NONLINEAR ADAPTIVE PREDICTION OF NONSTATIONARY SIGNALS AND ITS APPLICATION TO SPEECH COMMUNICATIONS

By

LIANG LI, B. Eng., M. Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Ph.D of Engineering

McMaster University

NONLINEAR ADAPTIVE PREDICTION OF NONSTATIONARY SIGNALS AND ITS APPLICATION TO SPEECH COMMUNICATIONS

PH.D OF ENGINEERING (1994) MCMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: Nonlinear Adaptive Prediction of Nonstationary Signals and its Application to Speech Communications

AUTHOR: Liang Li

B. Eng. (Beijing Institute of Technology)

M. Eng. (Beijing Institute of Technology)

SUPERVISOR(S): Dr. Simon Haykin

NUMBER OF PAGES: xix, 160

Abstract

Prediction of a signal is synonymous with modeling of the underlying physical mechanism responsible for its generation. Many of the physical signals encountered in practice exhibit two distinct characteristics: nonlinearity and nonstationary. Consider, for example, the important case of speech signals. The production of a speech signal is known to be the result of a dynamic process that is both nonlinear and nonstationary. To deal with the nonstationary nature of signals, the customary practice is to invoke the use of adaptive filtering. Unfortunately, the nonlinear nature of the signal generation process has not received the attention which it deserves in that much of the literature on the prediction of speech signals has focused almost exclusively on the use of linear adaptive filtering schemes.

This thesis is aimed at the study of nonlinear adaptive prediction of nonstationary signals using neural networks and its application to real-time speech communication. In this thesis, three basic questions are answered: 1) What kind of neural networks are suited for real-time adaptive signal processing? 2) How can an adaptive neural network predictor be designed? 3) Can a neural network predictor be used for the application of real-time communications?

In this thesis, a new Pipelined Recurrent Neural Network (PRNN) is designed. The PRNN is composed of M separate modules. The modules are identical, each designed as a recurrent neural network with a single output neuron. Information flow into and out of the modules proceeds in a synchronized fashion.

A new scheme for the nonlinear adaptive prediction of nonstationary signals is proposed. The nonlinear neural network-based filter, which consists of a PRNN and a linear filter, learns to adapt to statistical variations of the incoming time series while, at the same time, the prediction is going on. The dynamic behavior of the pipelined recurrent neural network-based predictor is demonstrated in the case of several speech signals; for these applications it is shown that the nonlinear adaptive predictor outperforms the traditional linear adaptive scheme in a significant way. It should however, be emphasized that the nonlinear adaptive predictor has a much wider range of applications such as the adaptive prediction of sea clutter.

The PRNN-based adaptive predictor is applied to the adaptive differential pulse code modulation. In the encoder and decoder parts of an ADPCM system, the predictor is successfully incorporated with an adaptive quantizer for low bit-rate speech communication. The research work involves a novel combination of pipelined recurrent neural network and a robust linear adaptive filter, and the design of a new 4-, 8- or 16-level adaptive quantizer. The nonlinear adaptive differential pulse code modulation algorithm is tested with different speech signals. The new algorithm is compared with a linear ADPCM algorithm, recommended by CCITT, from several aspects such as time domain, frequency domain, and listening tests. Speech experiments show that nonlinear adaptive differential pulse code modulation provides a promising new approach for high-quality communication at low bits rates.

Acknowledgment

This thesis could not have been completed without the help, guidance, patience, and labor of many others. Although there is not the space here to give my thanks and appreciation to all who deserve it, I would like to explicitly thank a few.

To my supervisor, Dr. Simon Haykin, I thank you for your guidance, encouragement and kindly support throughout this research work. It was a privilege to have studied with his guidance. Especially, I would like to express my deep gratitude to you for your trusts as I was in difficult situation. I am indebted to you.

To the member of my Supervisory Committee, Dr. Chuck Carter and Dr. S. Qiao, thank you for your helpful comments and guidance on my research and thesis.

To CRL professors, in particular Dr. Jim Reilly, Dr. Tom Luo, Dr. Max Wong and Dr. Pat Yip, I thank you for your advice on my study and research.

To fellow CRL researchers Dr. Andy Ukraninec, Dr. Tarun Bhattacharya, Mr. Dean McArthur, and Mr. Bob Li, I thank you for the many hours of discussion and help that you have given me throughout the course of my research. I further thank Andy and Dean for their guide on computer softwares. To Bob Li , I especially thank you for your friendship.

To the staff at the CRL, thank you so much for always being so helpful and cheerful. You brought a friendly personal touch to graduate student life.

To the Department of Electrical and Computer Engineering at McMaster University, I thank you for the financial support that you have provided me by way of scholarship and teaching works.

To Dr. Cox Richard, *AT&T* Bell Labs, and Dr. Paul Mermelstein, Bell Northern Research, thank you for providing an *ADPCM* software, and speech signals.

To Dr. N. S. Jayant, *AT&T* Bell Labs, I am also most grateful for his complete examination of my thesis. He made numerous suggestion for improving the thesis.

To my Mom and Dad, I am grateful for your love, support and encouragement. I wish you are proud of your son.

To the love of my life, my darling Lena, I wish to express my sincere appreciation of your love and understanding during my study for the Ph.D degree. You always encouraged me, and you were always patient with me.

And finally, to the source of all my strength, perseverance and good fortune, I thank my motherland—China.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose of the Research

Prediction, filtering, and smoothing are three basic information-processing operations. Prediction is the forecasting side of information processing. The aim is to derive information about a quantity of interest at time $t + \tau$ for some $\tau > o$ by using information up to and including time t. The carrier of information is referred to as the signal, examples of which are speech, image and radar returns. The case of speech signals will serve as a main case study in this thesis.

Prediction of a signal is synonymous with modeling of the underlying physical mechanism responsible for its generation. Many of the physical signals encountered in practice do indeed exhibit two distinct characteristics: nonlinearity and nonstationary. Consider, for example, the important case of speech signals. Presently, the production of a speech signal is known to be the result of dynamic process that is both nonlinear and nonstationary.

To deal with the nonstationary nature of signals, the customary practice is to invoke the use of adaptive filtering. Unfortunately, the nonlinear nature of the signal generation process has not received the attention which it deserves in that much of the literature on

the prediction of speech signals has focused almost exclusively on the use of linear adaptive filtering schemes.

In principle, a neural network is ideally suited for the nonlinear processing of nonstationary signals because of two inherent characteristics: *Nonlinearity and Adaptation.* The key challenge is how to design a neural network for the nonlinear processing of nonstationary signals. The traditional method of supervised learning is unsuitable because of its off-line training requirement.

This thesis is aimed at the study of nonlinear adaptive prediction of nonstationary signal by using neural networks and its application to real-time speech communication.

In this thesis, three basic questions are answered:

1. What kind of neural network is best suited for continuous adaptive signal processing? The neural network needed should be able to learn in an on-line fashion (i.e., real-time), in which case the network learns to adapt to statistical variations of the incoming time series while at the same time it performs its filtering role.

2. How can an optimal neural network predictor be designed? By choosing neural network theory as the basis for nonlinear adaptive prediction, an attempt is made to find an optimal neural network system and corresponding algorithms for on-line learning and processing. The system and algorithm is based on both neural network learning theory and linear adaptive filtering theory.

3. Can the neural network predictor be used in real-time communication application? The answer to this question examines the characteristics of the neural network predictor and introduces a new speech processing system to modern communication.

1.2 Linear Adaptive Filter

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 noise data for the purpose of extracting information about a prescribed quantity of interest. In all cases, the filter is used to perform three basic information-processing tasks: prediction, filtering and smoothing. When the filter is used to perform prediction, it is called a predictor. In a linear filter the filter output is a linear function of the observations applied to the filter input. A nonlinear filter, on the other hand, is characterized by an output that is a nonlinear function of its input. Clearly, the design of a nonlinear filter is more demanding than that of a linear one.

Much has been done on the classical approach to the linear optimum filtering (prediction) problem, assuming the availability of second-order statistical parameters (i.e., mean and covariance functions) [7] of the useful signal and unwanted additive noise. In this statistical approach, an error signal is usually defined between some desired response and the actual filter output. The filter is designed to minimize a cost function defined as the mean-squared value of the error signal. For stationary inputs, the resulting optimum filter is commonly known as the *Wiener filter*. A more general solution to the linear optimum filtering problem is provided by the *Kalman filter* [6]. Compared to the Wiener filter, the Kalman filter is a more powerful device, with a wide variety of engineering application [73].

The design of Wiener filters and Kalman filters requires a prior knowledge of the second-order statistics of the data to be processed. Therefore, these filters are optimum in their individual ways only when the assumed statistical model of the data exactly matches the actual statistical characteristics of the input data. In most practical situations, however, the statistical parameters needed to design a Wiener filter or Kalman filter are not available, in which case the use of an adaptive filter is required. Such a device is one that is self-designing in the sense that the adaptive filter relies on a *recursive algorithm* for its

operation. A recursive algorithm makes it possible for the filter to operate satisfactorily in an environment where knowledge of the relevant signal characteristics is not available. The algorithm begins from some prescribed initial conditions that represent complete ignorance about the environment; thereafter it proceeds to adjust the free parameters of the filter in a step-by-step fashion, such that after each step the filter becomes more knowledgeable about its environment. Consequently, an adaptive filter is time varying with data-dependent parameters. This, in turn, means that an adaptive filter is in reality a nonlinear device in the sense that it does not obey the principle of superposition.

Notwithstanding this nonlinear property, adaptive filters are commonly classified into linear and nonlinear adaptive filters. In a linear adaptive filter the estimate of a quantity of interest is computed at the filter output as a linear combination of the available set of observations applied to the filter input. The ubiquitous *least-mean-square(LMS) algorithm* is an important example of linear adaptive filtering. Indeed, the LMS algorithm is the work horse of traditional forms of the ever-expanding field of adaptive signal processing. Another important example of the linear adaptive filter is the *recursive least-squares(RLS) algorithm*, which may be viewed as a special case of the Kalman filter. The LMS algorithm is "stochastic", providing an approximation of the Wiener filter formulated in accordance with the method of steepest descent. On the other hand, the RLS algorithm is "exact," providing a recursive solution to the linear filtering problem formulated in accordance with the method of least squares that goes back to Gauss.

The LMS algorithm offers simplicity of implementation and a robust performance in tracking statistical variations of a nonstationary environment. The RLS algorithm, on the other hand, offers a faster rate of convergence (defined below) than the LMS algorithm, at the expense of increased computational complexity. Various schemes, known collectively as "fast algorithms", have been devised to improve the computational efficiency of recursive least-squares estimation. In any event, a major limitation of all linear adaptive filtering

algorithms is the inability to exploit higher-order statistics of the input data, which, in turn, narrows the scope of their practical applications. In contrast, nonlinear adaptive filters involve the use of some forms of nonlinearity, which makes it possible to exploit the full information content of input data. However, the presence of nonlinearity makes it difficult to analyze mathematically the behavior of nonlinear adaptive filters in a way comparable to their linear counterpart.

The contribution of this thesis is the development of a nonlinear adaptive filter, based on neural network theory, for the prediction of nonstationary signals. The procedure consists of the following steps:

- Studying the dynamical neural networks.
- Constructing new neural networks.
- Developing a nonlinear predictor and the corresponding adaptive filtering algorithm.
- Testing the new nonlinear prediction with nonstationary signals.
- Introducing the new nonlinear prediction to real-time speech communications.

1.3 Scope and Outline of the Thesis

Chapter 2 looks at the model of human speech generation. The chapter is begun by reviewing the human speech production process from the point of view of biological science. The discussion confirms that speech signals do indeed exhibit two distinct characteristics: nonlinearity and nonstationary. The chapter focuses on artificially linear and nonlinear mathematical models of the speech-production. Recent research suggests that the model of speech production may be a time-varying dynamical system. One of the approximation methods for such a dynamical system is to build a dynamical nonlinear predictor, which

can model the speech generation.

Chapter 3 will provide a thorough summary of the principle and application of neural networks. The discussion is concentrated on neural networks for adaptive signal processing. A thorough investigation of the dynamical recurrent neural network and its real-time training algorithm is made. These reviews lay the foundation for the next four chapters.

Chapter 4 and Chapter 5 represent the core of the thesis. In Chapter 4, the Pipelined Recurrent Neural Network is developed. The design of the pipelined structure follows some important engineering and biological principles. The mathematical model and the real-time learning algorithm of the new neural network are deduced in this chapter. The Pipelined Recurrent Neural Network is a powerful dynamical system. It enhances the ability of the conventional recurrent neural network and overcomes its deficiency.

In Chapter 5, a new scheme for the nonlinear adaptive prediction of nonstationary signals, whose generation is governed by a nonlinear mechanism, is described. The adaptive predictor consists of two subsections, one of which performs a nonlinear adaptively mapping, and the other performs a linear, adaptively mapping. The nonlinear subsection consists of a Pipelined Recurrent Neural Network (PRNN) that operates in on-line fashion. The linear subsection is a linear adaptive filter. In this chapter, the algorithm of nonlinear adaptive prediction is thoroughly discussed; and the process of choosing adequate parameters of a nonlinear predictor is carefully analyzed.

The dynamical behavior of the predictor is demonstrated for the case of a speech signal; for this application it is shown that the nonlinear adaptive predictor outperforms the traditional linear adaptive scheme in a significant way. Notably , the application of PRNN-based prediction for the adaptive prediction of sea clutter is also examined.

Chapter 6 and 7 present the research of the low bit Adaptive Differential Pulse-Code

Modulation of speech signals. The nonlinear adaptive predictor designed in Chapter 4, is introduced to the real-time speech coding algorithm. A PRNN-based robust adaptive filter and a new 4-, 8-, and 16-level adaptive quantizer are developed. To round off the chapter, a nonlinear Adaptive Differential Pulse-Code Modulation algorithm is developed.

In Chapter 7, a series of subjective tests with several speech signals (a male speech, a female speech and a four-person conversation) are taken to evaluate this nonlinear ADPCM algorithm. The test results demonstrate that nonlinear adaptive predictive coding provides a promising approach for the low bit-rate adaptive differential pulse coding modulation of speech signals in high-quality communication. This is indeed a remarkable achievement, made possible by the use of neural networks to the real-time predictive coding.

Finally, Chapter 8 concludes by summarizing results and important conclusions arising from the research work, and highlights the significant contributions.

Chapter 2

Mathematical Model of Human Speech Generation

As indicated in the first chapter, the focus in the thesis is on the adaptive prediction of nonstationary signals. A speech signal is a typical nonstationary signal. Before proceeding it, a review of the basic model of generating a speech signal is in order. It is important to discuss the mechanics of producing speech in human beings, and show the complex physiological process of generating a speech. A more general view of mathematical models of speech production is given by considering a linear AR model and nonlinear dynamical models.

2.1 The Process of Speech Production in Human Beings

The human vocal mechanism is shown in Fig. 2.1 [1]. Air enters the lungs via the normal breathing process. As air is expelled from the lungs through the *trachea* (windpipe), the tensed vocal cords within the *larynx* are caused to vibrate (in the mode of a relation oscillator) by the air flow. The air flow is chopped into a quasi-periodic pulse which is then modulated in frequency by passing through the *pharynx* (throat cavity), the mouth cavity

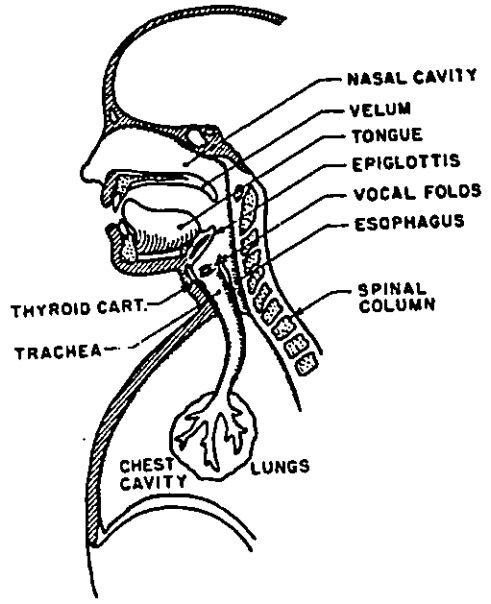


Figure 2.1: The human vocal mechanism.

, and possibly the nasal cavity. Depending on the positions of the various articulations (i.e. jaw tongue, velum, lips, mouth), different sounds are produced.

A simplified representation of the complete physiological mechanism for creating speech is shown in Fig. 2.2 [1]. The lungs and the associated muscles act as the source of air for exciting the vocal mechanism. The muscle force pushes air out of the lungs and through the bronchi and trachea. When the vocal cords are tensed, the air flow causes them to vibrate, producing so-called voiced speech sounds. When the vocal cords are relaxed, in order to produce a sound, the air flow either must pass through a construction in the vocal tract and thereby become turbulent, producing so-called unvoiced sounds, or it can build up pressure behind a point of total closure within the vocal tract, and when the closure is opened, the pressure is suddenly and abruptly released, causing a brief transient sound.

Speech is produced as a sequence of sounds. Hence the state of the vocal cords, as well as the positions, shape, and sizes of the various articulators, change over time to reflect

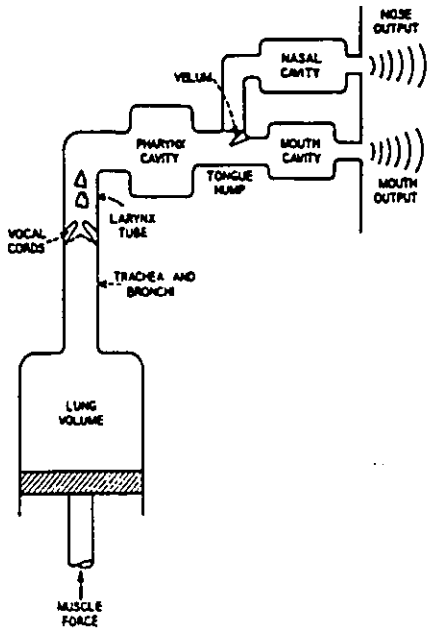


Figure 2.2: Schematic view of the human vocal mechanism.

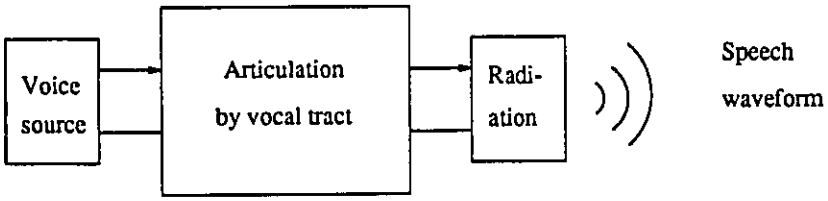


Figure 2.3: Fundamental process of speech production.

the sound being produced.

In summary, a speech waveform is produced by three actions: sound source generation, articulation by sound propagation through the vocal tract, and sound radiation from lips or nostrils, as shown in Fig. 2.3

2.2 Mathematical Model of the Speech-Production Process

Scientists and engineers want to build a model to describe the process of speech production. The mechanism of speech-production shows that the model should be a physiological one.

But, in the case of artificial speech processing, the physiological model is too complex.

Building an ideal mathematical model of the speech-production is an active research area. There should be two important objectives in mind.

Initially, the model must realistically represent mutual dependency between source generation and articulation action. A source-generating mechanism must be developed as an aerodynamic model of vocal-cord vibration and airflow turbulence. One example is a two-mass model of vocal-cord vibration to replace the present simple pulse train. Loading action of the vocal tract upon vocal-cord vibration, termination action of the vocal tract by the voice source, and format bandwidth widening of a consonant caused by an increase in loss of turbulence generation are all complex phenomena that should be treated and naturally in the nonlinear model.

It is essential to realistically account for losses due to speech wave propagation in the vocal tract. These includes heat loss, air resistance loss, leakage loss from the vocal tract wall (which is not an ideal hard wall), and turbulence loss. The loss must be described as a function of frequency and a nonlinear function of sound intensity. Bandwidth should be realized naturally and realistically by this loss.

2.2.1 Linear Model

A linear independent equivalent-circuit model of the speech-production was suggested by Atal in 1970s [61, 62]. The model is shown in Fig. 2.4(a). Two kinds of exciting sources have already been mentioned: the vocal tract to a periodic excitation for voiced speech, and a noise-like excitation for unvoiced speech.

Assume that the source spectrum is $G(\omega)$, the articulation is described by $H(\omega)$,

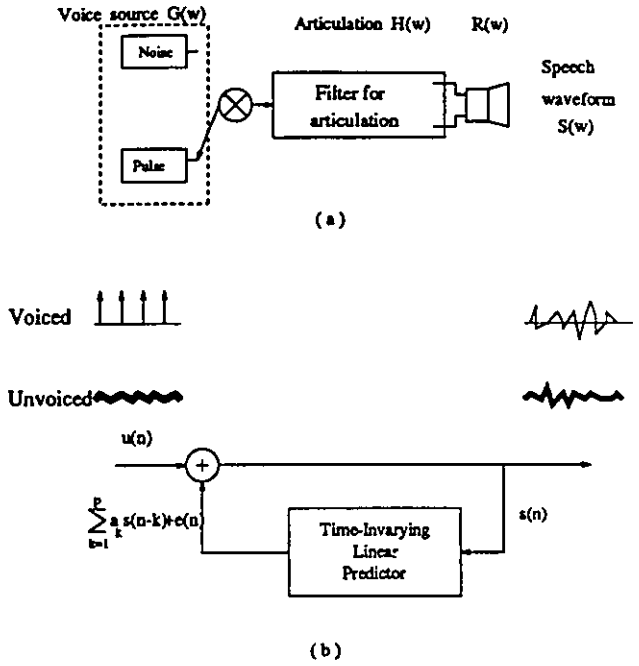


Figure 2.4: Model for linear speech production mechanisms.

and the radiation spectrum is $R(\omega)$. Then, the output speech waveform spectrum $S(\omega)$ is

$$S(\omega) = G(\omega)H(\omega)R(\omega) \tag{2.1}$$

In order to simplify the model, Atal [61] assumed the speech signal to be stationary. Under this assumption, the articulation by vocal-tract shape was simulated by all-pole or pole-zero time-unvarying linear filters, which has a transfer function $H(z)$ [1].

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2.2}$$

or

$$H(\omega) = \frac{1}{1 - \sum_{k=1}^{p} a_k e^{-j\omega k}} \tag{2.3}$$

Suppose coefficients $\{a\}$ are time invariant and $R(\omega)$ is equal to 1. According to the theory of stationary signal processes and models, the speech wave $s(n)$ is assumed to satisfy a difference equation of the form

[1] In theory, an all-pole model of the vocal tract can approximate the effect of antiresonance on the speech wave in the frequency range of interest to any desired accuracy [55]

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + u(n) + e(n) \tag{2.4}$$

where $u(n)$ is the input excitation to the system and $e(n)$ represents the modeling error in considering the speech generation process as the output of an all-pole system excited by a simplified source. The predictor coefficients $\{a_k\}$ account for the filtering action of the vocal tract, the radiation, and the glottal flow, and p is the number of poles.

Using this mathematical model, the representation for the speech signal is illustrated in Fig. 2.4(b). The results based on speech synthesis experiment indicate that, in most cases, a value of p equal to 12 is adequate at a sampling frequency of $8 \sim 10$ kHz.

2.2.2 Nonlinear Model

Although the linear model of the speech production is widely used, there is a basic question regarding the assumption of this model. Is the speech signal a nonstationary signal? If not, is the linear all-pole model an exact model?

The description of Sec. 2.1 presents that speech generation as a complex physiological process. The air flow, which is controlled by the movement of muscles, may not be described simply as a delta pulse. As well, the position of various articulators is indeed time-variant.

In recent years, many experiments have discovered the following facts:

- The voiced and invoiced signals do not correspond to the single impulse and pure white random noise as sources of excitation respectively. For different voices, the period and amplitude of the impulse signal should be different. Speech signals, in common with many of the physical signals encountered in practice, do indeed exhibit two distinct characteristics: nonlinearity and nonstationary.

- The all-pole or pole-zero model of an articulation is only suitable for short-time signals.

In fact, the speech signal is a long-time and continuous time series. There is a serious error in using a linear filtering model to describe a continuous speech signal.

The more efficient approach developed to deal with the statistical variations of speech (in both the frequency and temporal domains) consists of modeling the constituent speech units by *Hidden Markov Models* (HMMs), which was presented by Baker [3] in 1975. Baker indicated that a speech signal was a non-stationary process, but it was assumed to be "quasi-stationary". That is, it is assumed that over a short period of time the statistics of the speech do not differ from sample to sample. For example, an utterance is modeled as a succession of discrete stationary states, with instantaneous transitions between these states.

Essentially, a HMM is a stochastic finite state machine with a stochastic output process attached to each state to describe the probability of occurrence of some feature vectors. Thus there are two concurrent stochastic processes: the sequence of HMM states modeling the temporal structure of speech; and a set of state output processes modeling the (locally) stationary character of the speech signal. The HMM is called "hidden" because the sequence of states is not directly observable, but affects the observed sequence of events.

Under the "quasi-stationary" assumption, a preprocessor can extract statistically meaningful acoustic parameters (feature vectors) at regular intervals from a sampled speech waveform. Typically the original speech is sampled at 8 kHz to 16 kHz. The features are commonly extracted once per 8-16 millisecond, and are calculated from a 20 to 30 millisecond analysis window.

Recent progress in the understanding of nonlinear physical system opens an alternative route to the speech modeling. Tishby [4, 5] presented the mathematical model of

speech-production as a stochastical dynamical model. The speech signal is supposed to be the output of this complex dynamical system. A complete description of this system requires the solution of the nonlinear fluid mechanical partial differential equations with the highly varying boundary conditions of the vocal cords and the vocal tract cavities.

Nonlinear dynamical systems exhibit a dramatically richer behavior than linear systems, making it possible to model features of the signal that are totally absent in the traditional linear approach. Obtaining the complete time dependent solution of these equations is beyond the capability of present computational resources. As a result, a considerable reduction in the complexity of the dynamical system is required. The all-pole linear model and the HMM model of speech-production are two kinds of reduced models for the nonlinear dynamical system.

Moreover, the discovery of the universality of many nonlinear dynamical systems suggests a reduced dynamical description restricted to a locally low dimensional (correlation dimension) manifold, known as the attractor of the dynamics, on which a simple set of nonlinear maps can replace the high dimensional complex description of the complex dynamical system [63]. Tishby takes a series of experiments to calculate the correlation dimension of a speech signal. Carrying out the experiment for 20 segments of a voiced speech systematically reveals a correlation dimension between 3 and 5, across several speakers, independent of the uttered vowel. Similar measurements for an unvoiced speech result with higher correlation dimensions ($5 \sim 8$).

It should be emphasized that the low correlation dimension in itself is not a sufficient indication for a true deterministic underlying dynamics, or a dynamical system. Such a conclusion must be verified through the construction of an effective vector field, an accurate predictor of signals.

In general, we often face the problem of reconstructing the dynamical equation or

model, which describes the physical dynamical system, from a p· ·tion of a given empirical signal such as a speech signal. This is an "inverse problem" in dynamical systems which is equivalent to the problem of nonlinear signal prediction. Designing a good dynamical predictor for a speech signal may thus be viewed as equivalent to building a dynamical physical model of the speech production.

2.3 Summary

In this chapter, the model of human speech generation is reviewed. First, a brief discussion of the human speech production process from the point of view of biological science is presented. The discussion clarifies that speech signals do indeed exhibit two distinct characteristics: nonlinearity and nonstationary.

Then, the chapter focuses on artificially linear and nonlinear mathematical models of speech-production. Recent research suggests that the model of speech production may be a time-varying dynamical system. The all-pole model is an approximation of the complex dynamical model under the assumption that speech is a stationary signal.The Hidden Markov Model is the approximation under another assumption—that speech is a "quasi-stationary" signal.

It is not easy to describe the dynamical model by solving some differential equations with highly varying boundary conditions. A better way lies in designing a dynamical predictor which has the same ability to model the speech signal.

The analysis of Sec. 2.1 presents the human understanding of speech through the use of biological neural networks. Therefore, if an *Artificial Neural Network* is used to be a dynamical predictor, the result of modeling the speech signal should be better than linear or nonlinear engineering systems that are less analogous to biology.

Chapter 3

Neural Networks for Adaptive Signal Processing

Much of the current knowledge of adaptive signal processing is built around linear adaptive filters [7]. Linear adaptive filters are limited in their capacity to extract information from input data [73, 8]. This limitation is exemplified by the fact that they respond to second-order statistics of the input data, but no higher! It restricts the scope of their practical applications.

The significant points to note from much theoretical analysis and experiment are:

- Conventional adaptive filtering algorithms are usually phase blind. That is, they yield minimum phase solutions in the sense that they do not respond to phase information contained in the input signal in excess of a minimum phased amplitude characteristic. This points to a major limitation of conventional adaptive filtering algorithms which have their design based on second order statistics.

- To exploit the full information content of the input signal, nonlinear signal processing techniques need to be used. To be of practical value, the input signal may have a non-Gaussian distribution.

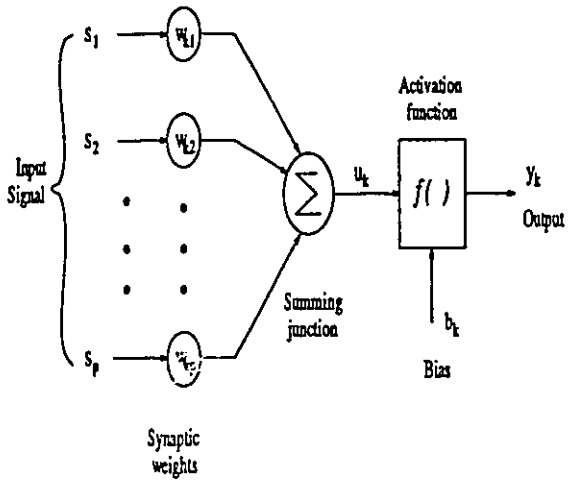


Figure 3.5: A model of an artificial neuron.

