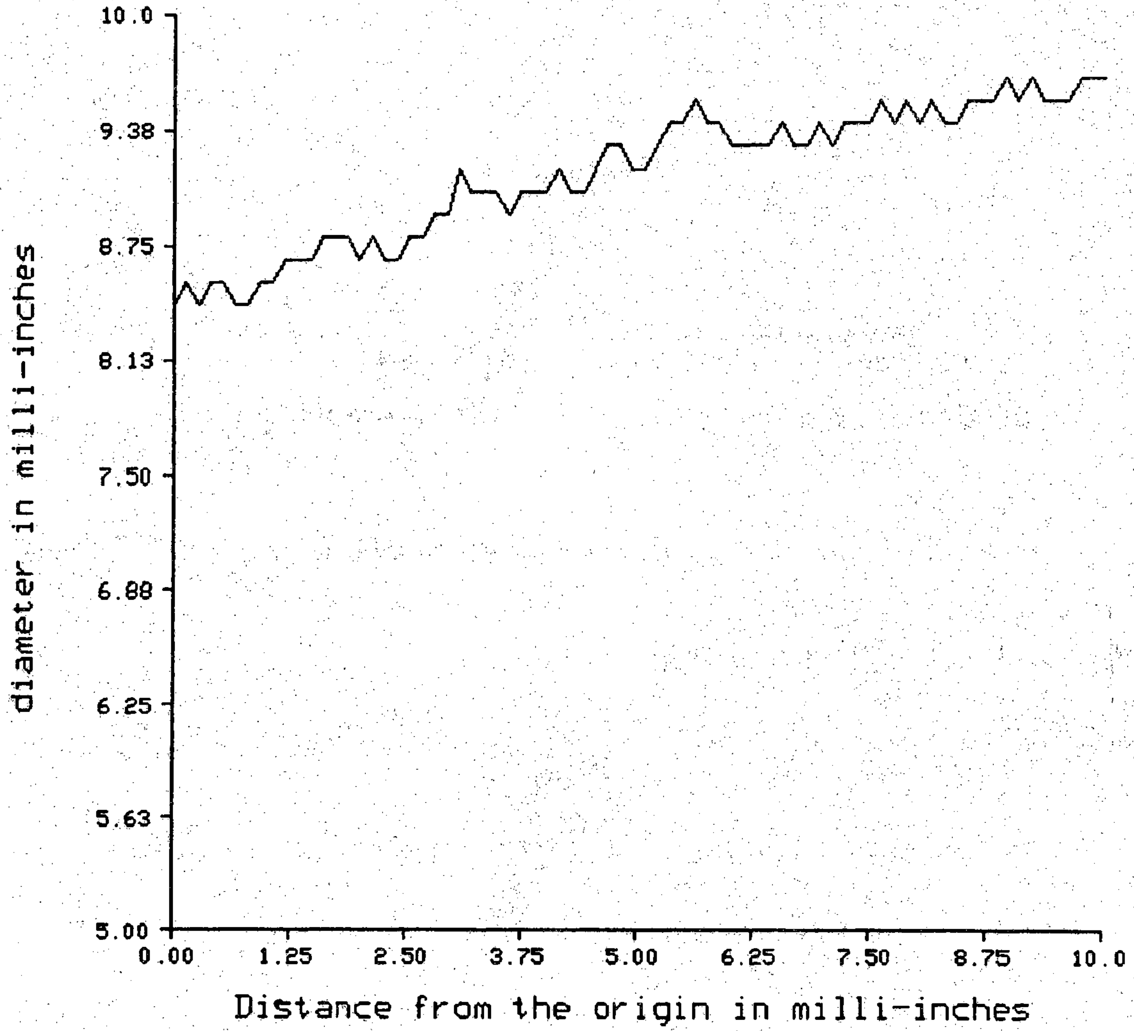


Figure 41. A Diameter Plot for the Right Orifice

Theoretically, the radius of curvature of the bevel (near the inlet) at each point is obtained by finding the radius of the osculating circle at that point[73]. If f(x) is the curve obtained by interpolating the discrete points corresponding to the edges of the images taken at different depths, then the radius of curvature at any point x_o is given by $\rho = \frac{[1 + f'(x_o)^2]^{\frac{3}{2}}}{|f''(x_o)|}$. Here we use the cubic spline interpolation to smooth the diameter plot. A typical diameter plot for a nozzle with intentionally rounded inlet is shown in Figure 42. Because of the spatial resolution of the pixels in the image, the smoothed(interpolated) curve still appears wavy as shown in the Figure 43, which makes it unsuitable for obtaining reliable estimates of the radius of curvature. Hence this data is passed through a bank of polynomial filters which then automatically selects that filter which yields the least average squared error between the data and the output of the polynomial filter. The best fit polynomial for the given set of data is shown in Figure 44. The radius of curvature is now obtained using the above equation which is shown in Figure 45 as a function of distance along the wall of the finger from the origin. The distance between two points x_o and x_1 along the curve is evaluated using $\int_{x_o}^{x_1} (1 + f'(x)^2)^{1/2}\,dx$.

The techniques used for analyzing the mold iamges have been fully automated such that once the image has been acquired, the measurements are obtained without the need for further human intervention.

We are currently extending these techniques to examine the use of direct imaging by confocal microscopy.