3.1 Neural Network

A neural network [9] is a massively distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. *Knowledge is acquired by the network through a learning process.*

2. *Interneuron connection strengths known as synaptic weights are used to store the knowledge.*

A neural network is also a machine that is designed to model the way in which the brain performs a particular task or function of interest [40]. It involves the use of some form of nonlinearity, which makes it possible to effectively extract more information content from environment for processing the particular task [39].

3.1.1 Models of an Artificial Neuron

The basic processor is called a neuron. Fig. 3.5 shows a simple construction of an artificial neuron (neuron k). It consists of a linear combiner followed by an activation function. The linear combiner itself consists of a set of synaptic weights $\{w\}$ connected to respective input terminals $\{s\}$, and whose weighted outputs are combined in a summing junction. An external bias b_k plus the linear combined output constitute the net input of the activation function $f(\cdot)$.

$$y_k = f(\sum_{j=1}^{p} w_{kj} s_j + b_k) \tag{3.5}$$

Four basic types of artificial neurons could be distinguished, depending on the exact description of the activation function:

- *Linear model*: in this model the nonlinear unit is replaced by a *direct connection*, with the result being that the output of the neuron is a weighted sum of its inputs Fig. 3.6(a).
- *McCullough-Pitts model:* in the second model of a neuron the nonlinear unit is characterized by a threshold function as depicted in Fig. 3.6(b).
- *Piece-wise linear model*: the input-output characteristic of the nonlinear unit for this model of a neuron is described in Fig. 3.6(c).
- *Sigmoidal model:* this last model of a neuron unit is by far the most widely used in practice. It is referred to as such because the input-output characteristic of the nonlinear unit is *S-shaped*. Let this characteristic be denoted by the nonlinear function $f(\cdot)$. A highly popular form of sigmoidal nonlinearity is defined by

$$y_k = f(\sum_{j=1}^{p} w_{kj} s_j + b_k) = \frac{1}{1 + exp(-\sum_{j=1}^{p} w_{kj} s_j + b_k)} \tag{3.6}$$

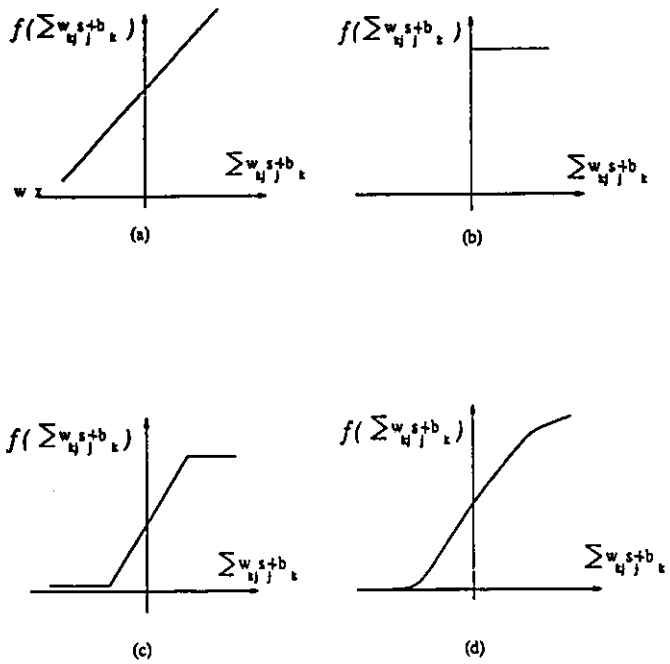


Figure 3.6: Four nonlinear functions.

Fig. 3.6(d) shows a depiction of the sigmoidal nonlinearity.

3.1.2 Classification of Neural Networks

Interest in artificial neural networks may be traced back to the pioneering work of McCullough and Pitts [10]. During the past eight years or so, interest in the theory and design of artificial neural networks and their applications has intensified. Indeed, there are now a number of journals and many international conferences devoted exclusively to neural networks.

Depending on how the desired response (target signal) for the learning process is provided, three basic classes of neural networks are identified.

- *Supervised learning*: in this form of learning, an *external teacher* is built into the design of the neural network for the purpose of providing the desired response. The

back-error propagation algorithm is an example of supervised learning algorithms.

- *Reinforcement learning*: in the second form of learning, a critic is built into the design of the neural network [18]. The critic helps the neural network to learn about its environment through a *punish and reward mechanism*. The network is rewarded for making a correct decision and punished for making a wrong decision.

- *Self-organized learning;* in this final form of learning, the neural network has no teacher. Instead, the network is designed in such a way that it can learn about its environment (i.e discover new classes automatically) without external supervision. To do this, the network usually requires a greater volume of data than a supervised one.

3.2 A New Tool for Adaptive Signal Processing

3.2.1 Nonlinearity and Learning Ability

Neural networks offer valuable characteristics that are, elsewhere, unavailable together [17]. For example:

- A neuron is basically a nonlinear device. Consequently, a neural network, made up of an interconnection of neurons, is itself nonlinear. Nonlinearity is a highly important property which makes it possible for the network to account for the nonlinear behavior of the physical phenomenon responsible for generating the input data. That is, neural networks can solve some complex problems more accurately than linear techniques do, particularly if the underlying physical mechanism responsible for the generation of an input signal is inherently nonlinear.

- A neural network can learn from the training examples representative of the working environment. The network is presented an example picked at random from the environment, and the synaptic weights of the network are modified so as to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criterion. Notably, the neural network can infer subtle, unknown relationships from data. This characteristic is useful because gathering data does not require explaining it.

- The neural networks have the ability to generalize. The generalization refers to the neural network producing reasonable outputs for inputs not encountered during learning. In this ways, neural networks can respond correctly to patterns that are only broadly similar to the original training patterns. Generalization is useful because real-world data is noisy, distorted, and often incomplete.

- A neural network can provide a universal approximation. In other words, after learning examples, the network can approximate an input-output continuous mapping to any desired accuracy.

These four information-processing capabilities make it possible for neural networks to solve complex problems that are currently intractable. For speech signal processing, there is a special reason for using neural networks:

Humans understand a speech through the use of biological neural networks; therefore artificial neural networks, which are models of the speech generation, should be able to approximate to the original processes better than engineering systems that are less analogous to biology [21].

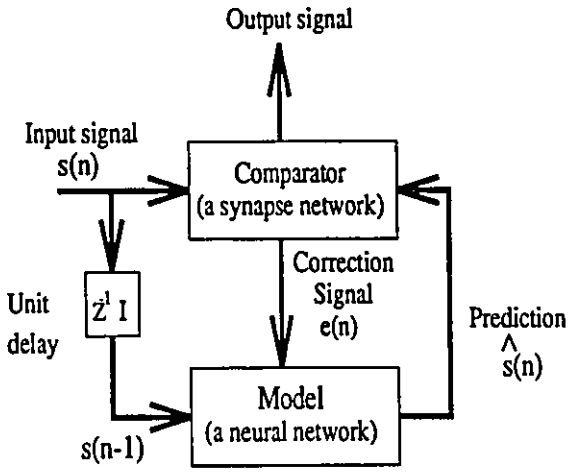


Figure 3.7: Block diagram of the adaptive system of a neuron level.

3.2.2 Self-Organized Adaptation

Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment. When a neural network operates in a stationary environment, the essential statistic of the environment can, in theory, be learned by the network under the supervision of a teacher. In particular, the synaptic weights of the network can be computed by having the network undergo a training session with a set of data that is representative of the environment. Once the training processes is completed, the synaptic weights of the network should capture the underlying statistical structure of the environment, which would justify "freezing" their value thereafter. In particular, a neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environment conditions.

Frequently, the environment of interest is nonstationary. A neural network can be designed to adapt its behavior to the temporal structure of the incoming signals in its behavior space [9]. This ability of a neural network is called *self-organized adaptation.*

The discussion of self-organized adaptation follows Mead [67]. To be specific, consider the elementary level of a multi-level processing system, the essential structure of which is described by the block diagram shown in Figure 3.7. The system is composed of many self-organized levels. In this figure, the box labeled "model" is a predictor, although perhaps a crude one, which is equipped with a model of the world. The predictor is given an input over time, and it is designed to compute what is likely to happen next, just before the actual input sample arrives. Then, when that particular input sample becomes available, it is compared to the prediction made by the network. If the two sample values are exactly the same, no new information is produced because the system already knows what is about to happen. In such a situation, the result is what is actually expected, and therefore no information is sent up to the next level of processing. But when something unexpected occurs, there is a difference or innovation, which is sent to the next level for interpretation. If this operation is repeated at each level of the system, the information will progressively be of higher quality, because one level processes only the information that could not be predicted at levels lower than the one of interest. From the point of view of a given level of the system, all lower levels may be viewed as preprocessing stages.

Learning at each level of the system is provided by the adaptation feedback from the comparator output to the model. If the model is making predictions that are systematically different from what actually occurs in the outside world, the ongoing corrections based on the individual differences will cause the model to learn what actually happens. It is only those events that are truly random, or that cannot be predicted at a particular level and therefore appear random, that will average out in an ensemble sense. The system parameters thus will undergo a local random walk; however they will be centered on the average of what the outside world provides as input.

The most important property of this kind of system is that the same mechanism that adapts out errors and mismatches in its individual components also enables the system

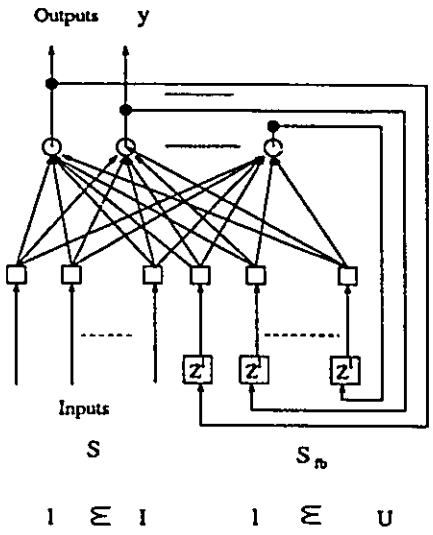


Figure 3.8: A recurrent neural network.

to build its own model of the outside world through continued exposure to information fed to it by the world. Therefore, the many-level system described herein is self-organized in the most profound sense [67].

3.3 Dynamic Recurrent Neural Networks

3.3.1 Nonlinear Dynamics of a Recurrent Neural Network

A recurrent neural network distinguishes itself from a feedforward neural network [9] in that it has at least one feedback loop. Fig. 3.8 presents a recurrent neural network.

We may distinguish two distinct layers in the network: a *concatenated input-output layer* and a processing layer. In the processing layer, there are output and hidden neurons. The output neurons send out the information y of this neural network. The outputs of all neurons are fed back to the *concatenated input-output layer.* All of the inputs consist of the external inputs s and feedback signals s_{fb}. The feedback loops involve the use of particular branches composed of unit-delay elements (denoted by z^{-1}). The I presents the group of

external input lines; and the U is the group of feedback lines.

The presence of feedback loops has a profound impact on the learning capability of the network, as well as on its performance. As a result of the delay and feedback, "time" is added to the network state. Inputs, outputs and the state of recurrent neural networks are functions of time. Therefore, the network has a nonlinear dynamical behavior by virtue of the nonlinear nature of the neurons.

In Fig. 3.8, suppose the network has N units, with P external input lines. Let $\mathbf{y}(t+\Delta t)$ denote the $N-$tuple of outputs of the units in the network at discrete time $t+\Delta t$ and let $\mathbf{s}(t)$ denote the $P-$tuple of external input signals applied to the network at time t. The one-step delay output $\mathbf{y}(t)$ is presented as the $\mathbf{s}_{fb}(t)$. The $\mathbf{s}$ and $\mathbf{s}_{fb}$ are concatenated to form the $(P+N)-$tuple $\mathbf{z}(t)$:

$$z_l(t) = \begin{cases} s_l(t) & \text{if } l \in \mathrm{I} \\ s_{fb_l}(t) & \text{if } l \in \mathrm{U} \end{cases} \tag{3.7}$$

Let $\mathbf{W}$ denote the weight matrix of the network, with a unique weight between every pair of units and also from each input line to each unit. By adopting the indexing convention just described, all the weights are incorporated in their single $N \times (P+N)$ matrix. To arrange each unit to have a bias weight, simply include one input which always has the value of $+1$.

To suppose that the network consists entirely of semilinear units, let

$$v_k(t) = \sum_{l \in U \cup I} W_{kl}(t) z_l(t) \tag{3.8}$$

denote the net input the k^{th} unit at time t, with its output being

$$y_k(t+\Delta t) = \phi_k(v_k(t)) \tag{3.9}$$

where ϕ_k is an activation function. It is important to note, however, that as indicated in Eq. 3.9, the external input vector $\mathbf{s}(t)$ at time point t does not influence the output of any neuron in the network until time $t+\Delta t$. The Δt is the processing time of a network.

The recurrent connections, which result in important capabilities not found in feed-forward networks, include both attractor dynamics and the ability to store information for later use. Of particular interest is their ability to deal with time-varying input or output through their own natural temporal operation.

Eqs. 3.8 and 3.9 describe the entire dynamics of the recurrent neural network. The general property making such networks interesting and potentially useful is that they manifest highly nonlinear dynamical behavior. One such type of dynamical behavior that has received much attention is probably that of greater importance both biologically and from an engineering viewpoint are time-varying behaviors.

The recurrent neural network can provide another important characteristic. Funahashi [26] proves that any finite time trajectory of a given n-dimensional dynamical system can be approximately realized by the internal state of the output units of a continuous time recurrent neural network with n output units, some hidden units, and an appropriate initial condition. Funahashi's theorem of approximation is:

> Let $\phi(\cdot)$ be a strictly increasing C^1-sigmoidal function such that $\phi(\mathbf{R}) = (0,1)$ (where $\mathbf{R}$ is an Euclidean space), and g: $I = [0,T] \rightarrow (0,1)^{n_o}$ be a *continuous curve*, where $0 < T < \infty$. Then, for an arbitrary $\epsilon > 0$, there exist a recurrent neural network with n_o output units and n_h hidden units such that
>
> $$\max_{t \in I} |\mathbf{g}(t) - \mathbf{y}(t)| < \epsilon \tag{3.10}$$
>
> where $\mathbf{y}(t) = [y_1(t), \ldots, y_{n_o}(t)]^T$ is the output of the recurrent network with the sigmoid output function ϕ of every neuron.

The recurrent network, which has the nonlinear dynamical behavior, is suitable for spatio-temporal information processing. Specifically, the character of the approximation of dynamical system gives the possibility of on-line approximating a time trajectory using

a recurrent network. It means that a recurrent neural network may be used to model a nonlinear dynamical system. Jerome has shown that the recurrent neural network is a special case of *nonlinear autoregressive moving average model* (NARAM) [34]. This being the case, recurrent networks are well-suited for time series analysis.

3.3.2 Real-Time Temporal Supervised Learning Algorithm

In 1989, Williams and Zipser [47] proposed a new learning algorithm of recurrent neural networks in the case of discrete time systems, which enjoys the generality of the backpropagation-though-time approach while not suffering from its growing memory requirement in arbitrarily long training sequences.

Williams's algorithm is designed for training the network described in Fig. 3.8, in a "temporal supervised learning" task, meaning that certain of the units' output values are to match specified target values at specified times.

Suppose that $n = t$ and $\Delta t = 1$, and rewrite Eq. 3.7, 3.8 and 3.9 as

$$z_l(n) = \begin{cases} s_l(n) & \text{if } l \in \mathrm{I} \\ s_{fb_l}(n) & \text{if } l \in \mathrm{U} \end{cases} \tag{3.11}$$

and

$$v_k(n) = \sum_{l \in U \cup I} W_{kl}(n) z_l(n) \tag{3.12}$$

denote the net input the k^{th} unit at the n^{th} time point, with its output at the next time step being

$$y_k(n+1) = \phi_k(v_k(n)) \tag{3.13}$$

where ϕ_k is an activation function.

Let $d_k(n)$ denote the desired response of neuron k at time n. Let χ denote the set of neurons that are chosen to act as "visible" units in that they provide externally

reachable output; the remaining neurons of the processing layer are hidden. Then define a time-varying $N - by - 1$ error vector $\mathbf{e}(n)$ whose kth element is by

$$e_k(n) = \begin{cases} d_k(n) - y_k(n) & \text{if } k \in \chi \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

Define the instantaneous sum of squared error at time n as

$$J(n) = \frac{1}{2} \sum_{k \in \chi} [e_k(n)]^2 \tag{3.15}$$

The objective is to minimize a cost function, obtained by summing $J(n)$ over all time n; that is

$$J_{total} = \sum_n J(n) \tag{3.16}$$

To accomplish this objective the method of steepest descent may be used, which requires knowledge of the gradient matrix, written as

$$\begin{aligned} \nabla_{\mathbf{W}} J_{total} &= \frac{\partial J_{total}}{\partial \mathbf{W}} \\ &= \sum_n \frac{\partial J(n)}{\partial \mathbf{W}} \\ &= \sum_n \nabla_{\mathbf{W}} J(n) \end{aligned}$$

where $\nabla_{\mathbf{W}} J(n)$ is the gradient of $J(n)$ with respect to the weight matrix $\mathbf{W}$. However, in order to develop a learning algorithm that can be used to train the recurrent network in real time, an instantaneous estimate of the gradient must be used, namely, $\nabla_{\mathbf{W}} J(n)$, which results in an approximation to the method of steepest descent.

For the case of a particular weight $w_{ij}(n)$, the incremental change $\Delta w_{ij}(n)$ made at time n is defined as follows:

$$\Delta w_{ij}(n) = -\eta \frac{\partial J(n)}{\partial w_{ij}(n)} \tag{3.17}$$

and η is a fixed positive learning rate.

We note that

$$-\frac{\partial J(n)}{\partial w_{ij}(n)} = \sum_{k \in \chi} e_k(n) \frac{\partial y_k(n)}{\partial w_{ij}(n)} \tag{3.18}$$

The problem is how to derive the $\frac{\partial y_k(n)}{\partial w_{ij}(n)}$. Differentiate Eq. 3.12, yielding

$$\frac{\partial v_k(n)}{\partial w_{ij}(n)} = [\sum_{l \in U} w_{kl} \frac{\partial z_l(n)}{\partial w_{ij}(n)} + \delta_{ik} z_j(n)] \tag{3.19}$$

By using the chain rule for differentiation,

$$\frac{\partial y_k(n+1)}{\partial w_{ij}(n)} = \phi'(v_k(n)) \frac{\partial v_k(n)}{\partial w_{ij}(n)} \tag{3.20}$$

and using Eq. 3.11

$$\frac{\partial z_l(n)}{\partial w_{ij}(n)} = \begin{cases} 0 & \text{otherwise} \\ \frac{\partial y_k(n)}{\partial w_{ij}(n)} & \text{if } l \in \mathbf{U} \end{cases} \tag{3.21}$$

the result is

$$\frac{\partial y_k(n+1)}{\partial w_{ij}(n)} = \phi_k'(v_k(n))[\sum_{l \in U} w_{kl}(n) \frac{\partial y_k(n)}{\partial w_{ij}(n)} + \delta_{ik} z_j(n)] \tag{3.22}$$

where δ_{ik} denotes the Kronecker delta equal to 1 when $i = k$ and zero otherwise.

It is natural to assume that the initial state of the network at time $n = 0$, say, has no functional dependence on the weights; the assumption implies that

$$\frac{\partial y_k(0)}{\partial w_{ij}(0)} = 0 \tag{3.23}$$

These equations hold for all $k \in \chi$, $i \in U$, and $j \in U \cup I$.

We may now create a dynamical system described by a triple indexed set of variable $\{p_{ij}^k\}$ for all $k \in \chi$, $i \in U$, and $j \in U \cup I$, where

$$p_{ij}^k(n) = \frac{\partial y_k(n)}{\partial w_{ij}(n)} \tag{3.24}$$

For every time step n and all appropriate i, j, and k, the dynamics of the system so defined are governed by

$$p_{ij}^k(n+1) = \phi_k'(v_k(n))[\sum_{l \in U} w_{kl}(n) p_{ij}^k(n) + \delta_{ik} z_j(n)] \tag{3.25}$$

with initial conditions

$$p_{ij}^{k}(0) = 0 \tag{3.26}$$

In the case when each unit in the network uses a logistic squashing function (a kind of sigmoidal model), the derivative $\phi_k'(\cdot)$ is given by

$$\phi_k'(v_k(n)) = y_k(n+1)[1 - y_k(n+1)] \tag{3.27}$$

in the calculation.

Finally, we write

$$\triangle w_{ij}(n) = \eta \sum_{k \in \mathbf{X}} e_k(n) p_{ij}^{k}(n) \tag{3.28}$$

and so update the weight w_{ij} in accordance with

$$w_{ij}(n+1) = w_{ij}(n) + \triangle w_{ij}(n) \tag{3.29}$$

The use of the instantaneous gradient $\nabla_{\mathbf{W}} J(n)$ means that the real-time recurrent learning algorithm described here deviates from a non-real-time one based on the true gradient $\nabla_{\mathbf{W}} J_{total}$. While the real-time recurrent learning algorithm is not guaranteed to follow the precise negative gradient of the total error function $J_{total}(\mathbf{W})$ with respect to the weight matrix $\mathbf{W}$, the practical differences between the real-time and non-real-time versions are often slight; indeed, these two versions become more nearly identical as the learning-rate parameter η is reduced. In general, we use the η small enough to make the time scale of the weight changes much smaller than the time scale of the network operation.

Learning algorithms which employ the steepest descent method for the modification of recurrent network weights have been proposed by Pearlmutter [30], Pinda [23] and Sato [31] for continuous time systems. The complex real-time recurrent learning algorithm was deduced by Kechriotis [35] in 1993.

3.3.3 Two Problems

As the recurrent neural networks are used for signal processing, there are two problems to be considered:

- The recurrent neural network is a nonlinear dynamical system. The "stability" of system should be emphasized. Specifically, the feedback loops may cause the instability of the system. According to Atiya [29], the necessary condition for a recurrent neural network of any kind to converge to a unique fixed-point attractor is to satisfy the condition

$$||\mathrm{W}||^2 < \frac{1}{(max|\phi'|)^2} \tag{3.30}$$

where ϕ' is the derivative of the nonlinear activation function with respect to its argument, and $||\mathrm{W}||$ is the Euclidean norm of the full synaptic weight matrix of the network. The latter quantity is defined by

$$||\mathrm{W}|| = (\sum_i \sum_j w_{ij}^2)^{1/2} \tag{3.31}$$

where w_{ij} denoted the synaptic weight of neuron i connected to neuron j. In Eq. 3.30 it is assumed that all the neurons use the same kind of activation function, which is the most common case encountered in practice.

In general, if more neurons are used in a recurrent neural network, the $||\mathrm{W}||$ is larger, which make it more difficult to satisfy the stability criterion of Eq. 3.30.

- The computational requirements of the real-time recurrent learning algorithm arise from the need to store and update all the p_{ij}^k values. In the general case of a fully interconnected network with a total of N neurons and P external input connections, there are a total of $N(N^2 + NP)$ values of the dynamic variable p_{ij}^k to be considered at any time n. The triple indexed set of values $\{p_{ij}^k(n)\}$ may be viewed as an $(N^2 + NP) - by - N$ matrix, each of whose rows correspond to a weight and each of whose

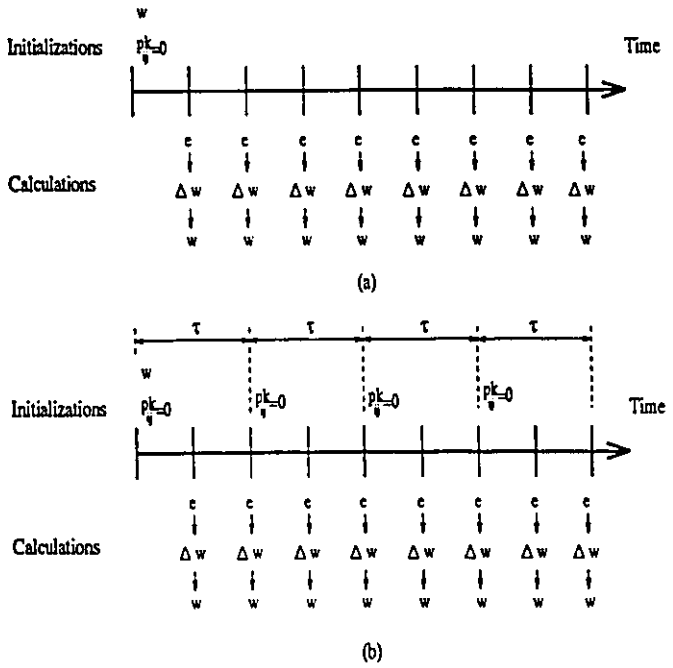


Figure 3.9: (a) The process of Williams's algorithm. (b) The process of Thierry's algorithm.

columns correspond to a neuron in the network. So that, when the network is fully connected and all weights are adaptable, the algorithm has space complexity in $O(N^3)$ and average time complexity in $O(N^4)$ at each time point.

3.3.4 Improved Version of the Real-Time Temporal Supervised Algorithm

In 1993, Thierry [32] proposed a method for improving the Williams's algorithm; the idea behind the method Thierry proposed being that in some cases knowledge about the time dependencies of input and output is available. What he wanted to show is that using this as an advanced temporal knowledge can speed up the learning algorithm. The fact that some input parameters have an effect on the output for Υ time units, and the fact the change of the weights during the learning process is dependent on what happens at the beginning of that process seems contradictory. For this reason, Thierry altered the real-time supervised

learning algorithm in such a way that the weights change at the time when n is mostly dependent on what happened less than Υ times ago. In fact, Thierry re-initialized real-time supervised learning algorithms after a certain number of cycles. He calls this number of cycles τ ($\tau < \Upsilon$, and multiples of a sampling cycle)

Fig. 3.9, (a) shows the process of Williams's algorithm. The impact p_{ij}^k is set to zero at the beginning of the learning phase and is updated at each sampling cycle. $\mathbf{W}$ is also initialized at the beginning of the learning phase and updated at each cycle. Fig. 3.9 (b) shows the process of Thierry's algorithm, after τ cycles the impacts p_{ij}^k are set to zero. It is obvious that one obtains the same results with a τ-value of ∞ as with the original algorithm of Williams.

Thierry demonstrates that the new method can speed up the learning phase. The proposed change, based on a re-initialization of the Williams's algorithm, causes the weight update to follow more precisely the true gradient of total error. This $\tau-$value is dependent on the problem and on the network size. Thierry has shown that the results of a learning session are less influenced by the starting values of the weights if the proposed algorithm is used. The networks trained with the proposed algorithm are less likely to get stuck in local minimal, so it is possible to use higher learning rates and speech up the training.

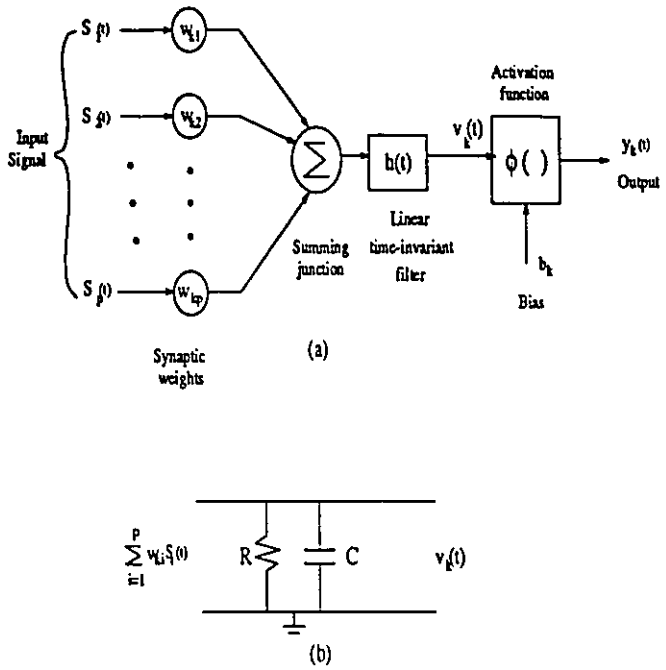


Figure 3.10: (a) RC model of an artificial neuron. (b) RC model of a filter with impulse response h(t).

3.4 New Versions of Recurrent Neural Network and Learning Algorithms

3.4.1 Spatio-Temporal Recurrent Neural Network and its Algorithm

Spatio-temporal model of a neuron

To extend the usefulness of the model of a neuron in Fig. 3.5 for temporal processing, a general method of representing temporal behavior is to introduce a linear, time-invariant filter, as shown in Fig. 3.10. The model is called the spatio-temporal model of a neuron.

The spatio-temporal model may be described with the following two equations:

$$C\frac{dv_k(t)}{dt} + \frac{v_k(t)}{R} = \frac{1}{R}\sum_{l=1}^{p} w_{kl}s_l(t) \tag{3.32}$$

$$\phi_k(v_k(t)) = \frac{1}{1 + exp(-v_k(t) + b_k)} \tag{3.33}$$

These equations can be written in discrete form.

$$v_k(n+1) = (1-1/\tau)v_k(n) + (1/\tau)\sum_{l=1}^{p} w_{kl}s_l(n) \tag{3.34}$$

$$\phi_k(v_k(n)) = \frac{1}{1+exp(-v_k(n)+b_k)} \tag{3.35}$$

where $\tau = RC$.

In introducing the spatio-temporal model to the recurrent neural network presented in Fig. 3.8, the spatio-temporal recurrent neural network is obtained.