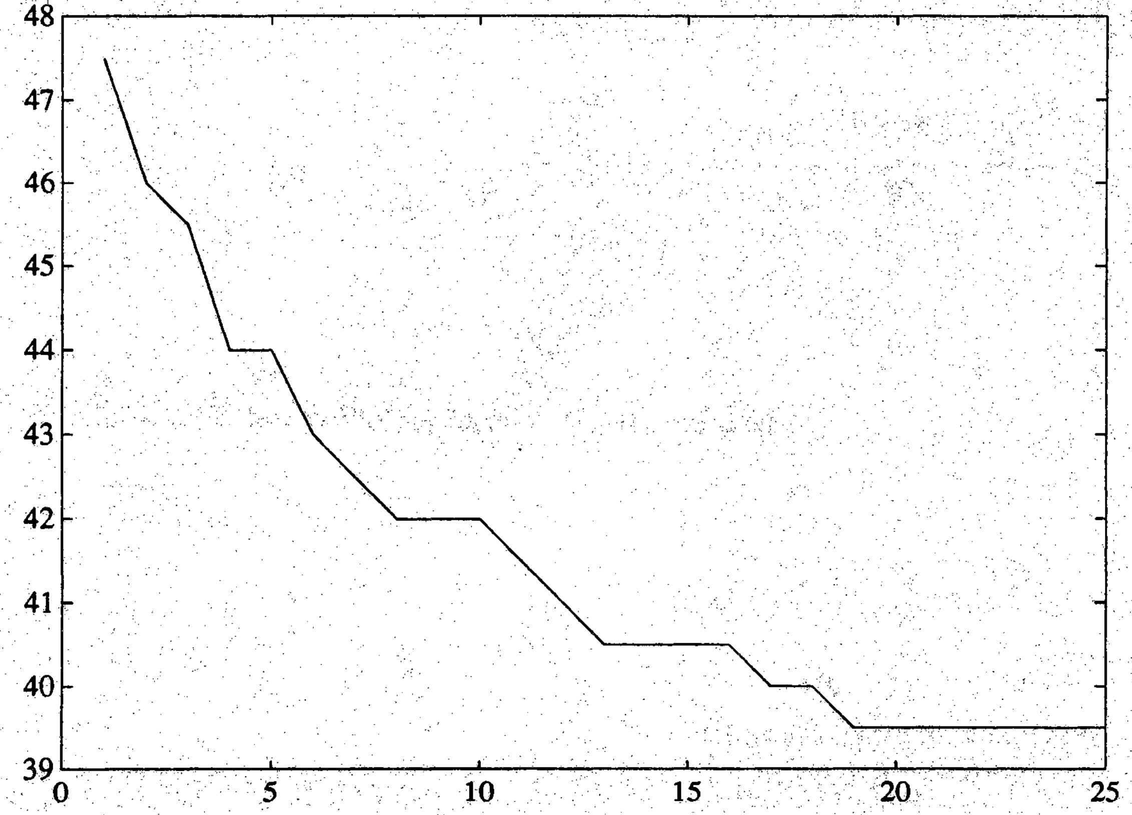


Figure 42. Diameter Plot for Intentionally Rounded Nozzle

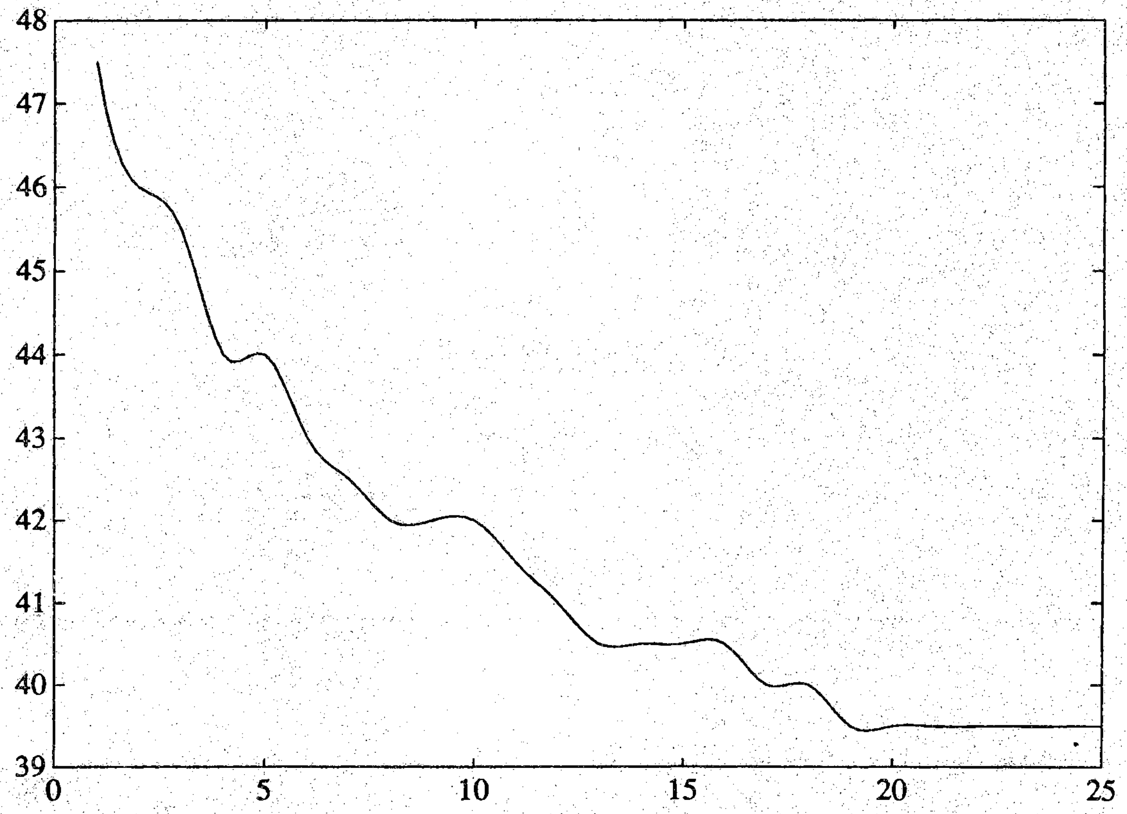


Figure 43. Interpolated Curve for the Plot in Figure 42

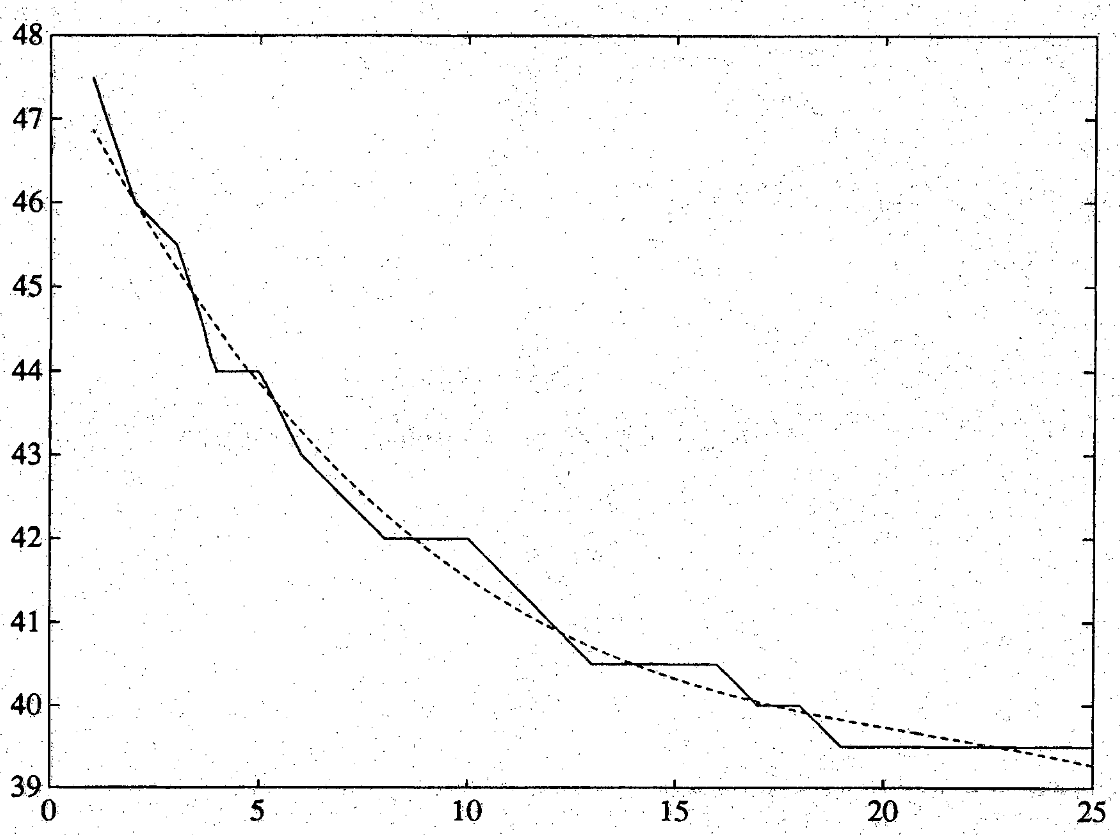


Figure 44. Best Polynomial fit for the Curve in Figure 42

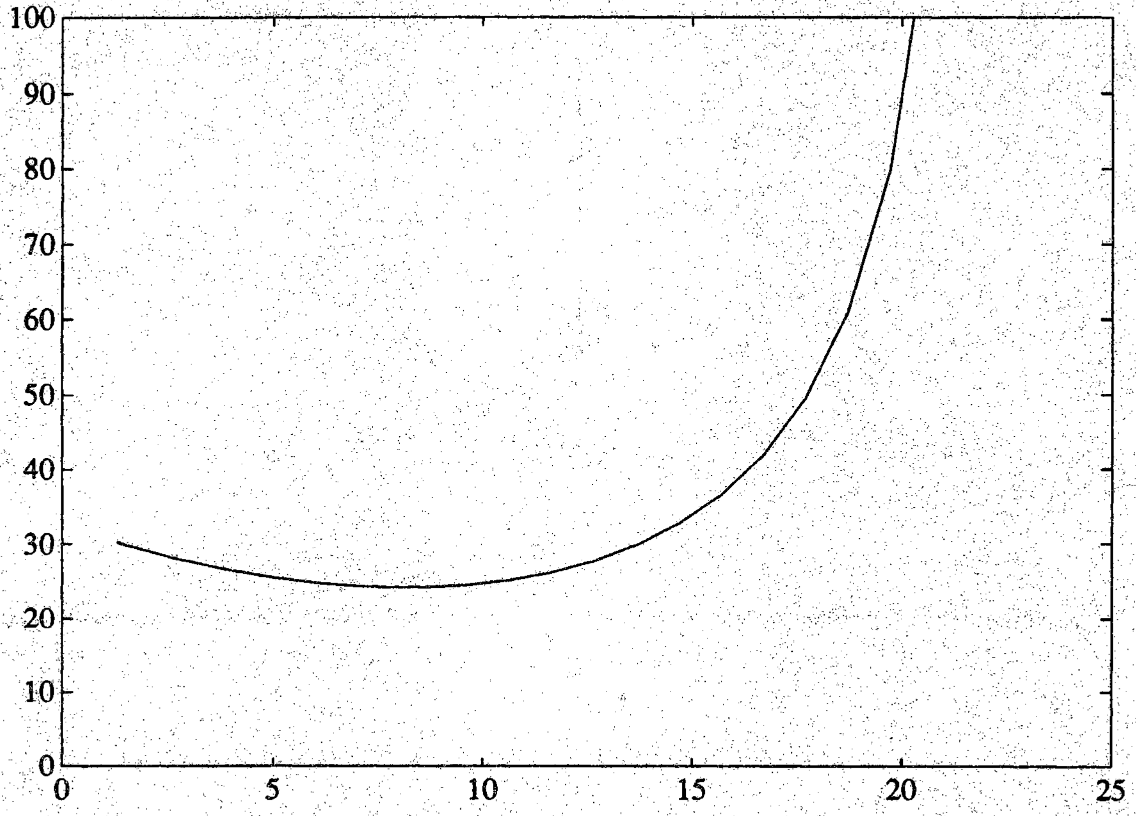


Figure 45. Radius of Curvature Plot vs. Distance from Origin

CHAPTER 8
CONCLUSIONS AND RECOMMENDATIONS

There is a general need for nonlinear adaptive filtering both due to the nature of signals that needs to be processed as well as performance limitations of linear adaptive filters in a variety of applications. We have demonstrated that the existing nonlinear adaptive filters such as polynomial (truncated Volterra series) filters suffer from several problems. First, as the order of the series is increased, the computational complexity becomes extremely large and the convergence rate becomes extremely slow. Second, there is no systematic way of neglecting higher order terms in the series.

Motivated by the success of classification and regression trees on difficult nonlinear and nonparametric problems, we proposed the idea of a tree-structured piecewise linear adaptive filter. In the proposed method each node in a tree is associated with a linear filter restricted to a polygonal domain, and this is done in such a way that each pruned subtree is associated with a piecewise linear filter. A training sequence is used to adaptively update the filter coefficients and domains at each node, and to select the best pruned subtree and the corresponding piecewise linear filter.

The tree-structured approach offers several advantages. First, it makes use of standard linear adaptive filtering techniques at each node to find the corresponding conditional linear filter. Second, it allows for efficient selection of the subtree and the corresponding piecewise linear filter of appropriate complexity. Overall, the approach is computationally efficient and conceptually simple.

The tree-structured piecewise linear adaptive filter bears some similarity to classification and regression trees. But it is actually quite different from a classification and regression tree. Here the terminal nodes are not just assigned a region and a class label or a regression value, but rather represent a linear filter with restricted domain. It is also different in that classification and regression trees are determined in a batch mode offline, whereas the tree-structured adaptive filter is determined recursively in real-time.

We first developed the specific structure of a tree-structured piecewise linear adaptive filter and derived a stochastic gradient-based training algorithm. We then carried out a rigorous convergence analysis of the proposed training algorithm for the tree-structured filter. We showed the mean-square convergence of the adaptively trained tree-structured piecewise linear filter to the optimal tree-structured piecewise linear filter. This involved an asymptotic order analysis of the fixed-gain stochastic gradient based training algorithms at the successive levels of the tree. The analysis was complicated by nonstandard dependent training data at non-root nodes corresponding to unconverged parameters at ancestor nodes. Some new techniques have been developed for analyzing stochastic gradient algorithms with fixed gains and (nonstandard) dependent data.

Numerical experiments showed the computational and performance advantages of the tree-structured piecewise linear filter over linear and polynomial filters for equalization of high frequency channels with severe intersymbol interference, echo cancellation in telephone networks and predictive coding of speech signals.

There are also several topics for further research. First, there are numerous variations of the proposed tree-structured filter and training algorithms which might be tried. Here we have used an LMS algorithm and a mean thresholding rule for computing the node weights and thresholds. Alternatively, we could use a signed LMS algorithm and a median thresholding rule for computing node weights and thresholds. We could also abandon stochastic gradient algorithms altogether and use appropriately conceived recursive least squares type algorithms.

LIST OF REFERENCES

LIST OF REFERENCES

[1] S. T. Alexander, *Adaptive Signal Processing: Theory and Applications.* Springer-Verlag, 1986.

[2] S. Haykin, *Adaptive Filter Theory.* New Jersy: Prentice Hall, 1985.