Vectors s and $\mathbf{s}_{fb}$ are concatenated to form the $(P+N)-$tuple $\mathbf{z}(n)$.

$$z_l(n) = \begin{cases} s_l(n) & \text{if } l \in \mathbf{I} \\ s_{fb_l}(n) & \text{if } l \in \mathbf{U} \end{cases} \tag{3.36}$$

Let

$$v_k(n+1) = (1-1/\tau)v_k(n) + (1/\tau)\sum_{l\in U\cup I} w_{kl}z_l(n) \tag{3.37}$$

denote the net input on the k^{th} unit at time n, with its output at the next time step being

$$y_k(n+1) = \phi_k(v_k(n)) \tag{3.38}$$

Real-time learning algorithm

The real-time learning algorithm is similar to the algorithm of recurrent neural network. The difference between them is caused by the time constant τ.

Update the weight $w_{ij}(n)$ in accordance with

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \tag{3.39}$$

where

$$\Delta w_{ij}(n) = \eta \sum_{k\in X} e_k(n)p_{ij}^k(n) \tag{3.40}$$

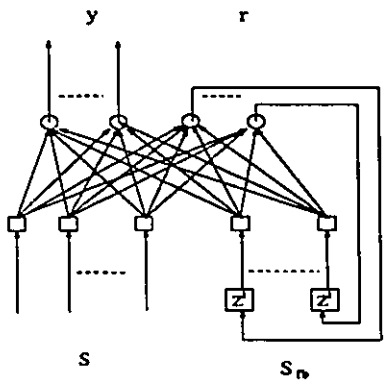


Figure 3.11: Partial recurrent neural network.

For every time step n and all appropriate i, j, and k, the dynamics of the system so defined are governed by

$$p_{ij}^k(n+2) = (1-1/\tau)p_{ij}^k(n+1) + \phi_k'(v_k(n))(1/\tau)[\sum_{l\in U} w_{kl}(n)p_{ij}^k(n) + \delta_{ij}z_j(n)] \quad (3.41)$$

with initial conditions

$$p_{ij}^k(0) = 0 \quad (3.42)$$

$$p_{ij}^k(1) = \phi'(v_k(0))[\sum_{l\in U} w_{kl}(0)p_{ij}^k(0) + \delta_{ik}z_j(0)] \quad (3.43)$$

In the case when each unit in the network uses a logistic squashing function (a kind of sigmoidal model), we use

$$\phi_k'(v_k(n)) = y_k(n+1)[1-y_k(n+1)] \quad (3.44)$$

The principle of the algorithm is the same as the real-time supervisor learning algorithm. But the update of $p_{ij}^k(n)$ is related to both one-step delay $p_{ij}^k(n-1)$ and two-step delay $p_{ij}^k(n-2)$. Rewrite Eq. 3.41 as

$$\tau p_{ij}^k(n+2) = (\tau-1)p_{ij}^k(n+1) + \phi_k'(v_k(n))[\sum_{l\in U} w_{kl}(n)p_{ij}^k(n) + \delta_{ij}z_j(n)] \quad (3.45)$$

Suppose $\tau \to 0$, Eq. 3.45 is equal to Eq. 3.25. In fact, the linear time-invariant filter RC adds a momentum item on the update of $p_{ij}^k(n)$ and $w_{ij}(n)$.

3.4.2 Partially Recurrent Network

Now, some changes on recurrent neural network structure are considered. In 1991, Robinson and Faillside [28] proposed a *partially recurrent neural network.* This structure, shown on Fig. 3.11, may be viewed as a simplified version of the recurrent network.

The partially recurrent network consists of an input layer of source and feedback nodes, and an output layer of computation nodes. Moreover, the computation nodes are split into two groups: K output neurons, and L context neurons. The output neurons produce the overall output vector $\mathbf{y}(n+1)$. The context neurons produce the feedback vector $\mathbf{r}(n)$ after a delay of one time unit applied to each of its elements. The network operates by concatenating the current external input vector $\mathbf{s}(n)$ and the feedback vector $\mathbf{r}(n)$ as follows:

$$\begin{aligned} \mathbf{z}(n) &= [\mathbf{s}(n), \mathbf{s}_{fb}(n)]^T \\ &= [\mathbf{s}(n), \mathbf{r}(n)]^T \end{aligned} \tag{3.46}$$

where $s_0(n) = -1$ or $+1$ is included in $\mathbf{s}(n)$ for the provision of a threshold applied to each neuron in the network. The concatenated input vector $\mathbf{z}(n)$ is multiplied by a matrix of synaptic weights, $\mathbf{W}(n)$, to yield an internal activity vector defined at time n as

$$\begin{aligned} \mathbf{v}(n) &= [\mathbf{v}_o(n), \mathbf{v}_c(n)]^T \\ &= \mathbf{W}(n)z(n) \end{aligned} \tag{3.47}$$

Let $\phi(\cdot)$ denote the activation function, assumed to be the same for all the neurons in the network. Accordingly, the output vector, computed at time $n+1$, is defined by

$$\mathbf{y}(n+1) = \phi(\mathbf{v}_o(n)) \tag{3.48}$$

Similarly, the feedback vector (before delay) is defined by

$$\mathbf{r}(n+1) = \phi(\mathbf{v}_c(n)) \tag{3.49}$$

Finally, the output vector $\mathbf{y}(n+1)$ is compared with a desired response vector $\mathbf{d}(n+1)$ in accordance with a cost function of interest. In the original description of the partial recurrent network presented in Robinson and Fallside, the network outputs are treated as probabilities. Specifically, the relative entropy is used to define the cost function as follows:

$$H_{d||y}(n+1) = \sum_{k=1}^{K}(d_k(n+1)logy_k(n+1) - [(1-d_k(n+1))log(1-y_k(n+1))]) \quad (3.50)$$

where $y_k(n+1)$ is the kth element of the output vector $\mathbf{y}(n+1)$, and $\mathbf{d}_k(n+1)$ is the corresponding value of the desired response. The objective is to adapt the synaptic weight matrix $W(n)$ so as to minimize the relative entropy of Eq. 3.50.

Robinson and Fallside describe the application of the partial recurrent network as the basis of speaker-independent phoneme and word recognition system. Leerking and Jabri [33] use the network as an adaptive world model to provide an evolving state for temporal difference learning applied to continuous speech recognition.

3.5 Summary

This chapter describes the principle of neural networks for the adaptive signal processing. First, the basic structures and learning processes of neural networks are introduced. The sigmoidal model of an artificial neuron is used as the element model of following neural networks. Then, the nonlinearity and learning ability of neural networks are discussed. Because of two abilities, neural networks are able to outperform conventional adaptive filters. The self-organized adaptation of a neural network is the basis of building many-level neural network system, which has stronger learning and approximation ability.

Particularly, we have made a thorough investigation of the dynamical recurrent

neural network and its real-time training algorithm. Recurrent neural networks exhibit highly nonlinear dynamical behavior. They have the ability to deal with time-varying input or output through their own natural temporal operation. The real-time learning algorithm gives the recurrent neural network an on-line (in-situ) learning fashion. So that, for a nonstationary signal, a recurrent neural network may learn to adapt to statistical variation of the incoming time series while performing its filtering role at the same time.

Unfortunately, the number of neurons needed for a recurrent neural network as a adaptive predictor is often so large that the cost of computation is too excessive. Also, it may be hard to control the stability of a larger recurrent neural network. These deficiencies block application of recurrent neural network on real-time signal processing.

We have also introduced an improved version of real-time temporal supervised algorithm, and new versions of the recurrent network. These new structures may improve the characteristics of recurrent neural network.

Chapter 4

Pipelined Recurrent Neural Network

As mentioned in the last chapter, real-time recurrent neural network can provide on-line (continuous) learning. A practical limitation of the real-time recurrent network, however is that its computation complexity increases exponentially with the number of neurons in the network. In order to develop a learning algorithm that can be used to train the recurrent network in real time, an instantaneous estimate of the gradient is introduced, which results in an approximation to the method of steepest descent.

In this chapter, we will begin by developing a Pipelined Recurrent Neural Network (PRNN). The design of the pipelined structure follows the important engineering *principle of divide and conquer* and biological *principle of neural network modules.* Then, the real-time algorithm of Pipelined Recurrent Neural Network is derived. The Pipelined Recurrent Neural Network enhances the computational ability of the conventional recurrent neural network and overcomes its deficiency.

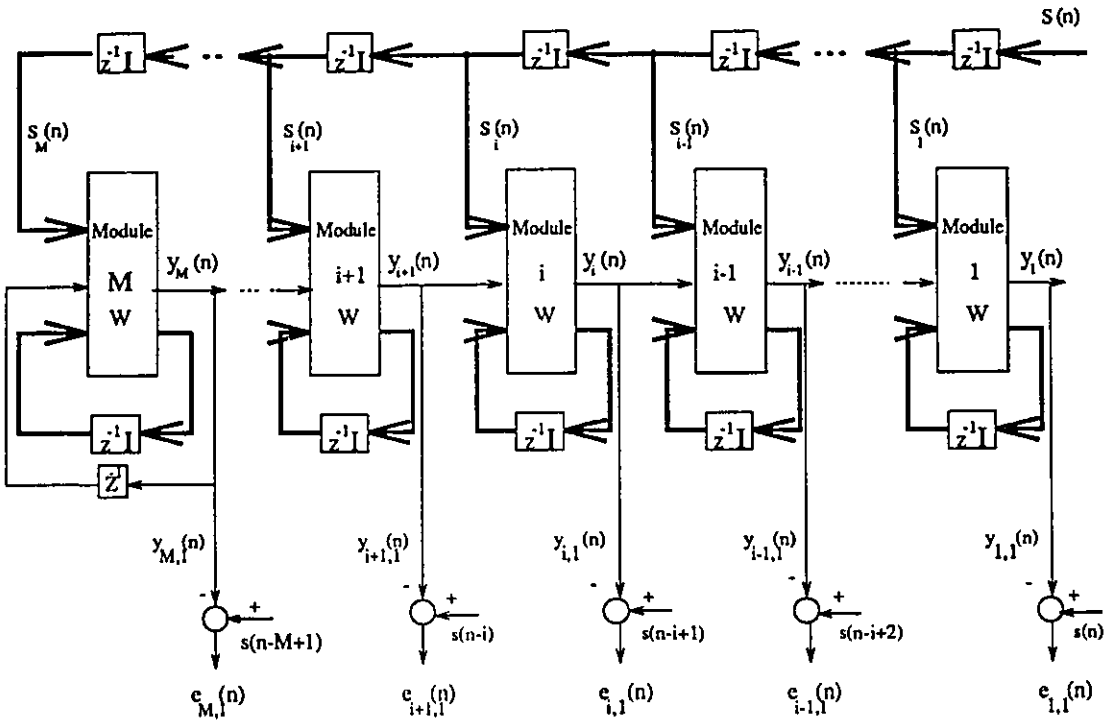


Figure 4.12: A pipelined neural network.

4.1 Construction of Pipelined Recurrent Neural Network

In 1993, a pipelined neural network structure was proposed [38], which is depicted in Fig. 4.12. This pipelined structure involves a total of M levels of processing. Each level has a module [44] and a comparator, and a module consists of a recurrent neural network [46].

Figure 4.13 shows the details of Level i. In every module, there are N neurons and a single output neuron. In addition to the p external inputs, there are N feedback inputs. To accommodate a bias for each neuron, besides the $p + N$ inputs, we include one input whose value is always $+1$. $N - 1$ outputs of each module are fed back to itself input, and the remaining output is applied to the next module.

In fact, Module i ($1 \leq i \leq M - 1$) is a *partially* recurrent neural network, described in Section 3.4. In the case of module M, a delayed version of the output is also fed back to

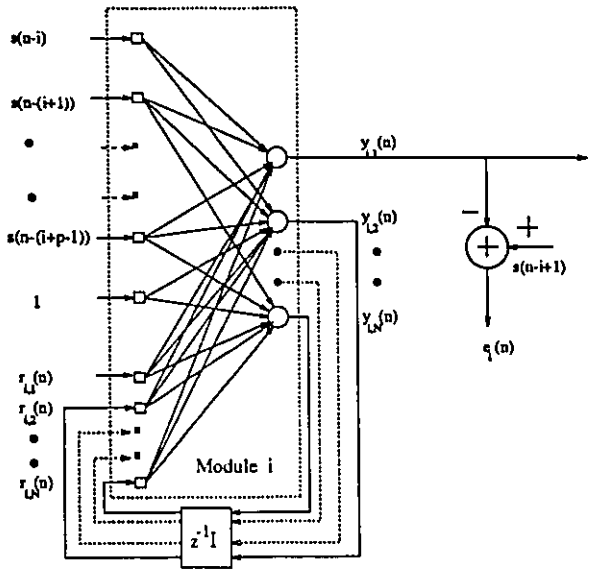


Figure 4.13: The construction of Level i.

the input. It is an "original" recurrent neural network.

All the modules operate similarly in that they all have the same number of external inputs and feedback nodes. Moreover, they are all designed to have exactly the same weight matrix. In making this design option, we are influenced by an important neurobilogical principle of modularity that suggests the repeated use of a modules of prescribed structure [44].

The activation function of every neuron in each module is a logistic function:

$$y_{i,k}(n+\Delta n) = \phi(v_{i,k}(n)) = \frac{1}{1+exp(-v_{i,k}(n))} \qquad i = 1,\ldots,M; \quad k = 1,\ldots,N \qquad (4.51)$$

where $v_{i,k}(n)$ is the net internal activation of the k^{th} neuron, and $y_{i,k}$ is the output of the k^{th} neuron of the i^{th} module at the n^{th} time point. The Δn is the processing time of a network.

Let W denote the synaptic weight matrix for each module $(N - by - (p + N + 1))$. The element w_{kl} represents the weight of the connection to the k^{th} neuron node from the

the l^{th} input unit. The weight matrix W can be written as:

$$\mathrm{W} = [\mathrm{w}_1, \cdots, \mathrm{w}_k, \cdots, \mathrm{w}_N]^T$$

where w_k is a vector defined as

$$\mathrm{w}_k = [w_{k1}, w_{k2}, \cdots, w_{k(p+N+1)}]$$

This pipelined structure is referred to as a Pipelined Recurrent Neural Network (PRNN).

4.2 Neurobilogical Considerations of PRNN

This pipelined neural network structure is originated from the principle of self-organized adaptation described in last Chapter. The fully neural network is itself built in the form of multiple-level processing. Every level basically consists of a comparator and a small neural network. The small neural network is a module, which provides a dynamical sub-model of the outside world. The multiple-level structure enhances the learning ability of a fully network.

According to Van Essen [45], the same principle is also reflected in the design of the brain, as follows:

- Separate modules are created for different subtasks, permitting the neural architecture to be optimized for particular types of computation.
- The same module is replicated several times over.
- A coordinated and efficient flow of information is maintained between the modules.

Thus, the design of the pipelined structure follows an important engineering principle, namely, the *principle of divide and conquer*.

To solve a complex problem, break it into a set of simpler problems.

All of the three elements of the principle of divide and conquer, viewed in a biological context, feature clearly in the Pipelined Recurrent Neural Network, as described here:

- The PRNN is composed of M separate modules, each of which is designed to perform a one-step nonlinear prediction on an appropriately delayed version of the input signal vector.
- The modules are identical, each designed as a recurrent neural network with a single output neuron. The last module in the chain is wired slightly differently from the rest in that its output is delayed by one time unit and then fed back to its own input, thereby ensuring that all the modules have the same size of input layer.
- Information flow into and out of the modules proceeds in a synchronized fashion.

4.3 Mathematical Model of PRNN

Suppose that a set of time series consisting of observation samples $s(1), s(2), \ldots, s(n)$ is given. M input vectors are formed using subsets of this time series:

$$\begin{aligned}
\mathbf{s}_1(n) &= [s(n-1), s(n-2), \cdots, s(n-p)]^T \\
\mathbf{s}_2(n) &= [s(n-2), s(n-3), \cdots, s(n-(p+1))]^T \\
\cdots &\quad \cdots \\
\mathbf{s}_M(n) &= [s(n-M), s(n-(M+1)), \cdots, s(n-(M+p-1))]^T
\end{aligned}$$

At the n^{th} time point, the external input of Module i is the $p-by-1$ vector

$$\mathbf{s}_i(n) = [s(n-i), s(n-(i+1)), \cdots, s(n-(i+p-1))]^T \tag{4.52}$$

where p is the prediction order. The M input vectors are sequentially sent to M modules.

The other input is the $N - by - 1$ "feedback vector"

$$\mathbf{r}_i(n) = [r_{i,1}(n), r_{i,2}(n), \cdots, r_{i,N}(n)]^T \tag{4.53}$$

At the n^{th} time point, the outputs of neurons in Module i are denoted as a vector:

$$\mathbf{y}_i(n) = [y_{i,1}(n), \cdots, y_{i,k}(n), \cdots, y_{i,N}(n)]^T \tag{4.54}$$

where the $y_{i,k}(n)$ is the output of neuron k of module i. If the processing time of a network Δn is far smaller than the sampling period of the signal $s(t)$, we suppose

$$y_{i,k}(n) \simeq y_{i,k}(n + \Delta n) = \phi(v_{i,k}(n)) \tag{4.55}$$

$$v_{i,k}(n) = \sum_{l=1}^{p} w_{kl} s(n - (i + l - 1)) + w_{k(p+1)} + \sum_{l=p+2}^{p+N+1} w_{kl} r_{i,l-(p+1)}(n) \tag{4.56}$$

For convenience of notation, the vector $\mathbf{y}_i(n)$ is denoted as follows:

$$\mathbf{y}_i(n) = \Phi(\mathbf{W}, \mathbf{s}_i(n), \mathbf{r}_i(n)) \tag{4.57}$$

$$= [\phi(\mathbf{w}_1, \mathbf{s}_i(n), \mathbf{r}_i(n)), \cdots, \phi(\mathbf{w}_k, \mathbf{s}_i(n), \mathbf{r}_i(n)), \cdots, \phi(\mathbf{w}_N, \mathbf{s}_i(n), \mathbf{r}_i(n))]^T \tag{4.58}$$

where Φ denotes a vector and the nonlinear function ϕ represents the activation function of a neuron.

As mentioned previously, the output of the first neuron $y_{i,1}(n)$, which is the real output of Module i, is fed directly to the following comparator and Module $i - 1$. Each level computes an error signal or innovation defined by

$$e_i(n) = s(n - i + 1) - y_{i,1}(n) \tag{4.59}$$

where the signal sample $s(n - i + 1)$ is the desired response of Module i.

Table 4-1: Summary of the inputs and outputs of the modules in the pipelined recurrent neural network

	Module 1	Module $1 < i < M$	Module M
External Inputs	$s(n-1)$ ⋮ $s(n-p)$ 1	$s(n-i)$ ⋮ $s(n-(i+p-1))$ 1	$s(n-M)$ ⋮ $s(n-(M+p-1))$ 1
Feedback Inputs	$y_{1,1}(n)$ $y_{1,2}(n-1)$ ⋮ $y_{1,N}(n-1)$	$y_{i+1,1}(n)$ $y_{i,2}(n-1)$ ⋮ $y_{i,N}(n-1)$	$y_{M,1}(n-1)$ $y_{M,2}(n-1)$ ⋮ $y_{M,N}(n-1)$
Outputs	$y_{1,1}(n)$	$y_{i,1}(n)$	$y_{M,1}(n)$

The feedback inputs of Module i consist of the first neuron's output of Module $i+1$ and the one-step delayed output signals of Neuron $2, \ldots, N$ of Module i fed back to itself. Thus, the $N - by - 1$ feedback input vector $\mathbf{r}_i(n)$ of the i^{th} module is defined as:

$$\begin{aligned} \mathbf{r}_i(n) &= [y_{i+1,1}(n), \mathbf{r}'_i(n)]^T \\ &= [y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)]^T \end{aligned} \tag{4.60}$$

where $\mathbf{r}'_i(n)$ denotes the part of the feedback inputs, originating from Module i itself.

The modules operate in a highly pipelined manner. Table 4-1 presents the relation of inputs and outputs on each module of PRNN. There are some special assumptions on the models of Module M and Module 1, that should be noted:

- Module M operates as a standard real-time recurrent neural network [46]. The output

vector $\mathbf{y}_M(n)$ of this module is fed back to itself input as feedback input vector after a delay of one time unit. At the same time, the output signal $y_{M,1}(n)$ is also fed directly into Module $M-1$. At the n^{th} time point, the feedback input vector of Module M is defined as:

$$\mathbf{r}_M(n) = \mathbf{y}_M(n-1) \tag{4.61}$$

- The outputs of neurons in Module 1 are denoted as

$$\mathbf{y}_1(n) = [\phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}_1(n)), \cdots, \phi(\mathbf{w}_N, \mathbf{s}_1(n), \mathbf{r}_1(n))]^T \tag{4.62}$$

where

$$\mathbf{s}_1(n) = [s(n-1), s(n-2), \cdots, s(n-p)]^T \tag{4.63}$$

$$\mathbf{r}_1(n) = [y_{2,1}(n), y_{1,2}(n-1), \cdots, y_{1,N}(n-1)]^T \tag{4.64}$$

$$\mathbf{w}_k = [w_{k1}, w_{k2}, \cdots w_{k(P+N+1)}]^T \tag{4.65}$$

The corresponding desired signal of Module 1 is $s(n)$.

The output of visible(first) neuron of Module 1 is the overall output of the pipelined recurrent network $y(n)$ as shown by

$$\begin{aligned} y(n) &= y_{1,1}(n) \\ &= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}_1(n)) \end{aligned} \tag{4.66}$$

where $\mathbf{w}_1$ is a function of the signal sample $s(n)$.

The PRNN differs from the conventional real-time recurrent neural network in that it is characterized by a *nested nonlinearity*. Specifically, we may express the functional dependence of the output $y(n)$ of the network in Figure 4.12 on the external inputs as follows:

$$y(n) = \phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}_1(n)) \tag{4.67}$$

$$
\begin{aligned}
&= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}'_1(n), \phi(\mathbf{w}_1, \mathbf{s}_2(n), \mathbf{r}_2(n))) \\
&= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}'_1(n), \phi(\mathbf{w}_1, \mathbf{r}_2(n), \mathbf{r}'_2(n), \phi(\mathbf{w}_1, \mathbf{s}_3(n), \mathbf{r}'_3(n), \\
&\quad \cdots, \phi(\mathbf{w}_1, \mathbf{s}_M(n), \mathbf{y}_M(n-1)), \cdots)))
\end{aligned}
$$

where $\mathbf{r}'_i(n)$ $(i = 1, \cdots, M-1)$ is a function of the weight matrix $\mathbf{W}$.

It is also important to note that the desired response needed to perform the continuous training of the PRNN is in fact derived from the incoming time series itself. Thus, the nonlinear adaptive predictor described here embodies a self-originated learning process as described in Section 3.2.2.

4.4 Real-Time Learning of PRNN

The training process of PRNN aims at finding the optimal weights of the network and the output signal $\mathbf{y}(n)$ with the input signal series $\{s(n)\}$.

At the instant n^{th} time point, according to the analysis of Section 4.3, the output of the k^{th} neuron $(1 \leq k \leq N)$ at Module i is defined by

$$
\begin{aligned}
\hat{y}_{i,k}(n) &= \Phi(\hat{v}_{i,k}(n)) \\
\hat{v}_{i,k}(n) &= \sum_{l=1}^{p} w_{kl} s(n-(i+l-1)) + w_{k(p+1)} + \sum_{l=p+2}^{p+N+1} w_{kl} r_{i,(l-(p+1))}(n) \\
e_i(n) &= s(n-(i+1)) - \hat{y}_{i,1}(n)
\end{aligned}
$$

where $\hat{y}_{i,k}(n)$ denotes the prior prediction output of the k^{th} neuron during the training process.

After every module of PRNN finishes its prior prediction calculation, a series of error signals are obtained $e_i(n)$ $(1 \leq i \leq M)$. An overall cost function for pipelined recurrent network of Figure 4.12 is thus defined by

$$
\varepsilon(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{4.68}
$$

where λ is an exponential forgetting factor that lies in the range $0 < \lambda \leq 1$. The λ^{i-1} is, roughly speaking, a measure of the memory of the pipelined recurrent network.

A gradient estimation algorithm, is used to calculate the $\Delta \mathbf{W}$ along the negative of the gradient $\nabla_{\mathbf{W}}\varepsilon(n)$ [48]:

The change applied to the kl^{th} element in the weight matrix can be written as

$$\begin{aligned} \Delta w_{kl} &= -\eta \frac{\partial}{\partial w_{kl}} (\sum_{i=1}^{M} \lambda^{i-1} e_i^2(n)) \\ &= -2\eta \sum_{i=1}^{M} \lambda^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial w_{kl}} \end{aligned} \tag{4.69}$$

where η is a fixed learning-rate parameter, $1 \leq k \leq N$ and $1 \leq l \leq (p+N+1)$.

The real-time supervised learning algorithm, described in Sec. 3.3.2, is used to calculate $\frac{\partial e_i(n)}{\partial w_{kl}}$ on every neural network module in the pipelined structure. We thus write

$$\begin{aligned} \frac{\partial e_i(n)}{\partial w_{kl}} &= -\frac{\partial \hat{y}_{i,1}(n)}{\partial w_{kl}} \\ \frac{\partial \hat{y}_{i,1}(n)}{\partial w_{kl}} &= \phi'(\hat{v}_{i,1}(n)) * \\ & \quad [\sum_{\alpha=1}^{N} w_{1(\alpha+p+1)} \frac{\partial r_{i,\alpha}(n)}{\partial w_{kl}} + \delta_{kl} s(n-i+(l-1))] \end{aligned} \tag{4.70}$$

where $\phi'(\hat{v}_{i,1}(n))$ is the derivative of the nonlinear function $\phi(\cdot)$ for the output neuron of the i^{th} module at the n^{th} time point, and $r_{i,\alpha}(n)$ is the feedback input of input Unit α in the i^{th} module at the n^{th} time point.

It is hard to calculate $\frac{\partial r_{i,\alpha}(n)}{\partial w_{kl}}$ directly. The recursive algorithm presented in Chapter 3.3.2, is introduced. Let,

$$p_{kl}^{i,\alpha}(n) = \frac{\partial r_{i,\alpha}(n)}{\partial w_{kl}(n)} \tag{4.71}$$

$$p_{kl}^{i,\alpha}(n) = \phi'(v_{i,\alpha}(n))[\sum_{\alpha=1}^{N} w_{1(\alpha+p+1)}(n) p_{kl}^{i,\alpha}(n-1) + \delta_{k\alpha} s_l(n-1-i+(l-1))] \tag{4.72}$$

As the $s_i(n-1)$ is equal to the $s_{i+1}(n)$, and modules are sequentially calculated, we suppose:

$$p_{kl}^{i,\alpha}(n-1) = p_{kl}^{i+1,\alpha}(n) \tag{4.73}$$

where $1 \leq i \leq M-1$.

$$p_{kl}^{M,\alpha}(n-1) = p_{kl}^{M,\alpha}(n) \tag{4.74}$$

This supposition is certified in the adaptive calculation of the PRNN-based prediction, which is described in Chapter 5.

So Eg. 4.72 is rewritten as

$$p_{kl}^{i,\alpha}(n) = \phi'(v_{i,\alpha}(n))[\sum_{\alpha=1}^{N} w_{1(\alpha+p+1)}(n) p_{kl}^{i+1,\alpha}(n) + \delta_{k\alpha} s_l(n-1-i+(l-1))] \tag{4.75}$$

where $1 \leq i \leq M-1$.

$$p_{kl}^{M,\alpha}(n) = \phi'(v_{i,\alpha}(n))[\sum_{\alpha=1}^{N} w_{1(\alpha+p+1)}(n) p_{kl}^{M,\alpha}(n-1) + \delta_{k\alpha} s_l(n-1-i+(l-1))] \tag{4.76}$$

As $n = 0$,

$$p_{kl}^{M,\alpha}(0) = 0$$

The notation δ_{kl} denotes the Kronecker delta

$$\delta_{kl} = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases}$$

and

$$\phi'(\hat{v}_{i,1}(n)) = \phi(\hat{v}_{i,1}(n))[1 - \varphi(\hat{v}_{i,1}(n))], \qquad (1 \leq i \leq M)$$

Sequentially, the supervised training is taken on every module and each $\frac{\partial e_i(n)}{\partial w_{kl}}$ $(1 \leq i \leq M)$ is obtained. So the Δw_{kl} is calculated from Eq. 4.69.

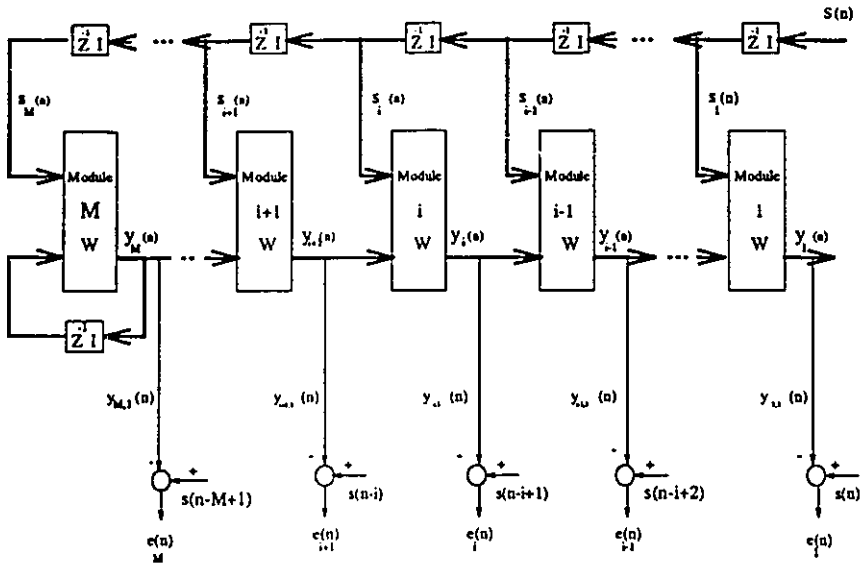


Figure 4.14: A simplified model of the pipelined recurrent neural networks.

Finally, the weight matrix is updated as

$$W \leftarrow W + \Delta W \tag{4.77}$$

The learning process of the PRNN is completely different from the *back-propagation through time* of training a recurrent neural network. For the process of PRNN, the *real-time recurrent learning* is taken on each module (small recurrent network). However, the latter uses a trick to turn an arbitrary recurrent network into an equivalent feed-forward network [52].

4.5 Simplified Pipelined Recurrent Neural Networks

The pipelined architecture of Figure 4.12 follows naturally from the self-organized adaptive system of Figure 3.7. We may reduce its storage requirement using the simplified model [9], shown in Figure 4.14. This configuration is the same as that of Figure 4.12, except for the fact that the feedback input of each module is different with that of Figure 4.12. Except

for Module M, in Module i ($1 \leq i \leq M-1$), all of the feedback inputs are obtained directly from the previous Module $i+1$. The memory, which stores the output vector of Module i, is in effect saved. We may define an $N-by-1$ "feedback input vector" $\mathbf{r}_i(n)$ of the i^{th} module as:

$$\begin{aligned} \mathbf{r}_i(n) &= \mathbf{y}_{i+1}(n) \\ &= [y_{i+1,1}(n), y_{i+1,2}(n), \ldots, y_{i+1,N}(n)]^T \end{aligned} \tag{4.78}$$

where $1 \leq i \leq M-1$.

Being the same as described in Section 4.1, Module M operates as a standard real-time recurrent neural network. After a delay of one time unit, the output vector $\mathbf{y}_M(n)$ of this module is fed back to itself as a feedback input vector.

$$\mathbf{r}_M(n) = \mathbf{y}_M(n-1) \tag{4.79}$$

Correspondingly, the functional dependence of the output $y(n)$ of the network may be expressed as follows:

$$\begin{aligned} y(n) &= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \mathbf{r}_1(n)) \\ &= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \Phi(\mathbf{W}, \mathbf{s}_2(n), \mathbf{r}_2(n))) \\ &= \phi(\mathbf{w}_1, \mathbf{s}_1(n), \Phi(\mathbf{W}, \mathbf{s}_2(n), \Phi(\mathbf{W}, \mathbf{s}_3(n), \cdots, \\ &\quad \Phi(\mathbf{W}, \mathbf{s}_M(n), \mathbf{y}_M(n-1)), \cdots))) \end{aligned} \tag{4.80}$$

4.6 Summary

The pipelined recurrent neural network offers the following attributes.

- *Nested nonlinearity.* The overall input-output relation of the PRNN exhibits a form of nested nonlinearity, similar to that found in a multilayer perception. The nested nonlinearity is described in Eq. 4.67. This characteristic has the beneficial effect of enhancing the computing power of the PRNN.

- *Smoothed cost function.* The cost function of the PRNN is defined as the exponentially weighted sum of the squared estimation errors computed by the individual modules as shown in Eq. 4.68. Hence it may be argued that the cost function $\varepsilon(n)$ so defined is "closer" to that used in deriving the more elaborate steepest descent version of the learning algorithm originally formulated for recurrent neural networks by Williams and Zipser.

- *Overlapping Data Windows.* From Eq. 4.67 it is readily apparent that the individual modules of the PRNN operate on data windows that overlap with each other by one sampling period. Specifically, at each iteration the PRNN processes a total of $p+M-1$ input samples, where p is the order of each vector **s** and it is the number of external inputs each module.

- *Improved Stability (convergence).* The necessary condition for a recurrent neural network of any kind to convergence to a unique fixed-point attractor is described in Eq. 3.30 and 3.31. Typically, the PRNN uses two or three neurons per module, which is much smaller than the corresponding number of neurons in a single recurrent network for solving a difficult signal processing task. This means that the Euclidean weight matrix **W** of each module in the PRNN is usually smaller than the corresponding value of a single recurrent network of comparable size (i.e., the same number of neurons). Accordingly, the PRNN is more likely to satisfy the stability criterion of Eq. 3.30 than a single recurrent neural network of comparable size.

- *Weight sharing.* Another important virtue of the PRNN is that all of its modules share exactly the same synaptic weight matrix W. Thus, with M modules and N neurons per module, the computational complexity of training the PRNN is $O(MN^4)$. On the other hand, a conventional recurrent structure of the same size has MN neurons, and its computational complexity is therefore $O(M^4N^4)$. Clearly, for $M > 1$ the PRNN has a significantly reduced computational complexity compared to a single recurrent neural network of the same size; the reduction in computational complexity becomes more pronounced with increasing M.

Chapter 5

Nonlinear Adaptive Prediction of Nonstationary Signals

In this chapter, we describe a computationally efficient scheme for the nonlinear adaptive prediction of nonstationary signals whose generation is governed by a nonlinear dynamical mechanism. The complete prediction consists of two subsections, one of which performs a nonlinear mapping from the input space to an intermediate space with the aim of linearizing the input signal, and the other subsection performs linear mapping from the new space to the output space. The nonlinear subsection consists of a pipelined recurrent neural network, and the linear section consists of a conventional tapped-delay-line filter. The prediction is an adaptive system, with adjustments to the free parameters of the network being made in *on-line* fashion while the signal processing is being performed. Then, we discuss thoroughly the algorithm of adaptive nonlinear prediction. The discussion includes the initialization, convergence and computational requirement of the adaptive prediction algorithm. Finally, the dynamical behavior of the PRNN-based predictor is evaluated via the adaptive prediction of various nonstationary signals.

5.1 Nonlinear Adaptive Predictor

5.1.1 Principle of Operation

Consider a nonstationary time series made up of a set of uniformly spaced samples $s(n), s(n-1), \ldots, s(1)$. Generally speaking, one-step prediction can be described as

$$\hat{s}(n+1) = \mathcal{F}(s(n), s(n-1), \cdots, s(1)) \tag{5.81}$$

where $\mathcal{F}$ is a nonlinear function as the generation of the signal is governed by a nonlinear dynamical mechanism. As mentioned in the Chapter 1, as there is not a linear relation between the $\hat{s}(n+1)$ and $s(n), s(n-1), \ldots, s(1)$, serious errors could be produced as using some classical and practical (such as LMS, RLS) algorithms for the nonlinear prediction.

For solving this nonlinear prediction of nonstationary signals, we proposed a new nonlinear prediction scheme. Suppose that signal samples $s(n), s(n-1), \ldots, s(1)$ are in an input space. The principle of the new way is to build an intermediate space. The intermediate space is constituted with the variance $y(n)$ and some past values of it $y(n-1), y(n-2), \ldots$, which are obtained with a nonlinear mapping from the input space. In the new intermediate space, there is a linear relation between the predicted value $\hat{s}(n+1)$ and variances $y(n), y(n-1), y(n-2), \ldots$. Then, many classically linear prediction algorithms could be used on the intermediate space.

Therefore, the complex nonlinear prediction is divided into two relative calculation steps: a nonlinear mapping between the input space and the intermediate space, and a linear mapping from the intermediate space to the final output space, which contains the predicted data. Both of these operations are performed adaptively on a continuous basis.

The principle of the new method is illustrated in Fig. 5.15, and we may write,

$$y(n) = \phi(s(n), s(n-1), \cdots s(1)), \tag{5.82}$$

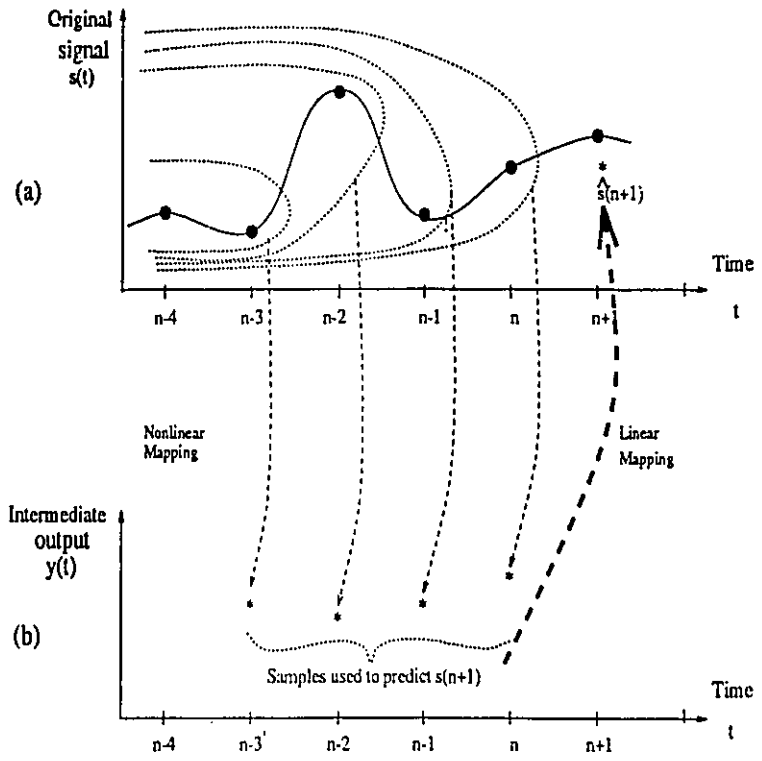


Figure 5.15: Illustrating the principle of the adaptive nonlinear prediction.

where ϕ is a nonlinear function. Each dotted curve expresses that all of information included in past signal samples are used for the nonlinear mapping. For example, the dotted curve of Point n, presents the information of past samples $s(n), s(n-1), s(n-2), \cdots$ to support the nonlinear mapping at the time point n.

$$\hat{s}(n+1) = \sum_{k=0}^{q-1} w_{l,k} y(n-k) \tag{5.83}$$

where q is the number of necessary variances on the intermediate space, and $w_{l,0}, w_{l,1}, \ldots, w_{l,(q-1)}$ constitute a set of linear weights.

Based on the principle, a nonlinear adaptive filtering system is designed. Fig. 5.16 shows the block diagram of the system. The pipelined recurrent neural network (PRNN),

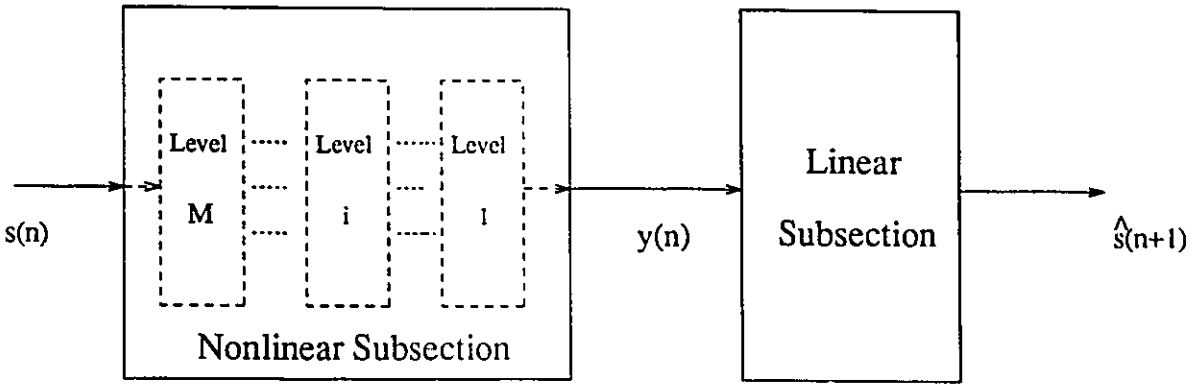


Figure 5.16: Block diagram of the new nonlinear adaptive predictor.

consisting of many levels of recurrent signal processing, constitutes the nonlinear subsection of the system. In addition, the system includes a linear subsection represented by a conventional *tapped-delay-line(TDL) filter*. These two subsections perform two distinct functions:

- The PRNN performs the nonlinear mapping from the input space to the intermediate space with the aim of linearizing the input signal. It does so by filtering a set of samples of the input $s(t)$, which represented by $s(n), s(n-1), s(n-2), \ldots$. These samples extend into the "infinite past" for varying discrete-time n by virtue of the feedback built into the design of each module of the PRNN. Thus, the PRNN has *infinite memory* of a fading nature.

- The TDL filter performs a linear mapping (calculation) from the new intermediate space to the output space. Specifically, it uses a linear combination of the sample $y(n), y(n-1), \ldots, y(n-q+1)$ derived from the output of the PRNN to produce a prediction $\hat{s}(n+1)$ of the original signal, one step into the future. In contrast to the PRNN, the TDL filter has *finite memory*.

Of course, both of these operations are performed adaptively on a continuous basis. Thus, the cascade combination of the PRNN and the TDL filter may be used to perform

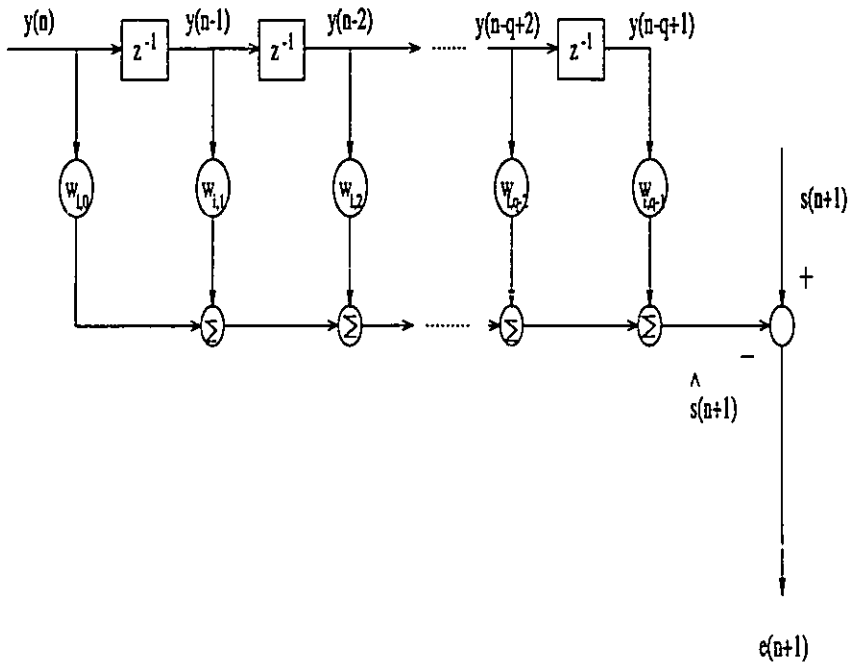


Figure 5.17: Linear subsection: a tapped-delay-line filter.

the nonlinear adaptive prediction of a nonstationary signal.

5.1.2 Construction of a Nonlinear Adaptive Predictor

The detail of nonlinear subsection is showed in Fig. 4.12 or in Fig. 4.14, which is the simplified PRNN as explained later. The linear subsection of the neural-network based-predictor is a tapped-delay-line filter, shown in Figure 5.17. The weight vector of the linear filter is denoted as [7]

$$\mathbf{w}_l = [w_{l,0}, w_{l,1}, \cdots, w_{l,q-1}]^T \tag{5.84}$$

The pipelined recurrent neural network of Fig. 4.12 performs a nonlinear mapping that is *coarse-to-fine*, depending on the number of modules M used and the size of each module. The PRNN differs from the conventional recurrent neural network in that it is characterized by a *nested nonlinearity*, as described in the previous chapter.

The inputs of this filter consist of the present output $y(n)$ computed by the pipelined recurrent network and $q-1$ past values $y(n-1), y(n-2), \ldots, y(n-q+1)$. During the actual calculation, the input of the linear subsection is the output of PRNN $y(n)$ at a filtering process step. For expressing clearly, we describe the output as $y_{filt}(n)$; more will be said on the computation of $y_{filt}(n)$ in Section 5.2. So,

$$\mathbf{y}_{filt}(n) = [y_{filt}(n), y_{filt}(n-1), \cdots, y_{filt}(n-q+1)]^T \tag{5.85}$$

The output of a linear filter is defined as

$$\hat{s}(n+1) = \mathbf{w}_l^T * \mathbf{y}_{filt}(n) \tag{5.86}$$

The tapped-delay-line filter fine-tunes the final result. The estimate $\hat{s}(n+1)$ is a nonlinear adaptive prediction of the actual sample $s(n+1)$ of the input signal, which is computed in an on-line fashion for varying time n.

The pipelined recurrent neural network and tapped-delay-line filter can be trained in an on-line fashion. So the nonlinear predictor described here is called PRNN-based adaptive filter (predictor).

5.2 Algorithmic Design of Adaptive Predictor

For adaptive processing of a nonstationary signal, the operation of adaptive estimation involves two basic processes:

- *An adaptive learning process.* The purpose of the process is to provide a mechanism for on-line training of a nonlinear subsystem and the adaptive control of an adjustable set of weights used in the linear subsystem.
- *A generation process.* The process is designed to produce an output in response to a sequence of input data.

These two processes work interactively on each signal sample. Thus the computation performed by the PRNN at time n is processed in a self-organized manner in three stages as follows:

1. Prediction. Given the input vectors $\mathbf{s}_1(n), \mathbf{s}_2, \ldots, \mathbf{s}_M(n)$ and the desired responses $s(n), s(n-1), \ldots, s(n-M+1)$, the individual modules of the PRNN compute the one-step prediction errors $e_1(n), e_2(n), \ldots, e_M(n)$ respectively.

2. Weight updating. The prediction errors are used to compute the matrix of local gradients, $\frac{\partial e(n)}{\partial \mathbf{W}}$, and therefore the correction $\Delta\mathbf{W}$ applied to the old weight matrix $\mathbf{W}$ of each module. Accordingly, the updated value, $\mathbf{W}_u$ is computed.

3. Filtering. The updated weight matrix W_u and the vectors $\mathbf{s}_1(n), \mathbf{s}_2(n), \ldots, \mathbf{s}_M(n)$ are used to compute the filtered output $y_{filt}(n)$ as the output of first (visible) neuron in module 1 of the PRNN. This is done by proceeding through the M modules, one by one.

Turning next to the linear subsection, the output $\hat{s}(n+1)$ of the TDL filter, produced in response to the sequence $y_{filt}(n), y_{filt}(n-1), \ldots, y_{filt}(n-q+1)$, is compared against the desired response $s(n+1)$. The well-known least-mean-square(LMS) algorithm is used to adjust the tap weights of the TDL filter. This is done in an attempt to minimize the error signal (i.e. the difference between $s(n+1)$ and $\hat{s}(n+1)$ in a "mean-square sense").

Algorithm for the on-Line Training and Nonlinear Prediction

1. *Adaptive Calculation of Nonlinear Subsection*

   - **Prediction Process**

     At the instant n^{th} time point, the input vectors $\mathbf{s}_1(n), \ldots, \mathbf{s}_M(n)$ are obtained from the input data. The prior prediction is taken from Level M in stages to

Level 1. The output vector of Module i and the error signal of Level i are defined by, respectively,

$$\begin{aligned} \hat{\mathbf{y}}_i(n) &= \Phi(\mathbf{W}, \mathbf{s}_i(n), \mathbf{r}_i(n)) \\ e_i(n) &= s(n-i+1) - \hat{y}_{i,1}(n) \end{aligned}$$

where the sample $s(n-i+1)$ of the input signal $s(t)$ is the desired response of module i.

After every module of PRNN finishes its prior prediction calculation, a series of error signals is obtained, namely, $e_1(n), e_2(n), \ldots, e_M(n)$.

- **Updating the Weight Matrix of Modules**

An overall cost function for the pipelined recurrent network of Figure 4.12 is defined by

$$\varepsilon(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{5.87}$$

By minimizing the cost function $\varepsilon(n)$, the gradient estimation algorithm is used to calculate the change $\Delta\mathbf{W}$ to the weight matrix $\mathbf{W}$ along the negative of the gradient $\nabla_{\mathbf{W}}\varepsilon(n)$.

The change applied to the kl^{th} element in the weight matrix can be written as

$$\Delta w_{kl} = -\eta \frac{\partial}{\partial w_{kl}} \left(\sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \right) \tag{5.88}$$

where η is a fixed learning-rate parameter, $1 \leq k \leq N$ and $1 \leq l \leq (p+N+1)$.

Finally, the weight matrix is updated as

$$\mathbf{W}_u \leftarrow \mathbf{W} + \Delta\mathbf{W} \tag{5.89}$$

where $\mathbf{W}_u$ is the updated weight matrix.

- **Filtering Process**

Using same input signal and the updated weight matrix $\mathbf{W}_u$, the filtering calculation of a module is the same as the prediction process. For example, the filtering process of Module i is defined by,

$$\mathbf{y}_i(n) = \Phi(\mathbf{W}_u, \mathbf{s}_i(n), \mathbf{r}_i(n)) \tag{5.90}$$

The vector $\mathbf{y}_i(n)$ is stored as the feedback input $\mathbf{r}_i(n+1)$ at the time point $n+1$, and

$$\mathbf{W} = \mathbf{W}_u \tag{5.91}$$

Sequentially, the filtering operation is executed from Module M to Module 1. The output $y_{1,1}(n)$ of Module 1 is sent to the tapped-delay-line filter as an overall filtering output $y_{filt}(n)$.

2. *Adaptive Operation of Linear Subsection*

- **Outputting the Final Estimate**

  The standard Least-Mean-Square algorithm is used to produce the optimum estimate of prediction $\hat{s}(n+1)$.

$$\hat{s}(n+1) = \mathbf{w}_l{}^T * \mathbf{y}_{filt}(n) \tag{5.92}$$

- **Updating the Weight of the Linear Filter**

  Input the new sample of signal $s(n+1)$ as the desired signal. The estimation error or residual $e(n+1)$ is defined as the difference between the desired response $s(n+1)$ and the estimated output, as shown by

$$e(n+1) = s(n+1) - \mathbf{w}_l{}^T * \mathbf{y}_{filt}(n) \tag{5.93}$$

  The updated weight vector of the tapped-delay-line filter is written as:

$$\mathbf{w}_l \leftarrow \mathbf{w}_l + \mu \mathbf{y}_{filt}(n) * e(n+1) \tag{5.94}$$

where μ is the step-size parameter.

3. *Recursive Calculation*

   Let $n = n + 1$ and return to Step 1. Repeat the on-line adaptive prediction until the input stops.

5.3 Adaptive Nonlinear Prediction of Speech Signals

In this section, the performance of the nonlinear continuous adaptive predictor developed in last sections is tested with real speech signals. The prediction of continuous speech data is a difficult task.

Criteria used to evaluate the nonlinear predictor are:

- The approximation of waveform of the predicted signal to that of the original signal.
- The value of *prediction gain*. Prediction gain is defined as

$$R_p = 10 * \log_{10}(\delta_s^2/\delta_p^2)$$

  where δ_s^2 is the mean-square value of a speech signal, and δ_p^2 is the mean-square prediction error of the corresponding signal.

- Spectrum of the prediction error. According to the theory of innovation process [53], the whiteness of the spectrum of prediction errors means that the (linear or nonlinear) predictor has removed most of the information contained in the original signal.
- Histogram of the prediction error.

We present highlights of the experimental work based on three speeches. They are records from several different male and female speakers. The audio files of these speech

Table: 5-1

Speech Signal	Sample Frequency	Sample Points	δ_s
Male Speech	8 kHz	10,000	0.3848
Female Speech	8 kHz	63,000	0.2521
Conversation	8 kHz	74,000	0.2525

signals, sampled at 8 kHz, are stored in the *Sun* station sparc 2. Table 5-1 gives the description of these audio files.

For processing these speech signals, the nonlinear predictor has the following parameters:

- Nonlinear Subsection:

  Number of modules, $M = 5$.

  Number of neurons per module, $N = 2$.

  Nonlinear prediction order, $p = 4$.

  Learning rate, $\eta = 0.0001$.

  Forgetting factor, $\lambda = 0.9$.

- Linear Subsection:

  Length of tapped-delay-line filter, $q = 12$.

  Learning rate, $\mu = 0.3$.

- A male speech: **when recording audio data....** This speech signal is originated from the reserved audio file of the *Sun* sparc workstation. The recorded time series corresponding to this speech signal, sampled at 8 kHz, is made up of 10,000 points.

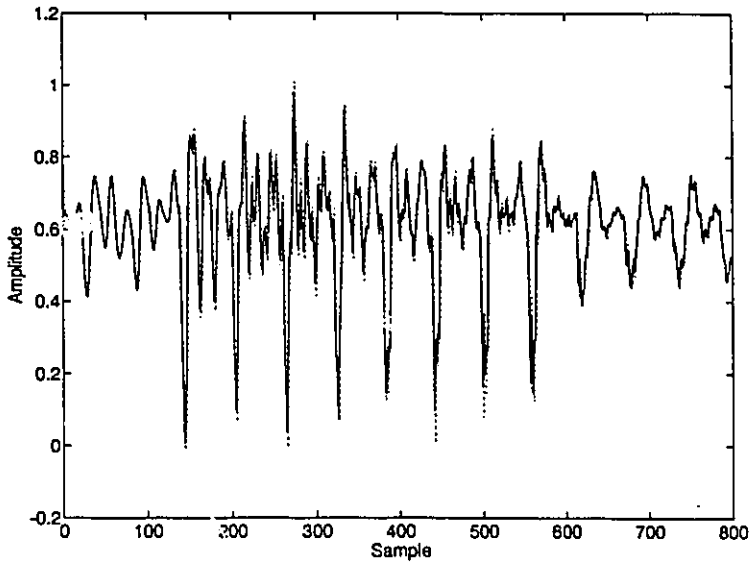


Figure 5.18: Adaptive nonlinear prediction of a speech signal. Continuous curve: actual speech signal; dashed curve: overall nonlinear prediction (including tapped-delay-line filtering). Specific parameters: number of modules $M = 5$, number of neurons/module $N = 2$, and number of taps in the linear subsection $q = 12$.

There is no any apparent noise accompanying the speech signal.

The experimental results are as follows:

1. Figure 5.18 displays the actual speech signal (solid curve) and the nonlinear prediction version (dashed curve). Figure 5.19 shows the linear prediction result of the same speech, which uses only a 12-tap *FIR* linear filter. The results described in Figure 5.18 and 5.19 clearly show that the prediction of a signal using the PRNN-based adaptive predictor provides a much better approximation to the actual speech signal than using the linear predictor. Figures 5.18 and 5.19 only show the prediction results of first 800 sample points (about 0.1 second) of this male speech.

2. The mean-square value δ_s^2 of 10,000 samples of the male speech is approximately 0.3848. The mean-square prediction error δ_p^2 is about $1.2 * 10^{-3}$ by using the PRNN-based nonlinear predictor, yielding $R_p = 25.06$dB; by using the linear

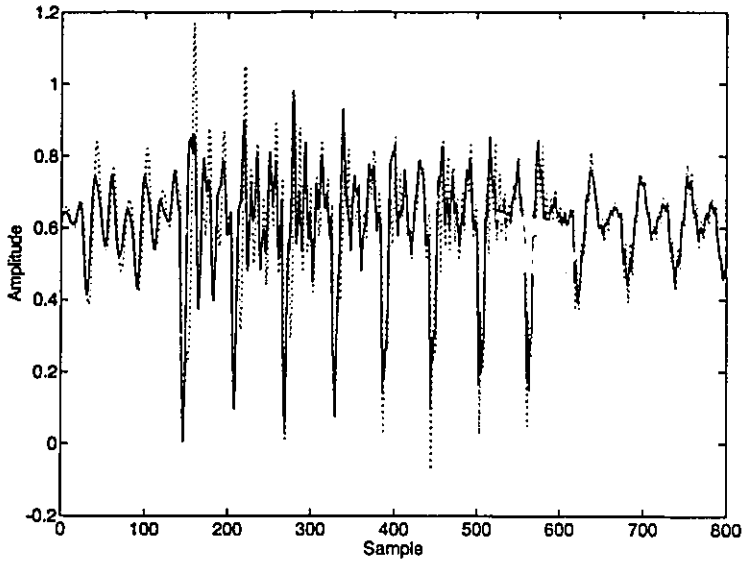


Figure 5.19: Adaptive linear prediction of a speech signal. Continuous curve: actual speech signal; dashed curve: overall linear prediction. Specific parameters: number of taps in the linear filter $q = 12$.

FIR filter, the corresponding mean-square linear prediction error δ_p^2 is about $2.5 * 10^{-3}$, yielding $R_p = 22.01$dB. This provides a quantitative demonstration of the superior performance of the nonlinear predictor over the linear predictor.

3. Figure 5.20 shows the power spectrum of the resulting nonlinear prediction error. The solid line presents the average power spectrum density on each frequency point. The distance of two dashed lines is the 95 percent confidence range. The power spectral density is fairly constant across a band from 200Hz to 3000 Hz.

4. Moreover, Figure 5.21 shows the histogram of the prediction errors. The x-axis presents the prediction error and y-axis presents number of samples. It is apparent that prediction errors of most samples are very small. According to the *Pearson system* theory [50], the distribution of nonlinear prediction errors is a p-distribution.

- A female speech: this is a record of a female telephone operator. The speech signal

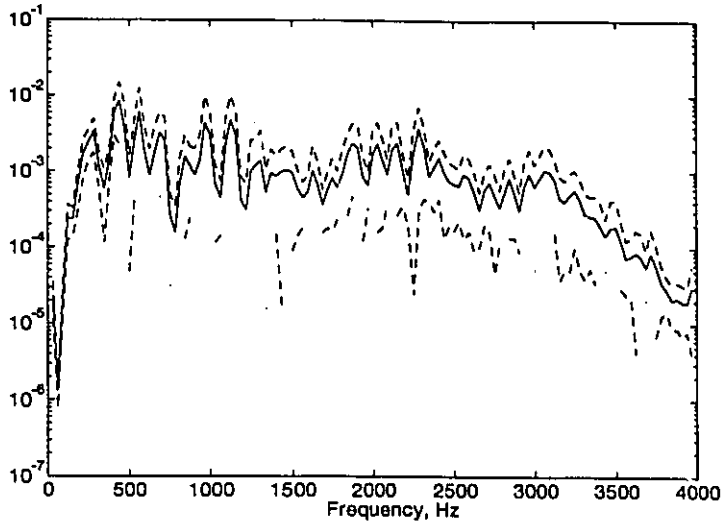


Figure 5.20: Power spectrum of the resulting overall nonlinear prediction error. Continuous curve: typical result; dashed curves: upper and lower bounds representing 95 percent confidence.