[3] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Prentice Hall, Inc., 1985.

[4] B. Friedlander, ''Lattice Filters for Adaptive Processing,'' *Proceedings of IEEE*, vol. 70, pp. 829-867, Aug. 1982.

[5] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters : Structures, Algorithms and Applications.* Kluwer Academic Publishers, 1984.

[6] B. Mulgrew and C. F. N. Cowan, *Adaptive Filters and Equalization.* Kluwer Academic Publishers, 1988.

[7] J. G. Proakis, *Digital Communications.* New York: McGraw Hill, 1983.

[8] N. S. Jayant and P. S. Noll, *Digital Coding of Waveforms : Principles and Applications to Speech and Video.* Prentice Hall, 1984.

[9] M. M. Sondhi and D. A. Berkley, ''Silencing Echoes on the Telephone Network,'' *Proceedings of the IEEE*, vol. 68, pp. 948-963, Aug. 1980.

[10] O. Agazzi, D. G. Messerrschmitt, and D. A. Hodges, ''Nonlinear Echo Cancellation of Data Signals,'' *IEEE Transactions on Communcations*, vol. COM-30, pp. 2421-2433, Nov. 1982.

[11] M. J. Coker and D. N. Simkins, ''A Nonlinear Adaptive Noise Canceller,'' *Proceedings of ICASSP*, pp. 470-473, 1980.

[12] G. L. Sicuranza, A. Bucconi, and P. Mitri, ''Adaptive Echo Cancellation with Nonlinear Digital Filters,'' *Proceedings of ICASSP*, March 1984.

[13] E. J. Thomas, "An Adaptive Echo Canceler in a Nonideal Environment (Nonlinear or Time Variant)," *Bell System Technical Journal*, vol. 50, pp. 2779-2795, 1971.

[14] J. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*. Prentice Hall, Inc., 1987.

[15] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters : Principles and Applications*. Kluwer Academic Publishers, 1990.

[16] M. Schetzen, *Volterra and Wiener Theories of Nonlinear Systems*. Wiley, Inc., 1980.

[17] T. Koh and E. J. Powers, "Second-Order Volterra Filtering and its Application to Nonlinear System Identification," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-33, pp. 1445-1455, Dec. 1985.

[18] A. J. Welch, C. E. Davila, and H. G. Rylander, "A Second-Order Adaptive Volterra Filter with Rapid Convergency," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-35, pp. 1259-1263, Sept. 1987.

[19] W. Greblicki and M. Pawlak, "Nonparametric Identification of Hammerstein Systems," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 409-418, March 1989.

[20] K. S. Narendra and P. G. Gallman, "An Iterative Method for the Identification of Nonlinear Systems using the Hammerstein Model," *IEEE Transactions on Automatic Control*, vol. AC-11, pp. 546-550, July 1966.

[21] R. W. Lucky, J. Salz, and E. J. Weldon, *Principles of Data Communication*. New York: McGraw Hill, 1968.

[22] D. D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," *IEEE Transactions on Communications*, vol. COM-26, pp. 1439-1446, Oct. 1978.

[23] M. S. Mueller, "Least Square Algorithms for Adaptive Equalizers," *Bell System Technical Journal*, vol. 6, pp. 1905-1925, Oct. 1981.

[24] E. H. Satorius and S. T. Alexander, "Channel Equalization using Adaptive Lattice Algorithms," *IEEE Transactions on Communications*, vol. COM-27, pp. 899-905, June 1979.

[25] E. H. Satorius and J. D. Pack, "Application of Least Square Algorithms to Adaptive Equalization," *IEEE Transactions on Communications*, vol. COM-29, pp. 136-142, Feb. 1981.

[26] S. U. H. Qureshi, ''Adaptive Equalization,'' *IEEE Communications Magazine*, vol. 20, pp. 9-16, March 1982.

[27] G. J. Foschini and J. Salz, "Digital Communication over Fading Radio Channels,'' *Bell System Technical Journal*, vol. 62, pp. 429-459, Feb. 1983.

[28] N. Demytko and K. S. English, ''Echo Cancellation on Time Variant Circuits,'' *Proceedings of the IEEE*, vol. 65, pp. 444-453, March 1977.