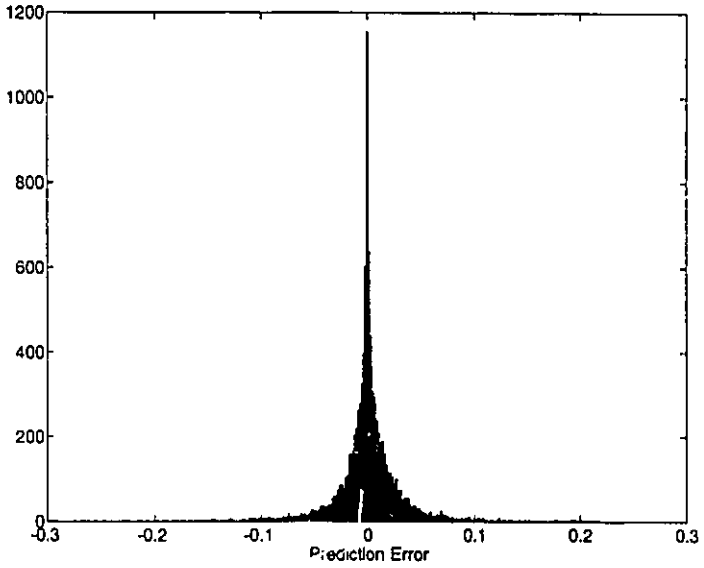


Figure 5.21: Histogram of the resulting overall nonlinear prediction error.

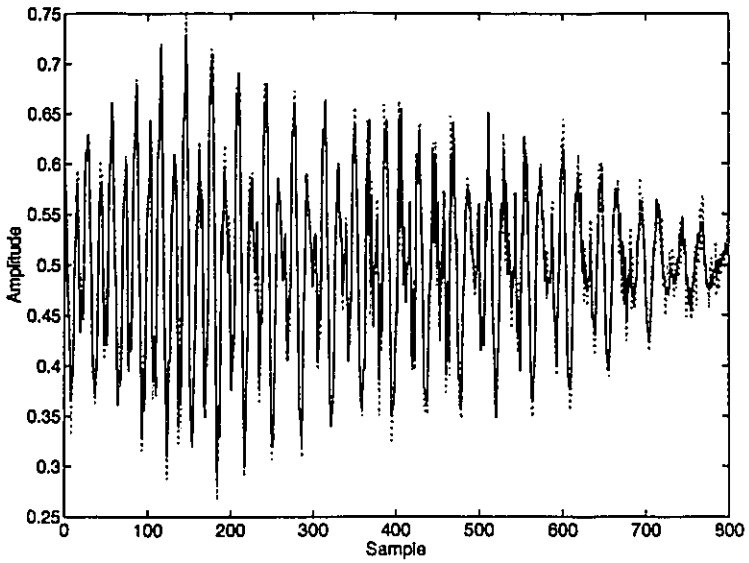


Figure 5.22: Adaptive nonlinear prediction of a female speech signal.

is provided by Bell Canada Company. The recorded time series corresponding to this speech signal consists of 63,800 points.

1. Figures 5.22 and 5.23 display the actual speech signal (solid curve) and those nonlinear and linear prediction versions (dashed curves). Compared with the results described in Figures 5.18 and 5.19, the results clearly shows that the prediction of a signal using the PRNN-based nonlinear on-line adaptive predictor provides a much better approximation to the actual speech signal than the linear predictor. In a manner similar to the figures of the previous male speech, Figures 5.22 and 5.23 only show the prediction results of first 800 sample points(about 0.1 second) of the female speech.

2. The mean-square value δ_s^2 of 63,800 speech samples is about 0.2521. By using the PRNN-based nonlinear predictor, the mean-square prediction error δ_p^2 is about $0.6 * 10^{-3}$, yielding $R_p = 26.2$dB; by using the linear *FIR* filter, the mean-square prediction error δ_p^2 is about $1.4 * 10^{-3}$, yielding $R_p = 22.55$dB. This provides a quantitative proof of the superior performance of the nonlinear predictor over

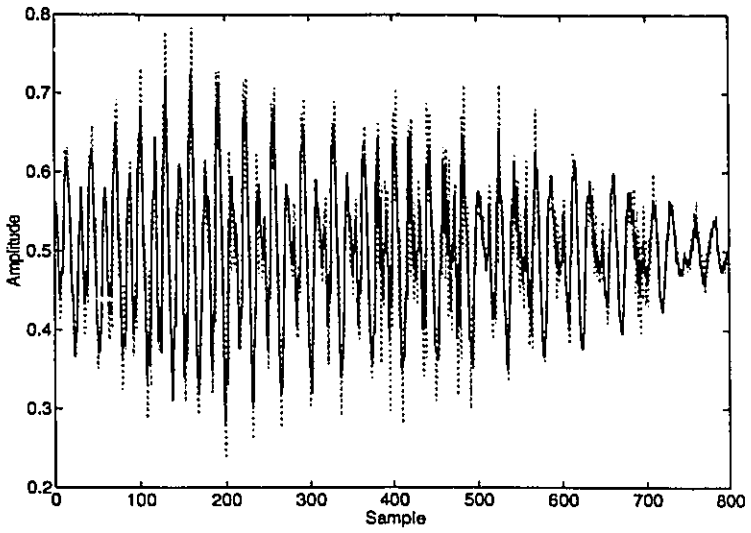


Figure 5.23: Adaptive linear prediction of a female speech signal.

that of the linear predictor.

3. Fig. 5.24 shows the power spectrum of the resulting nonlinear prediction errors. It is apparent that power spectral density is fairly constant on a wide frequency band from 500Hz to 3200 Hz.

4. Figure 5.25 shows the histogram of the nonlinear prediction errors. The distribution of the error is a β-distribution (the mean value of this distribution is near 0).

- Three-person (two males and one female) short conversation. The recorded time series corresponding to this speech signal, sampled at 8 kHz, consists of 74,000 points.

  1. By using the PRNN-based nonlinear predictor, the prediction gain R_p is 22.88dB; by using linear *FIR* filter, the prediction gain R_p is 16.69dB.

  2. Figure 5.26 shows the power spectrum of the resulting nonlinear prediction errors. The power spectral density is fairly constant across a band from 300Hz to 3600 Hz. Moreover, Figure 5.27 shows the histogram of the prediction errors,

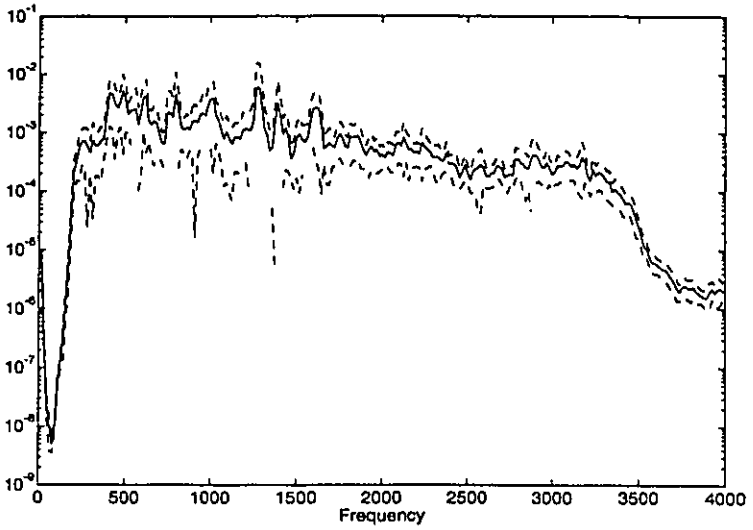


Figure 5.24: Power spectrum of the resulting overall nonlinear prediction error for a female speech.

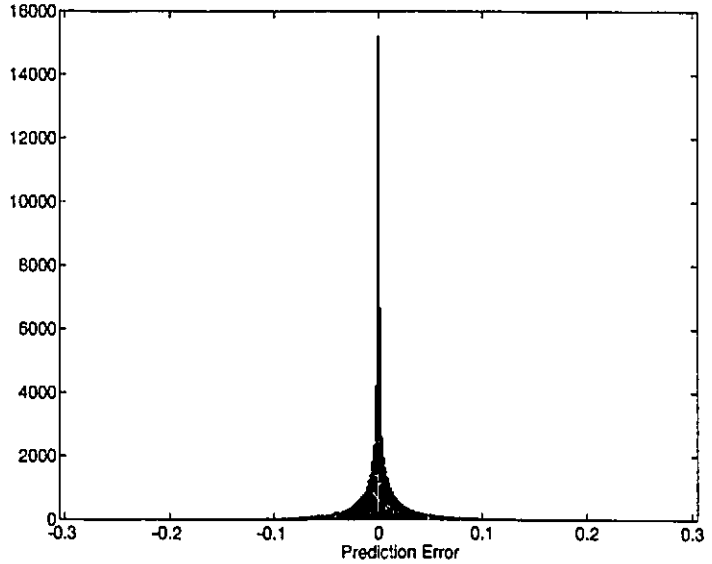


Figure 5.25: Histogram of the resulting overall nonlinear prediction error for a female speech.

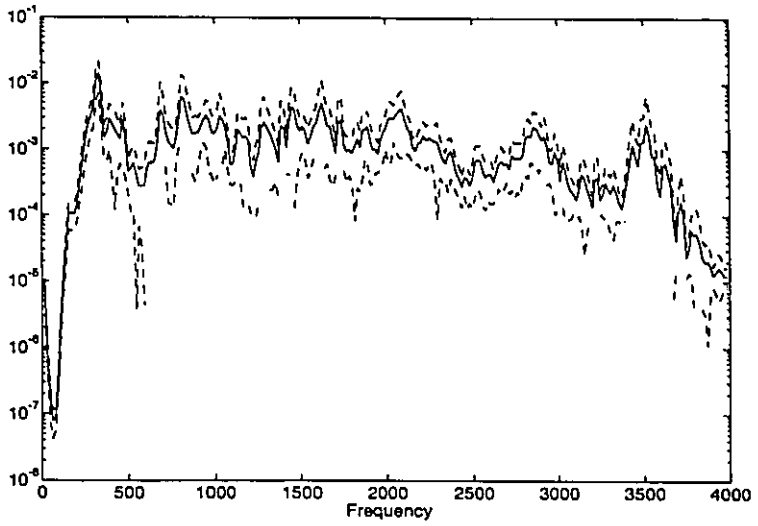


Figure 5.26: Power spectrum of the resulting overall nonlinear prediction error for a conversion signal.

which distribution is also a β-distribution.

The results of the nonlinear prediction of the three speech signals used as test signals clearly illustrate that:

- The results of Fig. 5.18 and Fig. 5.22 show that the prediction of speech signals by using of a PRNN-based nonlinear predictor provides a much better approximation to the actual speech signal than the linear predictor.

- The calculation of the prediction gain also gives a quantitative proof of the superior performance of the nonlinear prediction over that of the linear predictor. In general, the nonlinear predictor provides 3 to $5dB$ prediction gain.

- The histograms of three prediction errors show that the distribution of nonlinear prediction errors is a β-distribution. The nonlinear prediction errors of most samples are very small (near 0).

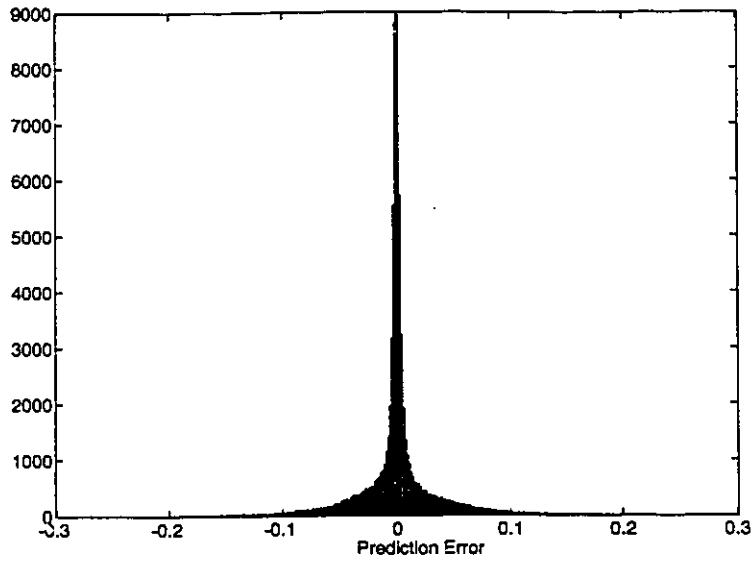


Figure 5.27: Histogram of the resulting overall nonlinear prediction error for a conversation signal.

- The power spectrum of nonlinear prediction error is almost constant for a wide frequency band ($300Hz$ to $3500Hz$). The conclusion to be drawn is that the nonlinear prediction error may be closely modeled as a white and β-distribution noise process. This is testimony to the efficiency of the neural network-based predictor in extracting the information content of the speech signal almost fully.

5.4 Selection of Parameters of a Nonlinear Predictor

- **Number of Modules**

  As mentioned in Chapter 4, the global nonlinear mapping performed by the pipelined recurrent neural network is coarse-to-fine, depending on the number of modules M uses. In principle, the more modules used, the more accurate the prediction obtained. Figure 5.28 shows the relation between the mean-square prediction error of a speech signal and the number of modules in the PRNN, assuming that $N = 2$, $p = 4$, and $q = 12$. For a sufficiently large number of modules, the decrease in error is almost

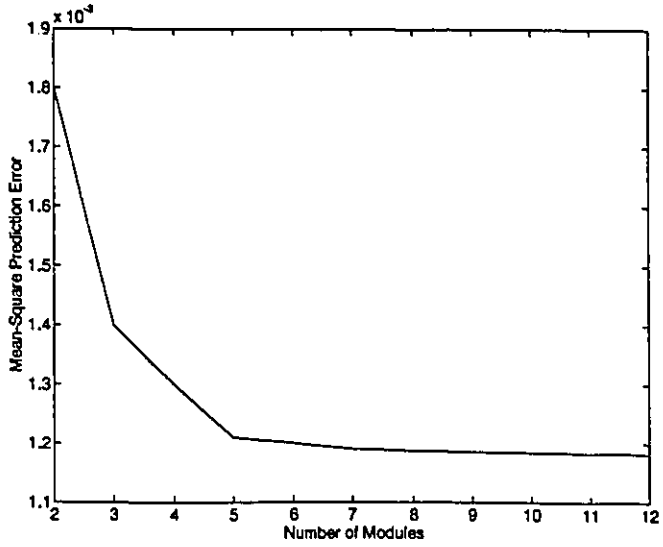


Figure 5.28: Illustration of the relation between the mean-square overall prediction error and the number of modules for the case of a male speech signal.

directly proportional to the number of modules. If M is larger than 6, the mean-square prediction error is almost constant.

- **Number of Neurons**

The linearization of the input space may also be improved by increasing the number of neurons in each module. Figure 5.29 illustrates the relation between the mean-square prediction error of a speech signal and the number of neurons N, assuming $M = 5$, $p = 4$ and $q = 12$. As we increase the number of neurons in each module, the overall mean-square prediction error is gradually reduced. However the computational requirement increases sharply with the addition of each new neuron.

- **Number of external inputs**

The number of the external inputs is also called the prediction order of PRNN-based

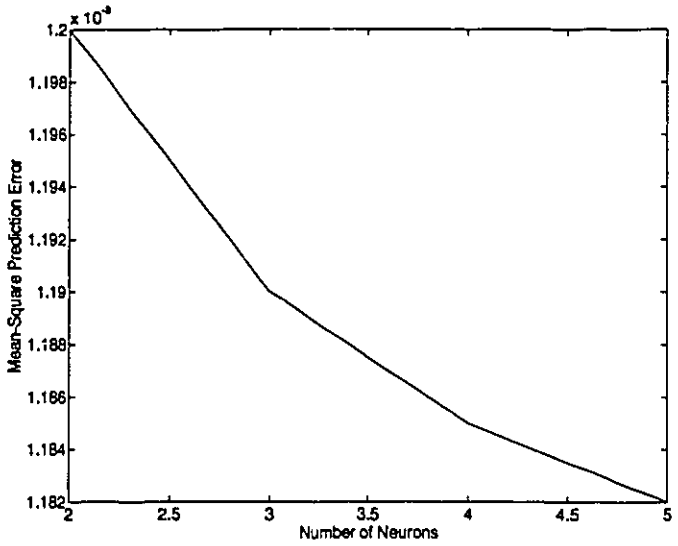


Figure 5.29: Illustration of the relation between the mean-square overall prediction error and the number of neurons in each module for the case of a male speech signal.

nonlinear adaptive predictor. According to the discussion of the approximation problem in Chapter 5, every module is a dynamical sub-model of the outside world:

$$y_1(n) = \phi(s(n-1), \cdots, s(n-p); y_1(n-1), \cdots, y_N(n-1)), \qquad n = p+1, p+2, \cdots \tag{5.95}$$

In general, the number p is a function of the correlation dimension, which could be used to describe the dynamical model of the outside world.

- **Number of Taps** Intuitively, we expect the accuracy of the one-step prediction performed by the linear subsection to be enhanced by increasing the number of taps q. This is borne out by the results shown in Figure 5.30. where the mean-square prediction error is plotted versus q for $M = 5$, $N = 2$ and $p = 4$. As the number of taps increase from 2 to 12, the prediction error decreases sharply, and once q is over 12, the error is only reduced of a small amount. The result is in keeping with choosing the orders of AR model for a speech signal[1]. The speech signal used in this section is

[1] At a sampling frequency is about 8 kHz, a 12-order linear AR model of a speech wave is considered to be adequate [55]

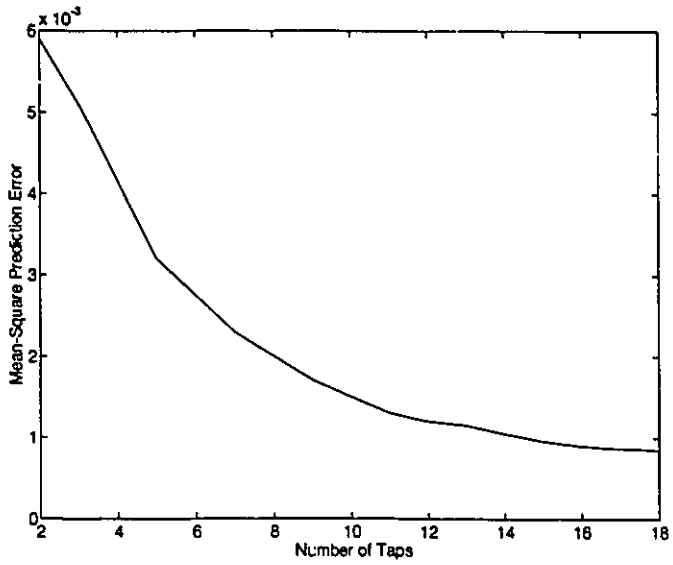


Figure 5.30: Illustration of the relation between the mean-square overall prediction error and the number of taps in the linear subsection for the case of a male speech signal.

the same as the male speech of Section 5.3

5.5 Discussion of Adaptive Nonlinear Prediction Algorithm

5.5.1 Initialization of Nonlinear Prediction Algorithm

To proceed with the on-line learning and filtering of the PRNN-based predictor, it is necessary to initialize the tap-weight vector $\mathbf{w}_l$ of the tapped-delay-line filter, and synaptic weight matrix $\mathbf{W}$ of every module in the pipelined neural network. Moreover, the initial feedback signal input vector $\mathbf{r}$ of every module should be specified.

Initialization of the weight vector $\mathbf{w}_l$ follows the customary practice of setting it equal to the null vector or a randomly distributed set of small values. However, initializations of the synaptic weight matrix $\mathbf{W}$ and the feedback signal vector $\mathbf{r}$ require special attention. For this initialization, we may use the traditional epochwise training method of recurrent neural networks, which is applied to one module operating with N_o samples of

input signal. The module is the same as that described in Section 4.3.

The training principle is summarized as follows:

1. Input first N_o samples of a signal, and obtain an input vector $\mathbf{s}(i)$ and a desired signal $d(i)$, when $1 \leq i \leq n'$ and $n' = N_o - p$.

$$\mathbf{s}(i) = [s(i + (p-1)), \cdots, s(i)]^T$$

$$d(i) = s(i+p)$$

2. The pre-training procedure begins from $i = 1$. Choose a random set of small values for weight matrix $\mathbf{W}$ and feedback input $\mathbf{r}(1)$.

3. Input the $\mathbf{s}(i), \mathbf{r}(i)$, and $\mathbf{d}(i)$ to a module. The following calculation is then executed:

$$\begin{aligned} y_k(i) &= \phi(v_k(i)) \\ v_k(i) &= \sum_{l=1}^{p} w_{kl} s(i + (p-j)) + w_{k(p+1)} + \sum_{l=p+2}^{p+N+1} w_{kl} r_{l-(p-1)}(i) \\ e(i) &= s(i+p) - y_1(i) \end{aligned}$$

4. Let $i = i + 1$,

$$\mathbf{r}(i) = \mathbf{y}(i-1)$$

and return to Step 3.

5. Repeat the calculation of Steps 3 and 4 until $i = n'$. If the average error $E(n')$ is less than a permitted error ϵ, the pre-training procedure is stopped; $E(n')$ is defined by

$$E(n') = \frac{1}{n'} \sum_{i=1}^{n'} e^2(i) \tag{5.96}$$

6. The $\Delta\mathbf{W}$ is obtained by using the gradient estimation algorithm along the negative of the gradient $\nabla_{\mathbf{W}} E(n')$, and then used to update weight matrix.

$$\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W} \tag{5.97}$$

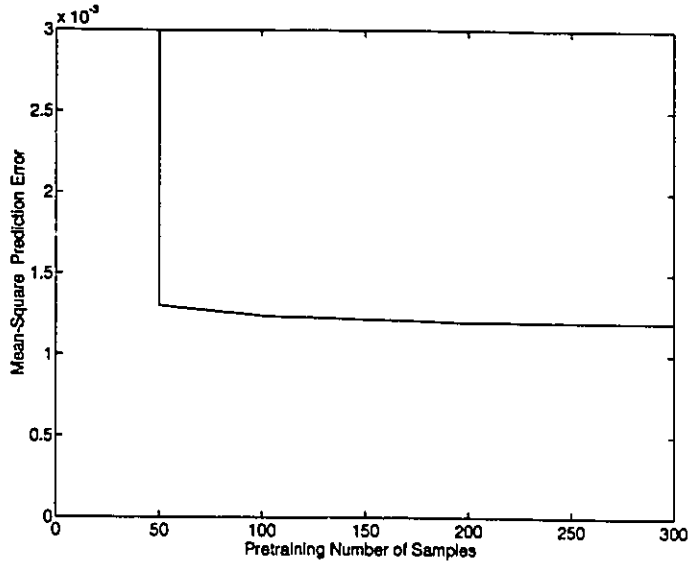


Figure 5.31: A plot of mean-square prediction error versus the number of pre-training samples.

7. Set i=1, and $\mathbf{r}(1) = \mathbf{y}(n')$, and return to Step 3.

It is important to choose the permitted error ϵ and adequate number of pre-training samples N_0. Roughly speaking, the permitted error ϵ is about one percent of the mean-value of the input time series.

In general, if the number N_0 is too small, the epochwise training method of the recurrent neural network could not determine the adequate initial weight matrix for adaptive on-line processing. An inadequate initial weight matrix may cause the PRNN-based nonlinear predictor to diverge.

The main purpose of pre-training is merely to determine an adequate set of initial weights. To economize on pre-training time, the number of pre-training samples N_0 is limited to about several hundreds.

Fig. 5.31 shows a plot of the mean-square prediction error of the male speech (about

10,000 samples) versus the number of pre-training samples by using the PRNN-based nonlinear predictor. When the number of pre-training samples is less than 50, the mean-square prediction error is huge and the prediction system tends to diverge. As the number is over 50, the mean-square prediction error decreases gradually. It is apparent that this decrease is slow!

5.5.2 Computational Requirement

The computational requirement of the continuous adaptive prediction process with the PRNN-based filter consists of computational requirements of both a nonlinear subsection (pipelined recurrent neural network) and a tapped-delay-line filter.

For the pipelined recurrent neural network, the computational requirement of a learning procedure is $O(MN^4)$ arithmetic operations, where N is the number of neurons in each module and M is the number of modules. Correspondingly, the computational requirement of the estimation procedure is $O(M * N^2)$. So, the total computational requirement of processing a signal sample in the nonlinear subsection is $O(MN^4 + M * (N^2))$ arithmetic operations.

Compared with computational requirements of the PRNN, that of a tapped-delay-line filter is much less. For example, the computational requirements of the LMS algorithm are only $2q+3$ multiplications for calculating one point and q additions per iteration, where q is the number of taps. So, the total computational requirement of processing a signal sample is $O(MN^4 + M * (N^2) + 3(q+1))$ arithmetic operations.

5.5.3 Convergence of Real-Time Nonlinear Prediction Algorithm

The traditional method of supervised learning is performed in an off-line manner. In the *off-line* case, the training procedure of a neural network (such as multilayer and rational bases function neural network) has two independent steps: supervised learning and testing. After the desired learning process is accomplished, the performance of the neural network is tested with another set of target responses. If the errors measure, such as the mean-square errors between the desired target responses and the corresponding outputs of the network, is lower than a determined threshold, the training procedure is stopped and the weights of the neural network are "frozen".

In contrast, the pipelined recurrent neural network is a dynamic system. The training produce of the PRNN is a continuous learning process. Every learning step is also a testing step of the previous learning processed. In fact, the squared error is obtained between the actual and predicted signal on every step.

A nonstationary signal is a time-varying signal. For instance, a speech signal includes different frequencies and amplitude sinusoids, which change with time. Here the squared error curve of a continuous learning or prediction process versus time is indeed fluctuating.

Thus, as a measure of converge performance we may use the tendency of average squared prediction errors on a series of sample blocks of a speech signal. Each block includes several thousand sampling points. If the adaptive prediction is convergent, the average squared error versus the number of blocks should gradually decrease, tending toward some constant value.

Fig. 5.32 shows a plot of the average squared prediction error of block of speech samples for a male speech, versus the number of blocks. The first sentence of the male speech is "**As recording audio signal...**". But the continual period of this male speech

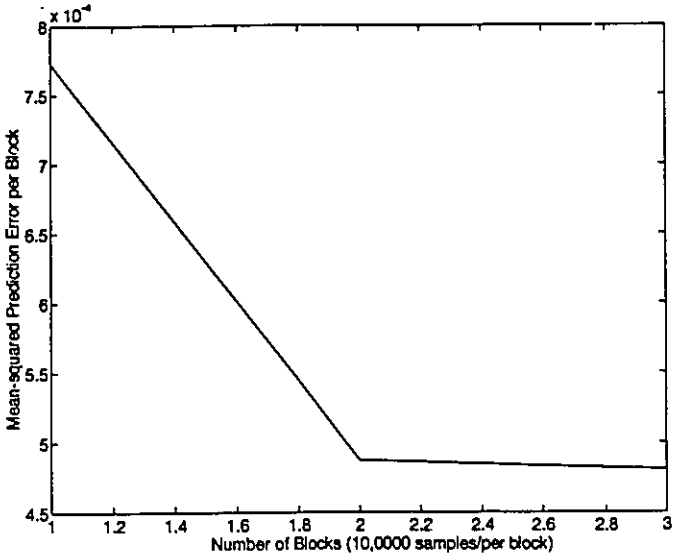


Figure 5.32: A plot of average squared errors of block of samples.

is about 40 seconds. So that, the corresponding time series of this speech consists of about 320,000 samples. Each block includes 100,000 continuous samples. Fig. 5.32 shows that the value of average squared prediction error decreases quickly as the number of blocks of samples is increased. The difference of average squared error between the first block of samples and the second one is larger than the difference between the second and third block, and so on. This shows that the nonlinear adaptive prediction is indeed convergent.

5.5.4 The Computation Ability of a Simplified PRNN-Based Predictor

Using the simplified pipelined recurrent neural network as the nonlinear subsection, we get a simplified PRNN-based adaptive predictor. The adaptive algorithm for the simplified predictor is almost the same as the algorithm of PRNN-based predictor described in the Section 5.2. The difference is that all of feedback signals of every module are originated from the outputs of the previous module except Module M.

$$\mathbf{r}_i(n) = \mathbf{y}_{i+1}(n), \qquad 1 \le i \le M-1$$

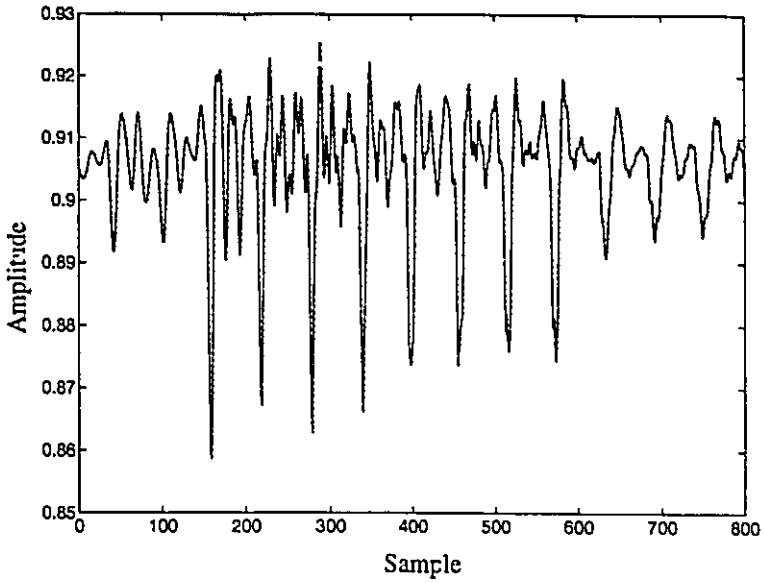


Figure 5.33: Comparison of the feedback inputs at the 2^{nd} node of module 1 obtain by the PRNN-based predictor (continuous curve), and the simplified PRNN-based predictor (dashed curve).

$$\mathbf{r}_i(n) = \mathbf{y}_i(n-1), \qquad i = M \tag{5.98}$$

In the simplified PRNN-based predictor, experiment results show that the feedback input $r_i(n)$ is:

$$\begin{aligned} \mathbf{r}_i(n) &= \mathbf{y}_{i+1}(n) \\ &\simeq [y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)]^T \end{aligned} \tag{5.99}$$

where $1 \leq i \leq M-1$. The vector $[y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)]^T$ is the feedback input vector of Module i in primary PRNN-based predictor.

The prediction of the male speech described in Section 5.3 is still used as an example. In the simplified PRNN-based predictor, there are 5 modules ($M = 5$) and 2 neurons ($N = 2$) in each module. Fig. 5.33 shows the continuous feedback signal applied to the 2^{nd} input node of Module 1 $r_{1,2}(n)$ by using the model of Fig. 4.12 (dashed line) and its counterpart computed by Fig. 4.14 (continuous line), both of them acting on the male

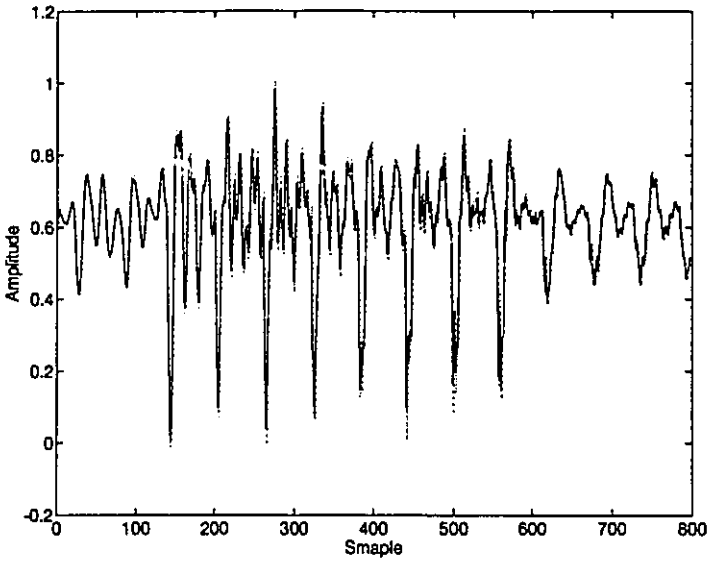


Figure 5.34: Adaptive nonlinear prediction of a male speech signal with the simplified model. Continuous curve: actual speech signal; Dashed curve: overall nonlinear prediction (including tapped-delay-line filtering). Specific parameters: number of modules, $M = 5$; number of neurons per module, $N = 2$; and number of taps in the linear subsection $q = 12$.

speech signal. An observation is the feedback input $r_{1,2}(n)$ of the primary PRNN-based prediction and that of simplified version are almost the same. It means that

$$y_{2,2}(n) \simeq y_{1,2}(n-1)$$

Otherwise, in two model, both of $r_{1,1}(n)$ are originated from the previous module.

$$r_{1,1}(n) = y_{2,1}(n)$$

So that the feedback input vector $\mathbf{r}_1(n)$ of Module 1 in two versions, are almost equal. The same results can be found in Modules 2, 3 and 4. Figure 5.34 displays the nonlinear prediction of the male speech signal with the simplified PRNN-based predictor. Compared with Fig. 5.18, it is apparent that the simplified model can obtain the same good prediction results.

The simplified predictor has the same computational ability as the Figure 4.12. In the case of speech signals consider in the previous section, the mean-square overall prediction

error of the simplified model is only 5% higher than that of the original model.

5.5.5 An Updated Improvement on the Algorithm

Before finishing the thesis, we made an updated improvement on the algorithm, which was described in Section 5.2. In the original algorithm, on the *filtering process* of PRNN, the updated weight matrix $\mathbf{W}$ and a set of input vectors $\mathbf{s}_1(n), \mathbf{s}_2(n), \ldots, \mathbf{s}_M(n)$ are used to calculate the filtering output $\mathbf{y}_{filt}(n)$. These input vectors are the same as those input vectors of the prediction process.

$$\mathbf{s}_i(n) = [s(n-i), s(n-(i+1)), \cdots, s(n-(i+p-1))]^T, \qquad i = 1, \ldots, M$$

and

$$\begin{aligned} y_{filt}(n) &= y_{1,1}(n) \\ &= \phi(\mathbf{w}_1, \mathbf{s}_1(n)\mathbf{r}_1(n)) \\ \mathbf{s}_1(n) &= [s(n-1), s(n-2), \cdots, s(n-p)]^T \end{aligned}$$

In fact, in the filtering process, the information representedly $s(n)$ is partly contained in the updated weight matrix $\mathbf{W}$.

In the improved algorithm, during the filtering process, the input vectors are also updated:

$$\mathbf{s}_i(n) = [s(n-i+1), s(n-(i+1)+1), \cdots, s(n-(i+p-1)+1)]^T, \qquad i = 1, \ldots, M \quad (5.100)$$

and

$$\begin{aligned} y_{filt}(n) &= y_{1,1}(n) && (5.101) \\ &= \phi(\mathbf{w}_1, \mathbf{s}_1(n)\mathbf{r}_1(n)) \\ \mathbf{s}_1(n) &= [s(n), s(n-1), \cdots, s(n-p+1)]^T && (5.102) \end{aligned}$$

In the filtering process of the improved algorithm, the input sample $s(n)$ is directly used in the filtering calculation. Therefore, the new filtering process uses the information of $s(n)$ more effectively.

The updated version introduces two improvement in the prediction calculation:

1. *Increasing Prediction Gain.* Because of more effectively using the information of the latest signal sample, the one-step nonlinear prediction scheme provides a slight increase in prediction gain. The prediction of the male speech described in Section 5.3 is still used as an example. With all of same parameters in the PRNN-based predictor, the mean-square prediction error of the updated version δ_p is $1.18 * 10^{-3}$, yielding $R_p = 25.134dB$. The prediction gain increases by $0.07dB$.

2. *Decreasing Computational Requirement.* The updated input vectors of the filtering process at the n^{th} time point, presented in the eq. 5.100, are also the input vectors of the prediction process at the $(n+1)^{th}$ time point. Therefore, the outputs of M modules at the filtering process can be used to calculate the cost function of PRNN directly at the next time point. In the updated algorithm, the filtering process of processing present sample is equal to the prediction process for next sample point. In the improved version, the computational requirement of an additional filtering proceeding is not necessary. Therefore, the computational requirement of the adaptive processing of PRNN is $O(MN^4)$; the total computational requirement of processing a signal sample is $O(MN^4 + 3(q+1))$ arithmetic operations.

It is possible that the new version would improve the quality of the nonlinear ADPCM, which is presented in Chapter 6.

5.6 Nonlinear Adaptive Prediction of Sea Clutter

The PRNN-based network has a wide range of applications. Here, an example of adaptive prediction of radar sea clutter is shown. In this example, we use the simplified nonlinear subsection in the PRNN-based adaptive predictor. The simplified predictor has the same computational ability as the original PRNN-based one. In the case of speech signals consider in the previous section, the mean-square overall prediction error of the simplified model is only 5% higher than that of the original model.

In 1989 and 1993, the radar group of the Communications Research Laboratory collected a larger data base of sea clutter by means of an instrument quality (IPIX) radar at a site on the East Coast of Canada. The solid line of Figure 5.35 presents 300 continual samples of a particular realization of sea clutter. To model the sea clutter, we use an adaptive PRNN-based nonlinear filter consisting of 10 modules ($M = 10$). For every module, the number of external inputs is chosen to be seven ($p = 7$), which is slightly greater than the correlation dimension of the sea clutter used in this study (estimated to be between 6.3 and 6.7). Three neurons are used in the upper layer ($N = 3$). The linear subsection is a seventh order FIR adaptive tapped-delay-line filter.

The adaptive prediction results of sea clutter, using the PRNN-based adaptive filter, are shown on Figure 5.35. The dashed curves present the nonlinear prediction version; and the solid curves present the actual sea clutter. The value of every point is normalized with respect to the maximum valve of the sea clutter. Figure 5.36 presents the corresponding prediction result for the sea clutter using a linear seventh order tapped-delay-line filter (using the LMS algorithm) under similar conditions. Comparison of Figure 5.35 and 5.36 clearly shows that the prediction performance of a nonlinear filter is superior to that of a linear filter for predicting sea clutter.

For comparison in quantitative terms, the *prediction gain* is used to evaluate the

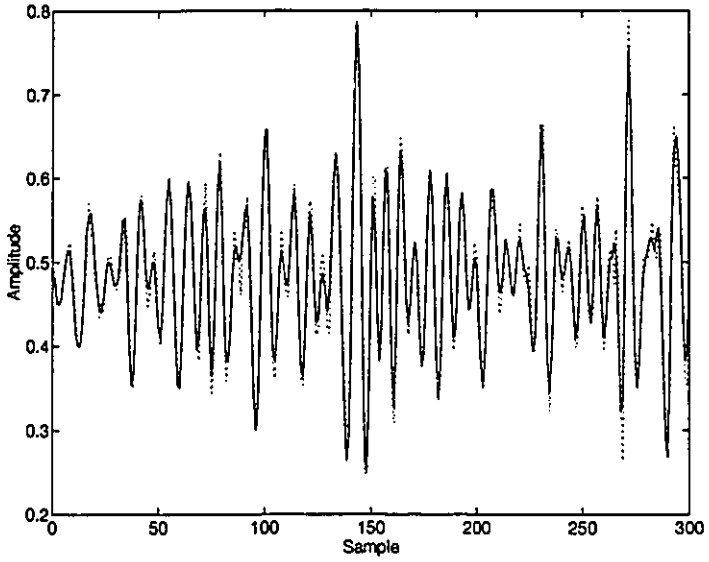


Figure 5.35: Nonlinear Adaptive prediction of sea clutter. Specific parameters: number of modules, $M = 10$; number of neurons per module, $N = 3$; and number of taps in the linear subsection, $q = 7$.

performance of prediction. For the 60,000 sea clutter samples, the mean-square value δ_s^2 of sea clutter samples is about 0.2387. The mean-square value of the prediction error δ_p^2 is $3.2 * 10^{-4}$ by using the PRNN-based nonlinear prediction, and is $9.1 * 10^{-4}$ by using the linear filter with the LMS algorithm. The corresponding *prediction gain* R_p of the former is about 28.75 dB, and the latter is about $24.2dB$. These results show an improvement of over $4dB$ in prediction gain in favor of the nonlinear prediction approach.

5.7 Summary

In this chapter, we have described a novel pipelined recurrent neural network (PRNN) which, together with a linear subsection in the form of a tapped-delay-line (TDL) filter, provides a powerful device for the nonlinear adaptive filtering of nonstationary time series. The adaptive predictor is called PRNN-based predictor. The coefficients of this predictor

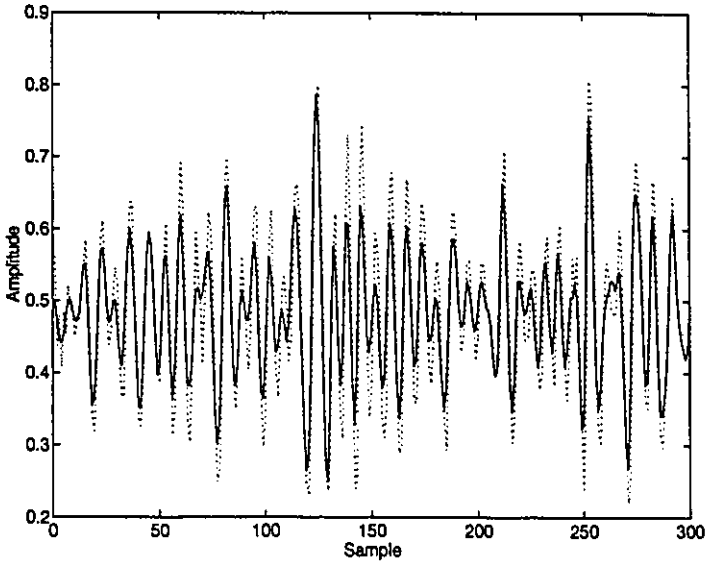


Figure 5.36: Linear adaptive prediction of sea clutter.

are determined experimentally, using real-life data.

The algorithm of the adaptive nonlinear prediction is carefully studied. In fact, if the required computational time of this PRNN nonlinear prediction on processing each sample is smaller than the sample period of a continuous signal, the on-line nonlinear prediction can be called *real-time nonlinear adaptive prediction* of this signal. However, the length of requirement computational time is related to the software of the adaptive algorithm, and the corresponding hardware.

The adaptive signal processing capability of this predictor has been demonstrated by studying the continuous one-step prediction of various speech signals. Wu(1993) proposed that a recurrent neural network (including 1 output neuron and 8 hidden neurons) could be used for the prediction of the speech [51]. Wu did not present whether his prediction was used in a continuous fashion. However, the on-line (continuous) prediction of speech signals is a more challenging job. In this chapter, the adaptive nonlinear prediction results show that the PRNN-based predictor could be used in the continuous processing of speeches.

In this chapter, we have used an additional example, the adaptive prediction of a sea clutter, to demonstrate the wide applicability of the PRNN-based adaptive predictor. Moreover, results present the nonlinear adaptive prediction outperforms the traditional linear adaptive scheme in a significant way.

Chapter 6

A Nonlinear Adaptive Differential Pulse-Code Modulation System

In the previous chapters, a neural network-based nonlinear adaptive predictor was developed. Its dynamical behavior is demonstrated for the case of nonstationary signals, such as speech signals. Herein, as the computational time of such a predictor satisfies the requirement of processing a speech signal in a real-time communication system, the PRNN-based predictor could be used in the real-time communications system.

The conventional form of Adaptive Differential Pulse-Code Modulation (ADPCM) relies on the use of a linear predictor, yet, it is widely recognized that the biological mechanism responsible for the generation of speech signals is in fact nonlinear. In this chapter, a nonlinear Adaptive Differential Pulse-Code Modulation is introduced. The PRNN-based nonlinear adaptive predictor is proposed for the adaptive prediction of speech signals in an encoder and a decoder. Correspondingly, a 4-, 8- or 16-level adaptive quantizer is designed for the nonlinear ADPCM. Finally, the algorithm of this nonlinear ADPCM is derived.

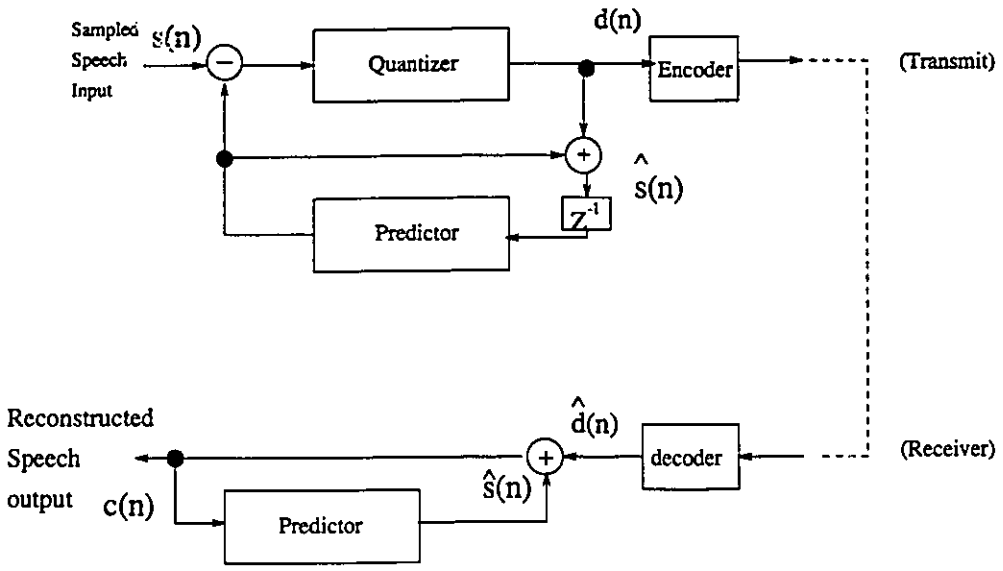


Figure 6.37: A predictive residual PCM system (generalized differential PCM).

6.1 Adaptive Predictive Coding System of Speech Signals

A standard method of digitizing speech signals for transmission over a communications channel involves the use of pulse-code modulation (PCM), which operates at the standard rate of 64 kb/s (sampling frequency being 8,000Hz and representing each sample with 8 bits). This high bit rate demands a higher channel bandwidth for its implementation. However, in certain applications (e.g. secure communication over radio channels that are of low capacity), channel bandwidth is at a premium. In applications of this kind, there is a definite need for speech coding at low bit rates, while maintaining an acceptable fidelity or quality of reproduction. One of the low rate waveform coding methods is *Differential Pulse-Code Modulation* (DPCM).

A differential pulse-code modulation system is depicted in Fig. 6.37, which involves the use of a predictor. The predictor is designed to exploit the correlation that exists between adjacent samples of the speech signal, in order to realize a reduction in the number of bits required for the transmission of each sample of the speech signal and yet maintain a

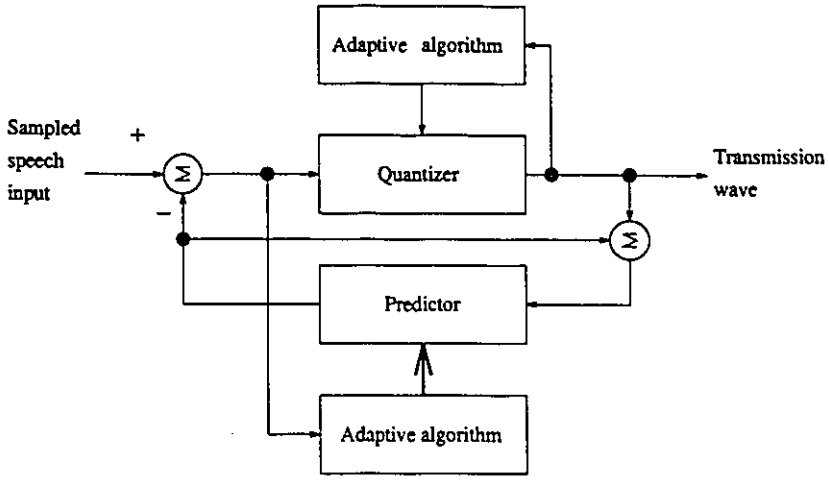


Figure 6.38: Encoder of an ADPCM system.

prescribed quality of performance [61]. This is achieved by quantizing and then coding the prediction error that results from the subtraction of the predictor output from the input signal. If the prediction is optimized, the variance of the prediction error will be significantly smaller than that of the original signal. Equivalently, for a quantizing error of prescribed variance, DPCM requires a smaller number of quantizing levels (and therefore a smaller bit rate) than PCM.

The reconstructed speech signal in Fig. 6.37 is

$$c(n) = \hat{s}(n) + \hat{d}(n) = \hat{s}(n) + [d(n) + e(n)] \tag{6.103}$$

$$= \hat{s}(n) + [(s(n) - \hat{s}(n) + e(n)] = s(n) + e(n) \tag{6.104}$$

where $s(n)$ is the original speech signal, $d(n)$ is the prediction error and $e(n)$ is the quantization noise and transmission error at time point n [57].

A further reduction the transmission rate can be achieved by using an adaptive quantizer together with an adaptive predictor of a sufficiently high order. This type of waveform coding is called *Adaptive Differential Pulse-Code Modulation* (ADPCM), where the adaptive predictor is used in order to account for the nonstationary nature of speech signals. The encoder of ADPCM is shown in Fig.6.38.

In June 1983, International Telephone and Telegraph Consultative Committee (CCITT) chartered a small seven-expert group to recommend a 32 kb/s ADPCM coding algorithm as a candidate for international standardization. They proposed several key constraints that should be observed in designing such an ADPCM algorithm:

- For simplicity, the algorithm should not rely on side information, for example, parameter transmission.

- For minimal transmission delay, all adaptation processes should be backward-acting in time.

- Particular attention should be paid to encoder/decoder mistracking recovery in the presence of quantization and transmission errors, for example, bit errors and slips.

The group embarked on an eighteen month work program that resulted in an ADPCM algorithm defined as a digital transcoding technique from 64 kb/s PCM. CCITT formally approved the algorithm as an international standard in October 1984 [58, 60].

The traditional form of ADPCM uses a linear adaptive predictor in encoder and the decoder [36, 57]. As mentioned in Chapter 2, the production of a speech signal is known to be the result of a dynamic process that exhibits two distinct characteristics: nonlinearity and nonstationary. The conventional form of ADPCM takes into account of the nonstationary but not the nonlinearity. It should not therefore to surprising to find that the use of a nonlinear predictor in the construction of an ADPCM may provide a significant improvement in its performance.

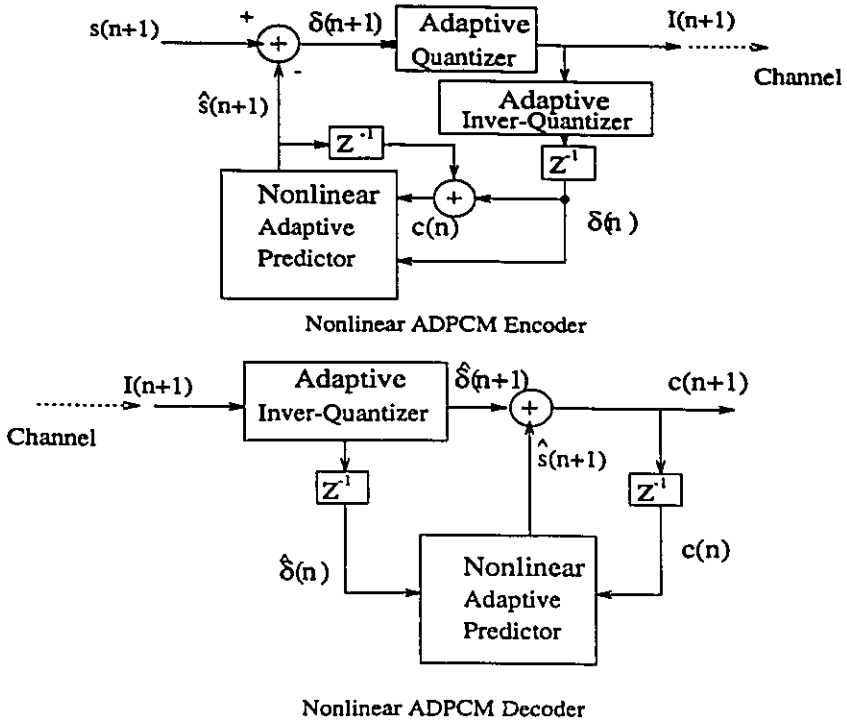


Figure 6.39: A nonlinear ADPCM system.

6.2 The Nonlinear Adaptive Differential Pulse-Code Modulation

In this section, we address this issue by investigating the use of the pipelined recurrent neural network as the basis for the design of a nonlinear predictor of ADPCM system [56]. The new system is called *Nonlinear Adaptive Differential Pulse-Code Modulation.* We suppose that the time of computational requirement of this system is still lower than the time needed for real-time speech communication. Thus, the nonlinear predictor is called *a real-time adaptive predictor.*

A block diagram illustrating the principle of the nonlinear ADPCM is shown in Figure 6.39, which consists of an encoder and a decoder separated by a communication channel. The encoder, located at the transmitter, uses a feedback scheme that includes

an adaptive quantizer and a PRNN-based adaptive predictor in the feedback path. The nonlinear predictor, acting on reconstructed signals $c(n)$, forms an one-step estimated value $\hat{s}(n+1)$. A prediction error $\delta(n+1)$, defined as the difference between the input signal $s(n+1)$ and one-step prediction $\hat{s}(n+1)$, is in turn quantized and encoded by using an adaptive quantizer. The encoded version $I(n+1)$ constitutes the transmitted wave. At the same time, $I(n+1)$ is also fed to the inverse quantizer. The inverse quantized signal $\hat{\delta}(n)$ and the reconstructed signal $\hat{c}(n)$, which is the sum of $\hat{s}(n)$ and $\hat{\delta}(n)$, are operated upon by a PRNN-based adaptive predictor.

The decoder, located at the receiver, employs an exact replica of the PRNN-based adaptive predictor used in the encoder. The transmitted signal $I(n+1)$ is decoded, and added to the estimated $\hat{s}(n+1)$. The sum is the reconstructed speech signal $c(n+1)$.

In this nonlinear ADPCM system, a 4-, 8- or 16-level adaptive quantizer is adopted to quantize the difference signal. One sample of the corresponding quantized different signal is coded with two, three of four binary digits. So, the output $I(n+1)$ forms the 16, 24, or 32kb/s transmitted signal.

6.3 Adaptive Prediction of Speech Signal in Nonlinear ADPCM

6.3.1 Adaptive Predictor of the Nonlinear ADPCM

The real-time nonlinear adaptive predictor of the nonlinear ADPCM system originates from the adaptive PRNN-based predictor. However, the choice of a suitable neural network predictor architecture for such an application is constrained by the fact that the predictor has to perform its learning in an **on-line** fashion as the speech signal is being continuously processed, and that the stability of the encoder and decoder of ADPCM should be assured.

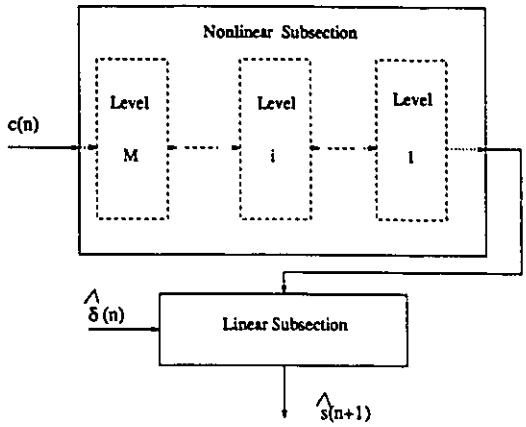


Figure 6.40: A nonlinear adaptive predictor.

In a manner similar to the PRNN-based nonlinear adaptive predictor described in Chapter 4, the predictor of the nonlinear ADPCM system consists of two parts: a nonlinear subsection and a linear subsection, as shown in Figure 6.40. The model of the nonlinear subsection of this nonlinear adaptive predictor is a pipelined recurrent neural network. A detailed structure of the nonlinear subsection is shown in Figure 6.41, involving M neural processing levels. Each level has a module and a comparator of its own. Every module is a dynamical sub-model, consisting of a recurrent neural network with N neurons.

From the Module $M-1$ to 1, the N feedback signals consist of their delayed output and the output of the previous module. The last module in the pipeline operates as a standard real-time recurrent neural network. The output vector $\mathbf{y}_M(n)$ of this module is fed back to its input as an input vector after a delay of one time unit. Thus, the M modules operate similarly in that they all have the same number of external inputs and feedback nodes. Moreover, they are designed to have exactly the same weight matrix. Let $\mathbf{W}$ denote the synaptic weight matrix for each module$((p+N+1)-by-N)$; and be written as:

$$\mathbf{W} = \{w_{kl}\}, \qquad k = 1,\ldots,N; \qquad l = 1,\ldots,p+N+1$$

The linear subsection uses an *infinite impulse response filter.* It is similar to the linear predictor of *CCITT* 32 kb/s linear ADPCM system [58]. The linear subsection, depicted

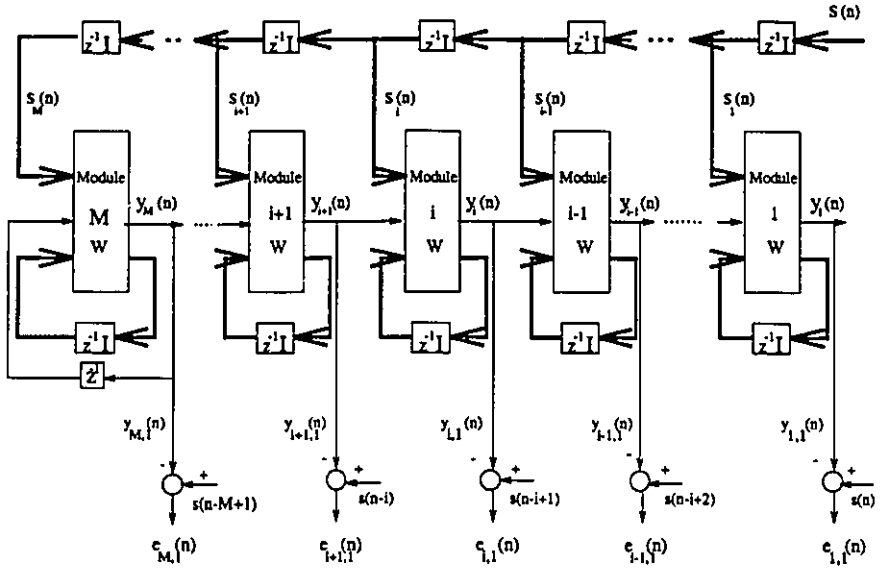


Figure 6.41: The nonlinear subsection of an ADPCM predictor: pipelined neural network.

in Fig. 6.42, is composed of two structures, a six-order section that models zero points in the input signal and a second-order section that models pole points in the input signal. The final choice of this structure results from the investigation of several factors: performance for narrowband and broadband speech signals, stability, encoder/decoder tracking with transmission errors and computational efficiency.