[29] K. Ochiai, T. Araseki, and T. Ogihara, ''Echo Canceler with Two Echo Path Models,'' *IEEE Transactions on Communications*, vol. COM-25, pp. 589-595, June 1977.

[30] J. Bellamy, *Digital Telephony*. John Wiley & Sons, Inc., 1982.

[31] J. L. Flanagan, *Speech Analysis, Synthesis and Perception*. New York: Springer Verlag, 1972.

[32] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. New Jersey: Prentice-Hall, Inc., 1978.

[33] B. S. Atal, ''Predictive Coding of Speech at Low Bit Rates,'' *IEEE Transactions on Communications*, vol. 1, pp. 600-614, April 1982.

[34] B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave,'' *Journal of Acoustic Society of America*, pp. 637-655, 1971.

[35] B. S. Atal and M. R. Schroeder, ''Stochastic Coding of Speech Signals at Very Low Bit Rates,'' *Proceedings of ICC*, pp. 1610-1613, May 1984.

[36] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*. New York: Springer Verlag, 1976.

[37] T. V. Sreenivas, ''Modeling LPC-Residue by Components for Good Quality Speech Coding,'' *Proceedings of ICASSP*, pp. 171-174, 1988.

[38] E. P. Neuburg and W. R. Bauer, "On the Source Filter Model of the Vocal Tract,'' *Proceedings of ICASSP*, pp. 1609-1612, 1986.

[39] D. A. George and M. J. T. Smith, "New Speech Coding Model Based on Least-Squares Sinusoidal Representation,'' *Proceedings of ICASSP*, pp. 1641-1644, 1987.

[40] N. Tishby, ''Nonlinear Dynamical Modeling of Speech Using Neural Networks,'' *First Annual Meeting of International Neural Network Society*, p. 279, 1988.

[41] B. L. Bardakjian, ''Modeling of the Laryngeal Acoustic Source by Liable Nonlinear Oscillators,'' *Proceedings of ICASSP*, pp. 36.4.1-36.4.4, 1984.

[42] J. L. Flanagan, K. Ishizaka, and K. L. Shipley, ''Synthesis of Speech From a Dynamic Model of the Vocal Cords and Vocal Tract,'' *Bell System Technical Journal*, pp. 485-506, March 1975.

[43] H. Fujisaki and M. Ljungqvist, ''Proposal and Evaluation of Models for the Glottal Source Waveform,'' *Proceedings of ICASSP*, pp. 1609-1612, 1986.

[44] D. Lin, ''Novel LPC Synthesis Model Using a Binary Pulse Source Excitation,'' *Proceedings of ICASSP*, pp. 461-464, 1986.

[45] T. Thomas and F. Fallside, ''A New Articulatory Model for Speech Production,'' *Proceedings of ICASSP*, pp. 1105-1108, 1985.

[46] C. A. Belifore and J. H. Park, ''Decision Feedback Equalization,'' *Proceedings of IEEE*, vol. 67, pp. 1143-1156, Aug. 1979.

[47] D. A. George, R. R. Bowen, and J. R. Storey, ''An Adaptive Decision Feedback Equalizer,'' *IEEE Transactions on Communication Technology*, vol. COM-19, pp. 278-293, June 1971.

[48] P. J. W. Rayner and M. R. Lynch, ''A New Connectionist Model Based on a Nonlinear Adaptive Filter,'' *Proceedings of ICASSP*, pp. 1191-1194, 1989.

[49] G. J. Gibson, S. Siu, and C. F. N. Cowan, ''Multilayer Perceptron Structures Applied to Adaptive Equalizers for Data Communications,'' *Proceedings of ICASSP*, pp. 1183-1186, 1989.

[50] J. R. Casar-Corredera, J. M. Alcazar, and Figueiras-Vidal, ''Some Simple Nonlinear Prediction Schemes,'' *Proceedings of International Conference on Digital Signal Processing*, pp. 348-352, 1984.

[51] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth Int.,, 1984.

[52] W. Feller, *Introduction to Probability Theory and its Applications*. Wiley, 1967, 1 .

[53] S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, ''An Iterative Growing and Pruning Algorithm for Classification Tree Design,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 163-174, Feb. 1991.