It is important to note that any side information (e.g., the weights of the neural network) of the encoder cannot be transmitted to the decoder. So, the real-time learning

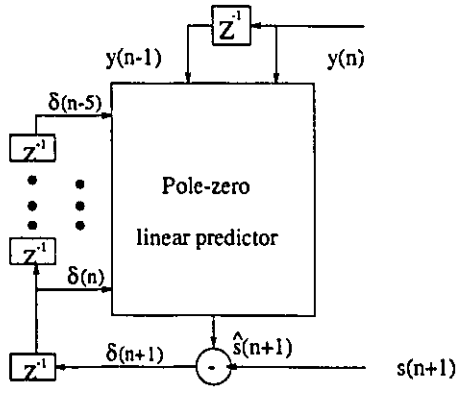


Figure 6.42: The linear subsection of a PRNN-based predictor.

of the nonlinear predictor of the decoder can only use the information contained in the quantized differential signal $I(n+1)$. The mistracking with transmission errors may cause the decoder system to become unstable. Especially, the stability of the on-line training and processing procedure of pipelined recurrent neural networks is sensitive to the input and calculation error.

6.3.2 The Calculation Procedure of the Nonlinear Section

The input of the nonlinear subsection is the reconstructed speech signal $c(n)$. At the n^{th} time point, the external input of Module i is a $p-by-1$ vector

$$\mathbf{c}_i(n) = [c(n-i), c(n-(i+1)), \cdots, c(n-(i+p-1))]^T \tag{6.105}$$

where $1 \leq i \leq M$.

The other input is a $N-by-1$ "feedback vector"

$$\mathbf{r}_i(n) = [r_{i,1}(n), \cdots, r_{i,N}(n)]^T, \qquad i = 1, \ldots, M \tag{6.106}$$

At the n^{th} time point, the output vector of Module i is defined by:

$$\mathbf{y}_i(n) = [y_{i,1}(n), \cdots, y_{i,N}(n)]^T \tag{6.107}$$

$$y_{i,k}(n) = \phi(\sum_{l=1}^{p} w_{kl} c(n-(i+l-1)) + w_{k(p+1)} + \sum_{l=p+2}^{p+N+1} w_{kl} r_{1,l-p-1}(n)) \tag{6.108}$$

where ϕ presents the activation function of a neuron, which is presented in Eq. 4.51.

The feedback vector of Module i $(1 \leq i \leq M-1)$ is described as:

$$\mathbf{r}_i(n) = [y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)]^T; \tag{6.109}$$

The feedback vector of Module M is defined as

$$\mathbf{r}_M(n) = [y_{M,1}(n-1), \cdots, y_{M,N}(n-1)]^T \tag{6.110}$$

As mentioned previously, the outputs of the first neuron in M modules are fed directly to individual comparators, which compute the error signals defined by

$$e_i(n) = c(n-i+1) - y_{i,1}(n); \tag{6.111}$$

where the reconstructed signal samples $c(n-i+1)$ $(i = 1, \ldots, M)$ are the desired responses from Module 1 to M at the n^{th} time point, respectively. It is important to note that the desired response needed to perform the training of the nonlinear subsection is in fact derived from the incoming time series often reconstructed speech signal. Thus, the nonlinear adaptive predictor described here embodies a self-organized, in-situ learning process.

The output of visible (first) neuron of Module 1 is the overall output of the pipelined recurrent network $y(n)$ as shown by

$$y(n) = y_{1,1}(n) \tag{6.112}$$

On the filtering process, the output of the pipelined recurrent network $y(n)$ is expressed as $y_{filt}(n)$, and sent to the linear subsection.

6.3.3 Mathematical Model and Calculation Procedure of the Linear Subsection

The pole-zero structure of the linear subsection originated from the IIR filter studied by *Nishitani* and *Cointot* [65, 66]. Nishitani proved that such an IIR filter is more effective than a structure employing only adaptive zeros when complexity is considered.

Figure 6.42 describes the input-output relation of the linear subsection. At the n^{th} time point, the input of this linear subsection (filter) consists of the output vector of the nonlinear subsection $\mathbf{y}_{filt}(n)$ and the inverse-quantized transmitted signal vector $\hat{\Delta}(n)$. The vector $\mathbf{y}_{filt}(n)$ contains present output $y_{filt}(n)$ of the pipelined recurrent network and

one-time delay past value $y_{filt}(n-1)$:

$$\mathbf{y}_{filt}(n) = [y_{filt}(n), y_{filt}(n-1)]^T \tag{6.113}$$

and the inversely quantized transmitted signal vector $\hat{\Delta}(n)$ is

$$\hat{\Delta}(n) = [\hat{\delta}(n), \hat{\delta}(n-1), \cdots, \hat{\delta}(n-5)]^T$$

The output of this linear filter is defined by

$$\hat{s}(n+1) = \mathbf{a}^T(n)\mathbf{y}_{filt}(n) + \mathbf{b}^T(n)\hat{\Delta}(n) \tag{6.114}$$

where the coefficient vector $\mathbf{a}$ is a $2-by-1$ vector, and $\mathbf{b}$ is a $6-by-1$ vector.

$$\mathbf{a}(n) = [a_1(n), a_2(n)]^T \tag{6.115}$$

$$\mathbf{b}(n) = [b_1(n), \cdots, b_6(n)]^T \tag{6.116}$$

For minimizing the mistracking behavior of the nonlinear predictor, a novel coefficient update algorithm (a simplified $sgn().sgn()$ gradient algorithm) is used to update both vectors of linear subsection coefficient $\mathbf{a}$ and $\mathbf{b}$ [65].

$$a_1(n) = (1-2^{-4})a_1(n-1) + 3*2^{-8}sgn(p(n))sgn(p(n-1)) \tag{6.117}$$

$$a_2(n) = (1-2^{-7})a_2(n-1) + 2^{-7}(sgn(p(n))sgn(p(n-2)) - f(a_1(n-1))sgn(p(n))sgn(p(n-1))) \tag{6.118}$$

where

$$p(n) = \hat{\delta}(n) + \mathbf{b}(n-1)\hat{\Delta}(n-1) \tag{6.119}$$

$$f(a_1(n-1)) = \begin{cases} 4a_1(n-1), & |a_1(n-1)| \le 2^{-1} \\ 2sgn(a_1(n-1)), & |a_1(n-1)| > 2^{-1} \end{cases} \tag{6.120}$$

$$sgn(t) = \begin{cases} 1, & t > 0 \\ 0, & t = 0 \\ -1, & t < 0 \end{cases} \tag{6.121}$$

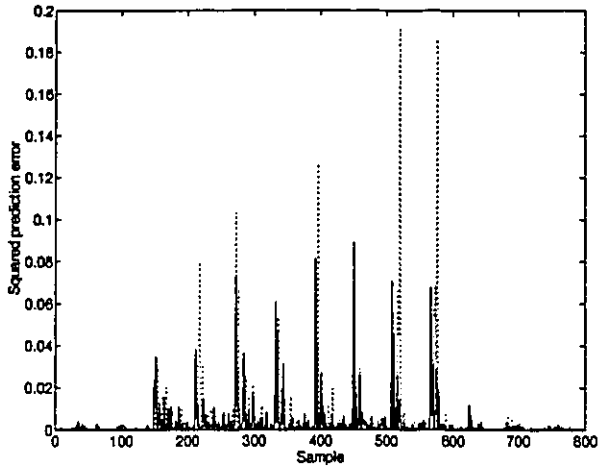


Figure 6.43: Squared prediction errors using the linear and nonlinear adaptive predictions.

with the stability constraints on $a_1(n)$ and $a_2(n)$:

$$|a_2(n)| \leq 0.75 \tag{6.122}$$

$$|a_1(n)| \leq 1 - 2^{-4} - a_2(n) \tag{6.123}$$

The i^{th} element of vector $\mathbf{b}$ is computed in accordance with the relation:

$$b_i(n) = (1 - 2^{-4})b_i(n-i) + 2^{-7}sgn(\delta(n))sqn(\delta(n-i)) \tag{6.124}$$

where $i = 1, \ldots, 6$

The dynamical processing ability of PRNN-based predictor (consisting of PRNN and the pole-zero linear filter) is demonstrated for the case of real-time prediction of speech signal. The speech signal is the male speech described in Section 5.3, (**when recording audio data...**). The recorded time series corresponding to this speech signal, sampled at 8 kHz, is made up of 10,000 points.

Fig. 6.43 shows the squared prediction errors of nonlinear and linear methods. The

solid curve pertains to the squared prediction error of the speech signal produced by using the PRNN-based predictor. The nonlinear predictor has the following parameters:

- Number of modules, $M = 4$.
- Number of neurons per module, $N = 2$.
- Number of external inputs, $p = 4$.

The dashed curve pertains to the squared prediction error produced by using only the pole-zero linear predictor on its own. The results described in Fig. 6.43 clearly show that the one-step prediction of a speech signal using a nonlinear real-time adaptive predictor provides a much better approximation to the actual speech signal than the linear predictor.

For 10,000 speech samples, by using the PRNN-based nonlinear predictor, R_p is about $25.9dB$; by using the linear *FIR* filter, R_p is about $23.01dB$.

6.4 Adaptive Quantizer

A 4-, 8-, or 16-level non-uniform adaptive quantizer is used to quantize the different signal $\delta(n+1)$. Fig. 6.44 shows details of the adaptive quantizer. In the adaptive quantizer, *adaptation speech control*, and *scale adaptation* are important parts, which adjust the input of the quantizer in real time. The outputs $I_0(n+1), I_1(n+1), I_2(n+1), I_3(n+1)$ or $I_4(n+1)$ are binary digits. I_0 expresses a sign bit. I_1, I_2, I_3 express the the absolute transmission value.

In order to maintain a wide dynamic range and minimize complexity, prior to quantization, $\delta(n+1)$ is converted to a base 2 logarithmic representation. In order to take advantage of the distortion masking ability of the human ear and a constant signal-to-distortion ratio over a range of input signal levels, the logarithmic conversion is necessary

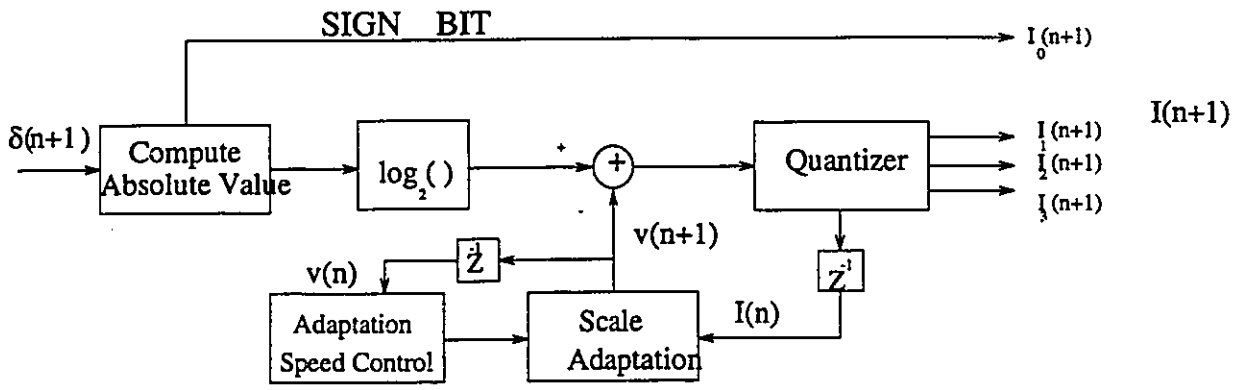


Figure 6.44: The detail of an adaptive quantizer.

[59].

The difference between the $log_2|\delta(n+1)|$ and the scale factor $v(n+1)$ is input to the quantizer. The $v(n+1)$ is an adaptive scale factor, which is updated in the scale adaptation block continuously. The basic principle used in updating the scale factor $v(n+1)$ is *bimodal adaptation*, which is described as follows:

- It is desirable for $v(n+1)$ to change quickly for different signals with large fluctuations (e.g. arising from speech).
- It is desirable for $v(n+1)$ to change slowly for different signals with small fluctuation (e.g. arising from tones, voiceband data) [69].

The fast and slow control of change is managed by the *adaptation Speed Control.* It outputs a fast factor $v_u(n)$ and a slow factor $v_l(n)$. Bimodal adaptation is realized by forming a linear combination of fast and slow adaptive scale factors.

The fast scale factor $v_u(n)$ is recursively computed from the resultant scale value of $v(n)$:

$$v_u(n) = (1 - 2^{-5})v(n) + 2^{-5}W[I(n)] \tag{6.125}$$

where $v_u(n)$ is limited by $1.06 \leq v_u(n) \leq 10$. The leak factor $(1 - 2^{-5})$ introduces finite

Table: 6-1-1

Normalized quantizer input range $\log_2\|\delta(n+1)\|$-v(n+1)	I	Normalized quantizer output $\log_2\|\delta(n+1)\|$-v(n+1)
(- ∞ , -6.0)	0	-8.5
[-6.0, - ∞)	1	-5.5

Table: 6-1-2

I	0	1
W[I]	1.76	28.02

Table: 6-2-1

Normalized quantizer input range $\log_2\|\delta(n+1)\|$-v(n+1)	I	Normalized quantizer output $\log_2\|\delta(n+1)\|$-v(n+1)
(- ∞,-8.0)	0	-9.0
[-8.0 ,-6.0)	1	-7.0
[-6.0 ,-4.0)	2	-5.0
[-4.0 , ∞)	3	-3.0

Table: 6-2-2

I	0	1	2	3
W[I]	-0.25	1.88	8.56	36.38

memory to aid encoder/decoder tracking recovery following transmission errors. For 2-bit, 3-bit and 4-bit (including 1 sign bit) operation, the corresponding discrete function $W[I]$ is defined in Table 6-1-2, Table 6-2-2, and Table 6-3-2, respectively.

A slow scale factor is then derived from $v_u(n)$ with a delayed slow scale factor:

$$v_l(n) = (1 - 2^{-6})v_l(n-1) + 2^{-6}v_u(n) \tag{6.126}$$

In the *scale adaptation*, the fast and slow scale factors are combined to form the factor $v(n+1)$:

$$v(n+1) = \bar{a}_l(n+1)v_u(n) + (1 - \bar{a}_l(n+1))v_l(n) \tag{6.127}$$

where $\bar{a}_l(n+1)$ is a controlling parameter.

The control parameter $\bar{a}_l(n+1)$, produced in the adaptation speed control block, can assume a continuous range of values between 1 and 0. It tends toward unity for speech signals

Table: 6-3-1

Normalized quantizer input range $\log_2\|\delta(n+1)\|$-v(n+1)	I	Normalized quantizer output $\log_2\|\delta(n+1)\|$-v(n+1)
(- ∞, -9.0)	0	-10.0
[-9.0 , -8.0)	1	- 8.5
[-8.0 , -7.0)	2	-7.5
[-7.0 , -6.0)	3	-6.5
[-6.0 , -5.0)	4	-5.5
[-5.0 , -4.0)	5	-4.5
[-4.0 , -3.0)	6	-3.5
[-3.0 , - ∞)	7	-2.5

Table: 6-3-2

I	0	1	2	3	4	5	6	7
W[I]	-0.75	0.25	1.69	3.13	6.13	11.5	21.25	69.25

and toward zero for voiceband data signals and tones. Thus the quantizer speech control transition between fast and slow modes is a smooth process, avoiding limited cycles arising from mistracking between scale factors in the encoder and decoder following a transmission error.

Finally, $\bar{a}_l(n)$ is obtained from a symmetrical limiting of $\bar{a}_p(n)$ to eliminate premature transitions for pulsed input signals (e.g. switched carrier voiceband data):

$$\bar{a}_l(n+1) = \begin{cases} 1, & \bar{a}_p(n) > 1 \\ \bar{a}_p(n), & \bar{a}_p(n) \leq 1 \end{cases} \tag{6.128}$$

The intermediate variable $\bar{a}_p(n)$ is defined by:

$$\bar{a}_p(n) = \begin{cases} (1-2^{-4})\bar{a}_p(n-1) + 2^{-3}, & v(n) < 3 \\ (1-2^{-4})\bar{a}_p(n-1), & \text{otherwise} \end{cases} \tag{6.129}$$

As the different signal between the $log_2|\delta(n+1)|$ and the scale factor $v(n+1)$ is sent to the quantizer, the quantized output decimal value I is obtained from Table 6-1-1, Table

6-2-1 and Table 6-3-1. The quantizer assigns 1, 2, and 3 binary digits to the value I, and forms the binary output $[I_1(n+1)]$, $[I_1(n+1), I_2(n+1)]$, or $[I_1(n+1), I_2(n+1), I_3(n+1)]$.

The inverse quantizer computes an inversely quantized version $\hat{\delta}(n+1)$ of the different signal using the transmitted bits $I(n+1)$, the scale factor $v(n+1)$ and Tables 6-1-1, 6-2-1 and 6-3-1.

6.5 Main Procedure of Nonlinear Adaptive Differential Pulse-Code Modulation Algorithm

1. **Real-time Prediction:**

   - *Training:* At the instant n^{th} time point, the output vectors and error signals of M modules are defined by, respectively,

$$\hat{\mathbf{y}}_i(n) = \Phi(\mathbf{W}, \mathbf{c}_i(n), \mathbf{r}_i(n)) \tag{6.130}$$

$$e_i(n) = c(n-i+1) - \hat{y}_{i,1}(n), \qquad i = 1, \ldots, M \tag{6.131}$$

   where Φ denotes a vector-valued function, $\hat{\mathbf{y}}_i(n)$ presents the prediction output vector of the i^{th} module during the training process, and $e_i(n)$ is the corresponding error signal. An overall cost function for the pipelined recurrent network of Fig. 6.41 is defined by

$$\varepsilon(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{6.132}$$

   where λ is an exponential forgetting factor that lies in the range $(0 < \lambda \leq 1)$. The cost function $\varepsilon(n)$ is minimized by using the gradient estimation algorithm of PRNN, described in Chapter 4, to calculate the change $\Delta\mathbf{W}$ along the negative

of the gradient $\nabla_{\mathbf{W}}\varepsilon(n)$, Finally, the weight matrix is updated as

$$\mathbf{W}_u \leftarrow \mathbf{W} + \Delta\mathbf{W} \tag{6.133}$$

where $\mathbf{W}_u$ is an updated weight matrix.

- *Outputting:* The filtering action of M modules is defined by,

$$\mathbf{y}_i(n) = \Phi(\mathbf{W}_u, \mathbf{c}_i(n), \mathbf{r}_i(n)), \qquad i = 1, \ldots, M \tag{6.134}$$

The vector $\mathbf{y}_1(n)$ is stored as the feedback input $\mathbf{r}_1$ at time point $n+1$, $y_{filt}(n) = y_{1,1}(n)$ and $\mathbf{W} = \mathbf{W}_u$.

- *Adaptive operation of the linear subsection:*

The output $y_{1,1}(n)$ of nonlinear subsection and the different signal $\hat{\delta}(n)$ are input to the linear subsection. The output of the total nonlinear predictor is:

$$\hat{s}(n+1) = \mathbf{a}^T\mathbf{y}_{filt}(n) + \mathbf{b}^T\hat{\Delta}(n) \tag{6.135}$$

where

$$\begin{aligned} \mathbf{y}(n) &= [y_{filt}(n), y_{filt}(n-1)]^T \\ \hat{\Delta}(n) &= [\hat{\delta}(n), \cdots, \hat{\delta}(n-5)]^T \end{aligned}$$

The weight vectors $\mathbf{a}(n)$ and $\mathbf{b}(n)$ are updated by using the adaptive algorithm, which is described in section 6.3.

2. **Real-Time Quantization and Coding:**

At the $(n+1)^{th}$ time point, the new speech sample $s(n+1)$ is input to the quantizer. The different signal $\delta(n+1)$ is obtained and then adaptively quantized, which is discussed in Section 6.4. The quantized output $I(n+1)$ is encoded with binary digits. With the $8,000Hz$ sample frequency, the 2-bit $[I_o, I_1]$, 3-bit $[I_0, I_1, I_2]$ or 4-bit $[I_0, I_1, I_2, I_3]$ output codes form the 16. 24 or 32 kbit/s output signal for transmission

over the communication channel. At same time, the $I(n+1)$ is sent to the adaptive inverse quantizer. The estimated difference $\hat{\delta}(n+1)$ is delayed and sent to the linear subsection of the adaptive nonlinear predictor.

3. **Real-Time Decoding:**

   At the decoder, the received signal is sent to the adaptive inverse-quantizer of the decoder. The delayed different signal $\hat{\delta}(n)$ and reconstructed $c(n)$ are fed to the nonlinear adaptive predictor, which operates similarly to that in the encoder. The nonlinear predictor has to perform on-line learning with the information of the different signal $\hat{\delta}(n)$, and produce the $\hat{s}(n+1)$. Finally, the reconstructed speech signal $c(n+1)$ is obtained by summing the predicted signal $\hat{s}(n+1)$ and inverse-quantized signal $\hat{\delta}(n+1)$.

4. **Recursive Calculation:**

   Let $n = n+1$, and return to Step 1.

6.6 Summary

In this chapter, a nonlinear Adaptive Differential Pulse-Code Modulation algorithm is developed. The PRNN-based predictor is employed as the nonlinear predictor for adaptive differential pulse-code modulation of speech signals. It may overcome the shortcomings of the conventional ADPCM, which uses the linear predictor. In keeping with the nonlinear predictor, system (including the 4-, 8- and 16-level adaptive quantizer) is redesigned.

Before the real-time coding procedure may begin, there is a pre-training procedure for the nonlinear predictor. The pre-training determines the initial weight matrix of the nonlinear predictor. The procedure is fairly short in that it needs the initial use of 100-200 samples of the speech signal. The pre-training procedure is performed only once during a

continuous speech signal (including the different voice and silence periods).

In next chapter, the performance of nonlinear Adaptive Differential Pulse-Code Modulation algorithm is evaluated, and compared to the standard linear form of the system.

Chapter 7

Quantitative Evaluations of the Nonlinear ADPCM Algorithm

In the previous chapter, the nonlinear Adaptive Differential Pulse-Code Modulation algorithm is proposed. The nonlinear pipelined recurrent neural network-based predictor is used in the ADPCM, and the operating part of the system is redesigned. In this chapter, the nonlinear ADPCM algorithm is evaluated with a series of subjective tests. The subjective tests include the calculation and analysis in the time- and frequency- domains, and listening tests. Also, this nonlinear ADPCM is compared with the 32 kb/s linear ADPCM algorithm, designed by *AT&T* Bell Labs.

A clear statement about processing delay of the nonlinear ADPCM is presented in this chapter. Finally, the sensitivity of the nonlinear ADPCM to noise in the input speech is examined, including the use of several subjective tests.

7.1 Experimental Equipment and Speech Signals

7.1.1 Audio Experimental Equipment and Software

For testing the nonlinear adaptive differential pulse coding modulation algorithm, a set of audio experimental equipment is built. The hardware equipment includes a *SUN* workstation (Sparc 2), a two-deck cassette tape recorder (*Luxman*), an earphone and a microphone.

The *Luxman* tape recorder is directly connected to the *SUN* workstation. It sends the original speech signal to the computer, and receives the corresponding reconstructed speech signal from the computer. The input Sensitivity/Impedance of the tape recorder is 100mv/47kΩ; the Output Level/Impedance is about 500mv/2.4kΩ. With a normal tape, its effective frequency response band extends from 20Hz to 16kHz.

The Sparc workstation is the center of this audio experiment system. It has several auxiliary audio parts, such as a speaker and a input/output audio signal jacket.

Three kinds of audio softwares ("Soundtool", "Audiotool" and "Xwave") are used to support the ADPCM experimental system. Using these softwares, samples of an audio signal are translated to differently encoded data files which can be read by the ADPCM algorithm, and vice versa. An audio signal could be translated to a decimal-code data file using "Audiotool" software, or to a binary-code data file using "Xwave" software. A decimal-code data file could be translated to a $\mu-$law compressed audio file using the "Soundtool" software.

To simplify the software of the PRNN-based nonlinear ADPCM algorithm, the input and output files are the normalized decimal-code data files. So, the input $\mu-law$ compressed audio files or binary data file may be converted to the linear decimal code file by using "Sountool" and "Matlab" software.

7.1.2 Speech Signals

The nonlinear ADPCM algorithm has been tested with manifold speech signals. In this thesis, the highlights of three speech signals are presented:

- *A male speech*, beginning with: *When recording an audio signal, ...* The speech file continues for about 40 seconds. With 8,000Hz sampling frequency, the corresponding time series of the speech file consists of 320,000 samples. The quality of the speech is high, and there are little noises in it.

- *A female speech*, being a recording of a telephone operator. A time series corresponding to the speech file consists of 63,800 samples.

- *Four-person (two females and two males) conversation*, being one of standard audio files for testing speech coding systems at the *AT&T* Bell laboratory. The conversation continues for 7 seconds. There are about 52,000 samples in the time series of this speech. The conversation is a high quality audio file, and there is no apparent noise in it.

7.2 Nonlinear ADPCM Quantitative Evaluation

7.2.1 Index of Evaluation

In evaluating the performance of a predictive coding system, it is most important to examine the quality of a reconstructed speech and compare the reconstructed speech with the original one. The following indexes of evaluation are presented:

- *Waveform Approximation.* It evaluates the approximation of the waveform of a reconstructed speech signal to that of an original one in the time domain.

Table 7-1: A subjective measure for listening quality

Score	Quality	Impairment
4	Excellent	Imperceptible
3	Good	Just perceptible but not annoying
2	Fair	Slightly annoying
1	Poor	Annoying but not too objectionable
0	Bad	Very annoying

- *Segmental Signal-to Noise Ratio Track.* The segmental Signal-to-Noise Ratio (SNR) or Signal-to-Distortion Ratio (SDR) [59] track curve evaluates the quality of a reconstructed continuous speech signal. Each segment consists of 80 continuous samples of a reconstructed signal. At segment i, the Signal-to-Noise Ratio(SNR) is defined as

$$SNR(i) = \frac{\sum_{j=1}^{80} s(80 \times (i-1)+j)^2}{\sum_{j=1}^{80} (s(80 \times (i-1)+j) - c(80 \times (i-1)+j))^2} \tag{7.136}$$

  where $\{s\}$ is the time series of an original speech signal, and $\{c\}$ is that of a corresponding reconstructed speech signal. In general, a SNR track consists of $600 \sim 800$ segments. According to the recommendation of G.712 (CCITT,1988), the average segmental Signal-to-Noise (distortion) should be higher than $25dB$.

- *Coherence Function.* This function evaluates the correlation between an original speech and a corresponding reconstructed one in the frequency domain. At one frequency point, the value of a coherence function displays the relation between the original and reconstructed speech signals for a specific frequency.

- *Listening Test.* The user's opinion about the listening quality of a speech communication system is a crucial parameter which a designer must take into account. According to the paper of Kitawaki and Nagabuch (1988) [71], the listening quality of a reconstructed speech is awarded a score from 0 to 4, as given in Table 7-1.

7.2.2 Subjective Tests of Nonlinear ADPCM Algorithms with a Male Speech

1. Figures 7.45, 7.46 and 7.47 describe the male speech waveforms reconstructed by the 32 kb/s, 24 kb/s and 16 kb/s nonlinear ADPCM algorithm. The continuous curves pertain to the original male speech signal depicted in Section 7.1, and the dashed curves pertain to the reconstructed versions measured at the output of a decoder. In these plots, only the first 800 samples of the original and reconstructed speech signals are drawn. Table 7-2 summarizes the mean-square values of the error between the original and reconstructed speech (320,000 samples) using three bit-rate nonlinear ADPCM algorithms. The results presented herein clearly show that:

   (a) The speech signal reconstructed by a nonlinear ADPCM algorithm may provide a good approximation to the original speech.

   (b) In making a comparison among these three figures, it is apparent that the 32 kb/s nonlinear ADPCM algorithm provides the most accurate approximation.

2. Fig. 7.48, Fig. 7.49 and Fig. 7.50 show the segmental SNR track curves of these speech signals reconstructed by the 32 kb/s, 24 kb/s and 16 kb/s nonlinear ADPCM algorithms respectively. Fig. 7.48 is the segmental SNR track curve of the reconstructed male speech using the 32 kb/s nonlinear ADPCM; and the average segmental SNR of 800 segments is $46.1dB$. The average values of segmental SNR, corresponding to the 24kb/s and 16 kb/s ADPCM nonlinear algorithms, are $39.6dB$ and $33.8dB$ respectively. All three average segmental SNR (even that obtained from the 16 kb/s nonlinear ADPCM) are higher than $25dB$.

3. The coherence functions between the original male speech and reconstructed ones are shown in Fig. 7.51, Fig. 7.52 and Fig. 7.53. From these figures, we may make the following observations:

(a) The value of the coherence function between the original and reconstructed speech, using nonlinear 32 kb/s ADPCM, is higher than or equal to 0.9 on a wide frequency band (100Hz–1,600Hz), and is higher than 0.7 between 1,500Hz to 3,400Hz. This means that the reconstructed speech preserves the information of the original speech signal on the main frequency band, and partially preserves the information on the higher frequency band.

(b) In using the lower bit nonlinear ADPCM algorithm, the frequency bandwidth, on which the value of a coherence function is higher than 0.9, is shortened. For the 24 kb/s algorithm, the corresponding frequency band stretches from 100Hz to 1,000Hz; for 16 kb/s algorithm, the corresponding frequency band extends from 100Hz to 700Hz. However, on wider frequency bands (100-2,600Hz, for the 24 kb/s algorithm; 100Hz-1,500Hz for the 16 kb/s one), the values of two coherence functions are still higher than 0.5. These results demonstrate that the output of a lower bit-rate nonlinear ADPCM may adequately represent the original speech. The difference between the original and reconstructed signals is increased as the bit-rate is decreased.