[54] P. A. Chou, T. Lookabaugh, and R. M. Gray, ''Optimal Pruning with Applications to Tree Structured Source Coding and Modeling,'' *IEEE Transactions on Information Theory*, vol. IT-35, pp. 299-315, March 1985.

[55] R. M. Goodman and P. Smyth, ''Decision Tree from a Communication Theory Standpoint,'' *IEEE Transactions on Information Theory*, vol. IT-34, pp. 979-993, 1988.

[56] J. K. Kim and L. D. Davisson, ''Adaptive Linear Estimation for Stationary M-Dependent Processes,'' *IEEE Transactions on Information Theory*, vol. IT-21, pp. 23-31, Jan. 1975.

[57] D. Farden, ''Stochastic Approximation with Correlated Data,'' *IEEE Transactions on Information Theory*, vol. IT-27, pp. 105-113, Jan. 1981.

[58] S. K. Jones, R. K. Cavin, and W. M. Reed, ''Analysis of Error-Gradient Adaptive Linear Estimator for a Class of Stationary Dependent Processes,'' *IEEE Transactions on Information Theory*, vol. IT-28, pp. 318-329, March 1982.

[59] O. Macchi and E. Eweda, ''Second Order Convergence Analysis of Stochastic Adaptive Linear Filtering,'' *IEEE Transactions on Automatic Control*, vol. AC-28, pp. 76-85, Jan. 1983.

[60] A. Krieger and E. Masry, ''Convergence Analysis of Adaptive Linear Estimation for Stationary Dependent Processes,'' *IEEE Transactions on Information Theory*, vol. IT-34, pp. 642-654, July 1988.

[61] O. Macchi, ''Resolution adaptative de l'equation de Wiener Hopf,'' *Ann. Inst. Henri Poincare*, vol. 14, pp. 357-379, 1978.

[62] S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, ''Tree-Structured Piecewise Linear Adaptive Equalization,'' presented at *IEEE International Conference on Communications*, Denver, June 1991, and to appear in *IEEE Transactions on Communications*.

[63] N. Holte and S. stueflotten, ''A New Digital Echo Canceler for Two-Wire Subscriber Lines,'' *IEEE Transactions on Communications*, vol. COM-29, pp. 1573-1581, Nov. 1981.

[64] S. B. Gelfand and C. S. Ravishankar, ''A Tree-Structured Piecewise Linear Adaptive Filter,'' presented at *IEEE International Conference on Acoustics, Speech and Signal Processing*, Toronto, May 1991 and submitted to *IEEE Transactions on Information Theory*.

[65] C. S. Ravishankar and S. B. Gelfand, ''Tree-Structured Piecewise Linear Adaptive Prediction,'' *submitted to IEEE Transactions on Signal Processing*.

[66] A. H. Lefebvre, D. W. Senser, H. E. Snyder, N. E. Kosinski, A. E. Mitchell, and M. R. Supple, ''Basic Studies on Paint Spray Characteristics,'' *CIDMAC Annual Report*, pp. 234-238, 1986.

[67] E. J. Delp and J. Song, ''Application of Mathematical Morphology to the Inspection of Fuel Injector Nozzle Geometry,'' *CIDMAC Annual Report*, vol. 240-244, pp. 240-244, 1988.

[68] E. J. Delp, C. S. Ravishankar, and J. Song, ''Automatic Inspection of Nozzle Geometry,'' *CIDMAC Annual Report*, pp. 182-184, 1989.

[69] E. J. Delp, D. W. Senser, C. S. Ravishankar, and J. Song, ''Flow and Spray Characterization of Diesel Injectors: Automatic Inspection,'' *CIDMAC Annual Report*, 1990.

[70] A. Farag and E. J. Delp, ''Some Experiments with Histogram-based Segmentation,'' *Proceedings of the Fourth Annual Conference on Intelligent Systems and Machines*, pp. 251-256, Michigan, April 1986.

[71] E. J. Delp, D. W. Senser, C. S. Ravishankar, and J. Song, ''Automatic Inspection of Fuel Injector Geometry,'' *CIDMAC Annual Report*, 1991.

[72] J. Song, ''A Generalized Morphological Filter,'' *Ph.D Thesis*, Purdue University, 1991.

[73] P. Lancaster and K. Salkauskas, *Curve and Surface Fitting*. Academic Press, 1986.