4. Some colleagues in our research laboratory were invited to listen to the male speech signals reconstructed by the nonlinear ADPCM algorithms. They gave the following evaluations:

   There is little difference between the original speech and the speech reconstructed by using the 32 kb/s nonlinear ADPCM algorithm (the estimated quality score of the reconstructed speech $\geq$ 3.5). There are slight noises in the speech reproduced by the 24 kb/s nonlinear ADPCM (its estimated quality score $\geq$ 3.0). In using the nonlinear 16 kb/s ADPCM system, some annoying noise is heard in its reconstructed speech (the estimated quality score $\geq$ 2.5).

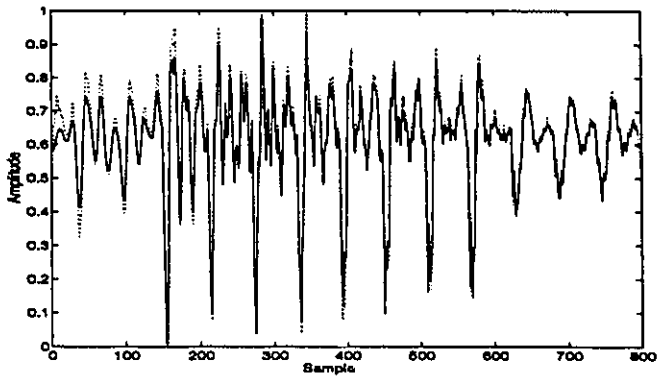


Figure 7.45: Reconstruction of a male speech using the 32 kb/s nonlinear ADPCM algorithm. Continuous curve: original speech signal; dashed curve: reconstructed speech signal.

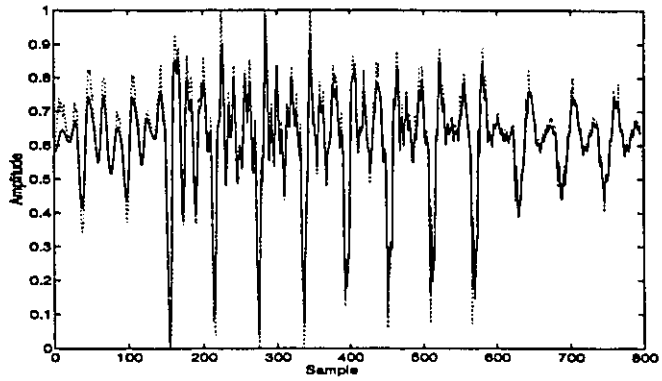


Figure 7.46: Reconstruction of a male speech using the 24 kb/s nonlinear ADPCM algorithm.

7.2.3 Subjective Tests of Nonlinear ADPCM Algorithms with a Female Speech

1. The second test speech is the female speech depicted in Section 7.1. Figs. 7.54, 7.55 and 7.56 show, respectively, the waveforms (800 samples) of the original speech and these speech signals reconstructed by 32 kb/s, 24 kb/s and 16 kb/s nonlinear ADPCM algorithms. The dashed curves pertain to the reconstructed speech signals obtained at the decoder. The continuous curves pertain to the original female speech signal. Table 7-3 shows the values of mean-square error between the original speech (60,000 samples) and the corresponding reconstructed one by three nonlinear ADPCM

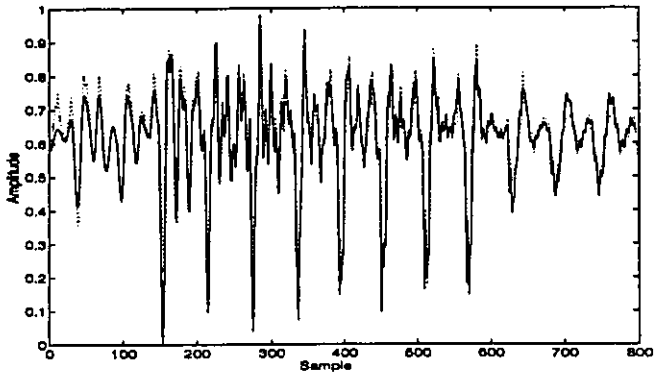


Figure 7.47: Reconstruction of a male speech using the 16 kb/s nonlinear ADPCM algorithm.

Table 7-2

	32 kb/s Nonlinear ADPCM	24 kb/s Nonlinear ADPCM	16 kb/s Nonlinear ADPCM
Mean-square Error	$1.3*10^{-4}$	$3.8*10^{-4}$	$8.7*10^{-4}$

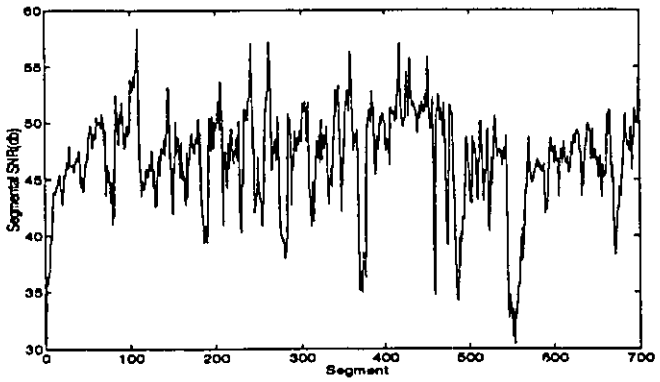


Figure 7.48: The segmental SNR track of the male speech reconstructed by the 32 kb/s nonlinear ADPCM algorithm.

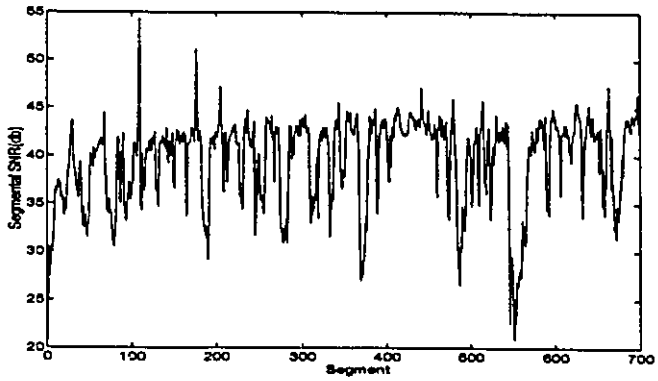


Figure 7.49: The segmental SNR track of the male speech reconstructed by the 24 kb/s nonlinear ADPCM algorithm.

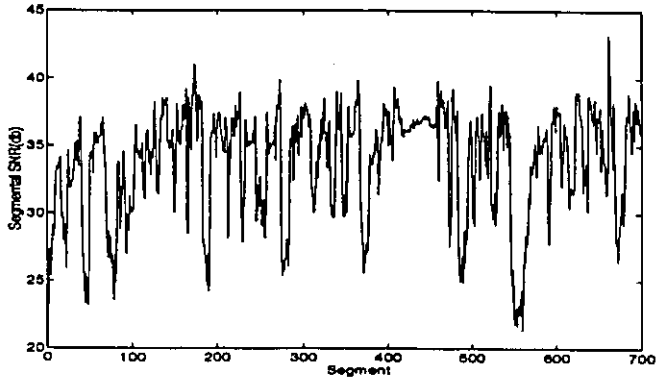


Figure 7.50: The segmental SNR track of the male speech reconstructed by the 16 kb/s nonlinear ADPCM algorithm.

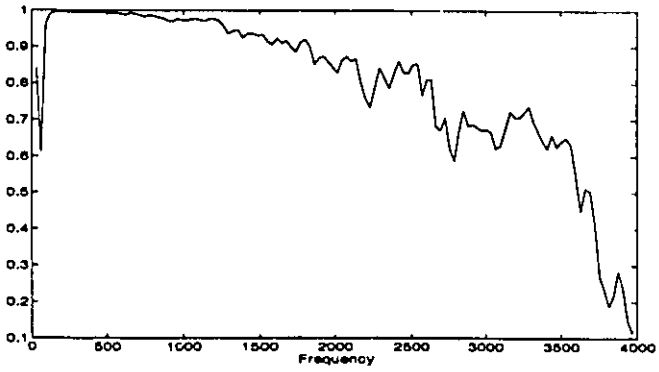


Figure 7.51: By using the 32 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed male speech.

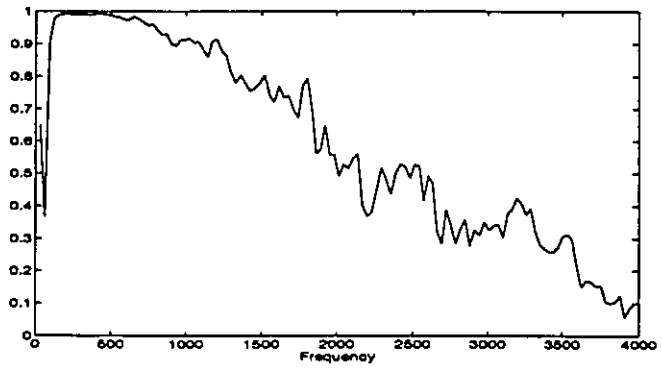


Figure 7.52: By using the 24 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed male speech.

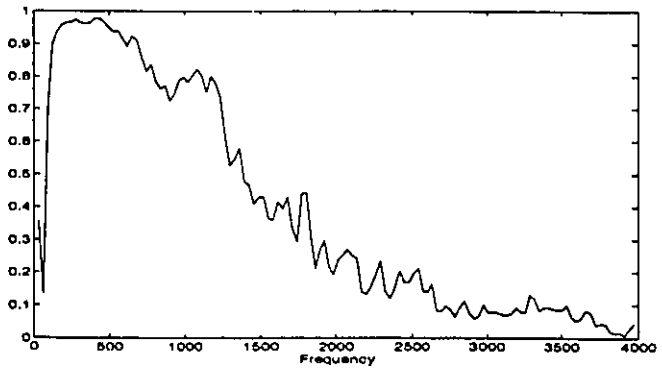


Figure 7.53: By using the 16 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed male speech.

algorithms. These results clearly show that:

(a) The errors between the reconstructed signals and the original signal are very small.

(b) The more levels are used in the quantization, the more accurate is the approximation to the original speech obtained from the nonlinear ADPCM.

2. Segmental Signal-to-Noise (or Distortion) Ratio track curves of the reconstructed female speeches are shown in Figs. 7.57, 7.58 and 7.59. Fig. 7.57 shows the curve of the signal reconstructed with 32 kb/s nonlinear ADPCM, and the average segment SNR of these 800 segments is $37.4dB$. The average segmental SNR values of speech signals reconstructed by the 24 kb/s and 16 kb/s ADPCM algorithms are $33.7dB$ and $29.5dB$, respectively.

3. The frequency-domain coherence function curves are given in Fig. 7.60, Fig. 7.61 and Fig. 7.62. Relevant observations are summed up as follows:

(a) The value of the coherence function between the original and reconstructed female speech, using nonlinear 32 kb/s ADPCM, is higher than or equal to 0.9 on a wide frequency band (100Hz–1,900Hz), and is higher than 0.8 between 1,500Hz to 3,400Hz. This means that this reconstructed speech may almost exactly preserve the information content of the original speech signal inside the main frequency band.

(b) In using the lower bit-rate nonlinear ADPCM algorithm, the frequency bandwidth, on which the value of a coherence function is higher than 0.9, is shortened. For the 24 kb/s algorithm, the frequency band stretches from 100Hz to 1,400Hz; for 16 kb/s algorithm, the frequency band extends from 100Hz to 1,000Hz. However, on wider frequency bands (100Hz-3,200Hz, 24 kb/s algorithm;

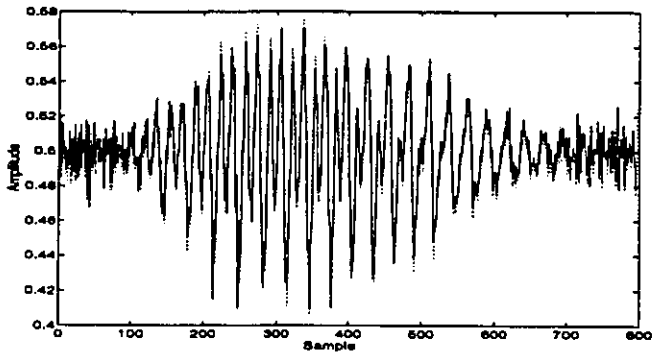


Figure 7.54: Reconstruction of a female speech using the 32 kb/s nonlinear ADPCM algorithm. Continuous curve: original speech signal; dashed curve: reconstructed speech signal.

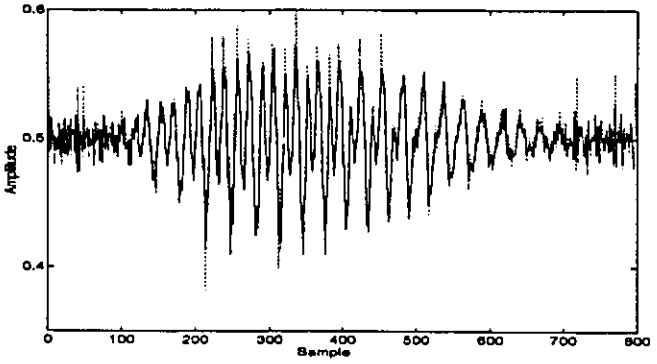


Figure 7.55: Reconstruction of a female speech using the 24 kb/s nonlinear ADPCM algorithm.

100Hz-1,700Hz, 16 kb/s algorithm), the values of the two coherence functions are still higher than 0.5.

(c) Compared with the coherence function for the previous male speech, at the same frequency point, the value of the coherence function for the female speech is higher.

4. Listening evaluations of the reconstructed female speech are summarized as follows:

There is little difference between the original and the reconstructed female speech by the 32 kb/s nonlinear ADPCM algorithm (the estimated quality

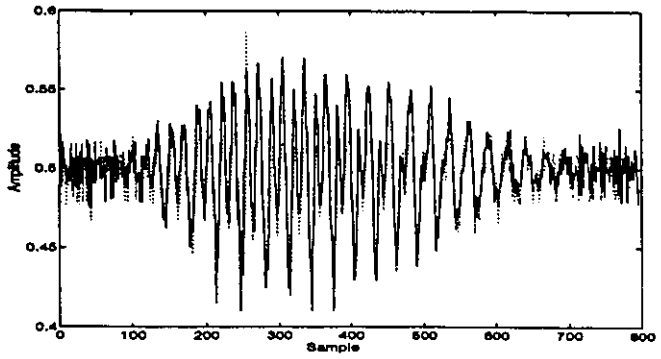


Figure 7.56: Reconstruction of a female speech using the 16 kb/s nonlinear ADPCM algorithm.

Table 7-3

	32 kb/s Nonlinear ADPCM	24 kb/s Nonlinear ADPCM	16 kb/s Nonlinear ADPCM
Mean-square Error	$9.5*10^{-5}$	$2.25*10^{-4}$	$4.82*10^{-4}$

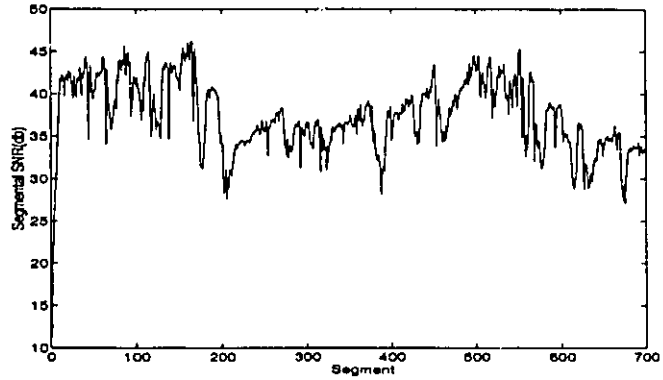


Figure 7.57: The segmental SNR track of the female speech reconstructed by the 32 kb/s nonlinear ADPCM algorithm.

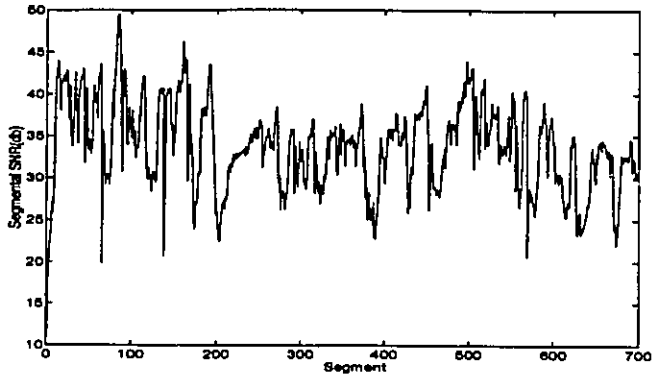


Figure 7.58: The segmental SNR track of the female speech reconstructed by the 24 kb/s nonlinear ADPCM algorithm.

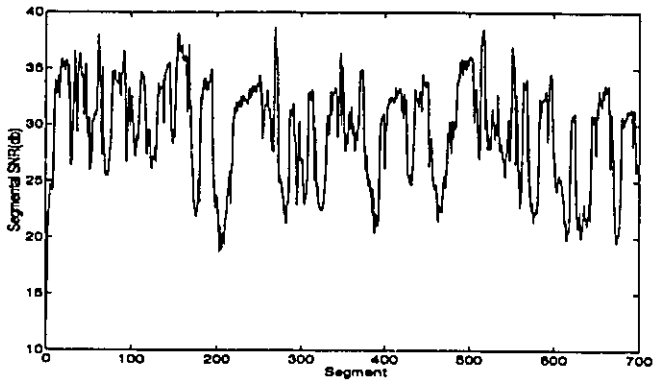


Figure 7.59: The segmental SNR track of the female speech reconstructed by the 16 kb/s nonlinear ADPCM algorithm.

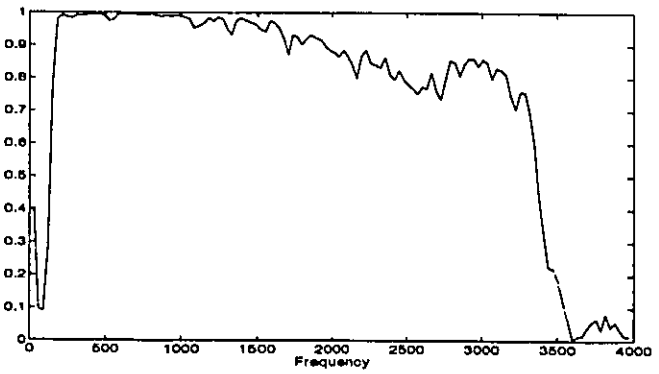


Figure 7.60: By using the 32 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed female speech.

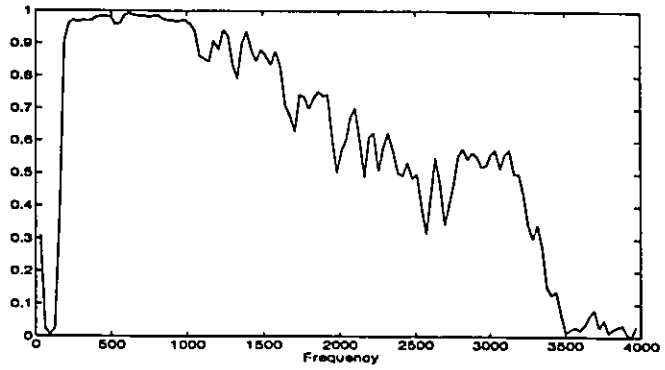


Figure 7.61: By using the 24 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed female speech.

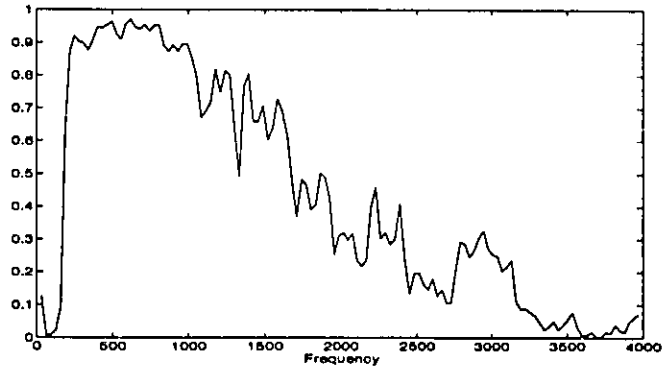


Figure 7.62: By using the 16 kb/s nonlinear ADPCM algorithm, the coherence function between the original and reconstructed female speech.

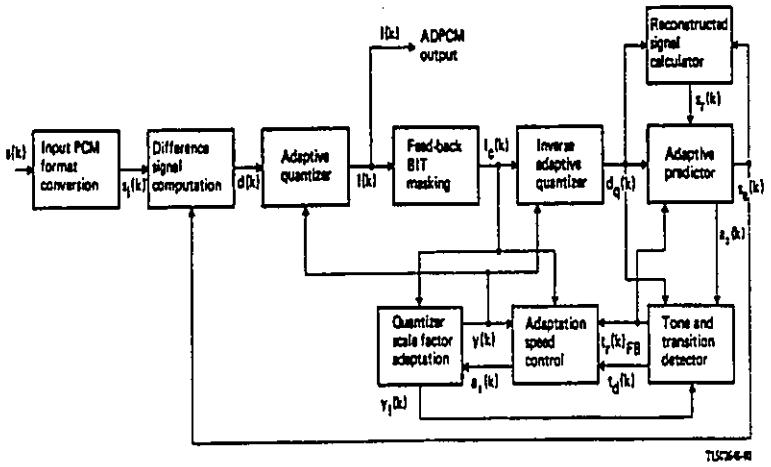


Figure 7.63: Encoder block of an *AT&T* 32 kb/s linear ADPCM algorithm.

score of the reconstructed speech $\geq$ 3.5). There is a little click noise in the female speech reconstructed by the 24 kb/s nonlinear system, and the estimated quality score is not lower than 3.0. In using the lowest bit rate (16 kb/s) ADPCM algorithm, some annoying noise is heard in the reconstructed female speech; and the estimated quality score may be somewhat higher than 2.5.

7.3 Comparison with the Linear ADPCM

In January 1994, *AT&T* Bell laboratory provided us the software of linear ADPCM algorithm, which was recommended by CCITT as a standard 32 kb/s ADPCM algorithm [72]. The schematic encoder block and the decoder block of the 32kb/s linear ADPCM are shown in Fig. 7.63 and Fig. 7.64.

The predictor used in *AT&T* algorithm is a linear adaptive predictor, which has a construction that is similar to the linear part of an adaptive predictor described in Section

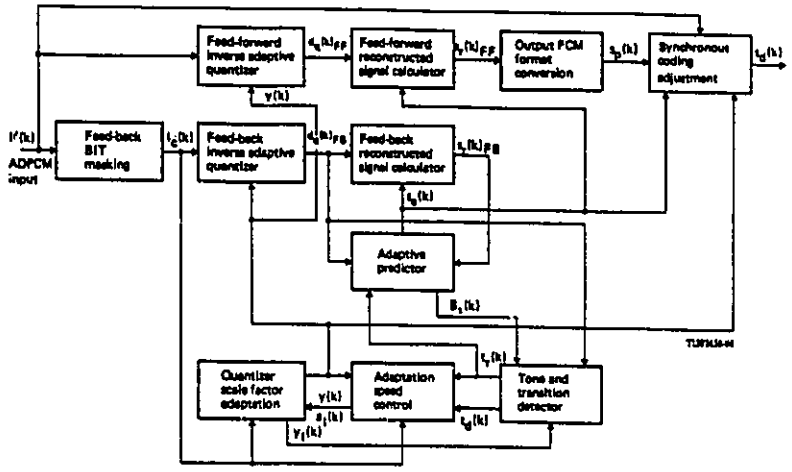


Figure 7.64: Decoder block of an *AT&T* 32 kb/s linear ADPCM algorithm.

6.3.3. The primary function of the adaptive predictor is to compute the signal estimate from the quantized difference signal. Two adaptive predictor structures are used, a sixth order that models zeros and a second section that models poles in the input signal. This dual structure effectively caters to the variety of input signals which might be encountered in practice.

Compared with the ADPCM system described in [36, 2], the 32 kb/s linear ADPCM algorithm of *AT&T* company is a much more complex system. Several auxiliary parts are used in the encoder and decoder, exploiting prior knowledge about the speech signal. In the encoder part, *Tone and Transition Detector,* and *Reconstructed Signal Calculator* are added. In the decoder, with the exception of the above added parts, *Synchronous Coding Adjustment* is used. The primary function of *Synchronous Coding Adjustment* is to prevent cumulative distortion and improve the quality of the reconstructed speech. In addition, a feed-forward path is included to reproduce the speech signal in the decoder. One of reasons to build such a complex system is to overcome weaknesses that may arise in using a linear predictor model of a speech signal.

In order to further evaluate the performance of our nonlinear ADPCM algorithm, we compare the features of speech signals reconstructed by ours and the $AT\&T$ linear ADPCM algorithm. The comparison items are included here:

- Time domain:
  1. Approximation of the reconstructed speech waveforms to the corresponding original speech wave.
  2. Segmental SNR track curve of reconstructed speeches and the average segmental SNR (dB).
- Frequency domain:
  1. Power spectrum plots of reconstructed speeches.
  2. Coherence functions between the original speech and corresponding reconstructed ones.
- Listening Tests.

The test speech is the four-person conversation described in section 7.1.

7.3.1 Reconstructed Speech Waveforms

Fig. 7.65(a) presents the waveform of the speech reconstructed by the $AT\&T$ 32 kb/s linear ADPCM algorithm. The continuous curve pertains to the original speech signal wave, and the dashed curve pertains to the reconstructed versions measured at the output of the decoder. Fig. 7.65(b) is the plot of the corresponding squared error between the original and reconstructed speech. Figs. 7.66(a), 7.67(a) and 7.68(a) show respectively the speech waveforms reconstructed by the nonlinear 32 kb/s, 24 kb/s and 16 kb/s ADPCM algorithm, and Figs. 7.66(b), 7.67(b) and 7.68(b) show the plots of their squared errors. In Fig. 7.69,

the dashed line is the squared error plot of 32 kb/s linear ADPCM algorithm, and the continuous curve presents the plot of the 32 kb/s nonlinear ADPCM algorithm.

These figures show that the 32 kb/s nonlinear ADPCM algorithm provides a better approximation to the original speech than that of the 32 kb/s linear ADPCM algorithm, even though the linear system has many auxiliary parts. However, the approximations provided, respectively, by the 24 kb/s, and the 16 kb/s nonlinear ADPCM are slightly poorer than that of the 32 kb/s linear ADPCM algorithm. These results use only 800 continual samples of the conversation signal.

7.3.2 Segmental SNR Track Curves

The segmental SNR track curve of the conversation reconstructed by the 32 kb/s linear ADPCM system is given in Fig. 7.70. The average segmental SNR of 600 segments is $26.02dB$.

Three segmental SNR track curves, which correspond to those speech signals reproduced by nonlinear ADPCM algorithms, are shown in Fig. 7.71, Fig. 7.72 and Fig. 7.73. The values of the average segmental SNR for the 32 kb/s, 24 kb/s and 16 kb/s nonlinear ADPCM, are $38.2dB$, $34.4dB$ and $30.5dB$ respectively.

From these figures, we make the following observations: Though its average segmental SNR is the lowest, the segmental SNR track curve of the $AT\&T$ ADPCM algorithm is the most stable. The fluctuation of this curve is lower than $5dB$; those of the segmental SNR track curves with nonlinear ADPCM algorithms are over 10 dB.

7.3.3 Power Spectrum of Output Signals

The power spectrum plots are drawn in Fig. 7.74, Fig. 7.75, Fig. 7.76, Fig. 7.77 and Fig. 7.78. Fig. 7.74 is the estimated power spectrum of the original conversation. Fig. 7.75 is the estimated power spectrum of the conversation reconstructed by the $AT\&T$'s 32 kb/s linear ADPCM algorithm. Correspondingly, Figs. 7.76$\sim$ 7.78 present the estimated spectrum plots of these conversations by three different forms of the nonlinear ADPCM algorithm.

From these power spectrum plots, we may make following observations:

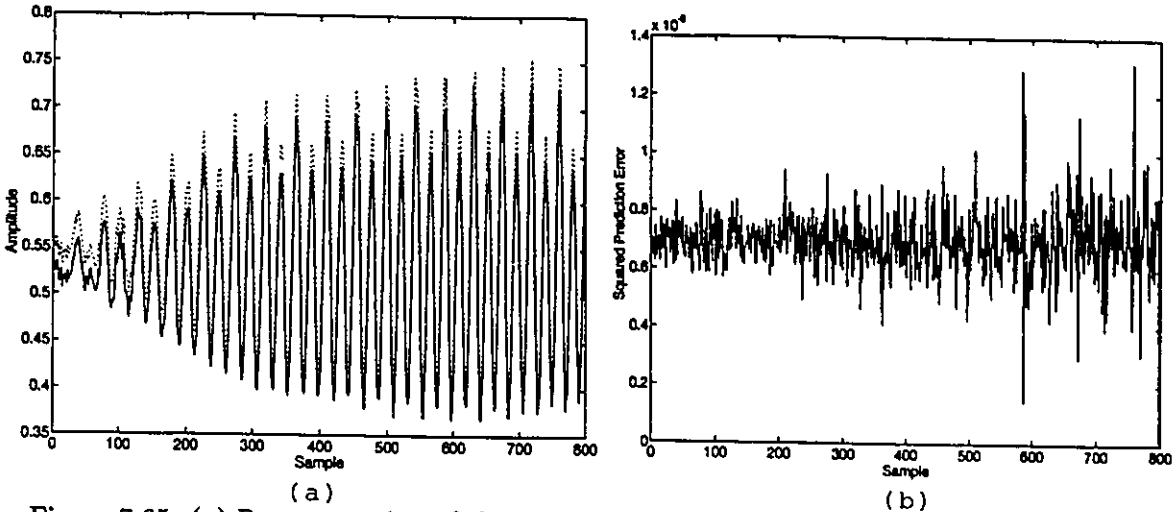
1. There is a definite difference between the power spectrum of the original speech and that of speech signals reconstructed by a linear or nonlinear algorithm. The difference confirms that some information is lost in the process of coding and decoding.

2. Power spectrum plots of two speech signals reconstructed by the linear and nonlinear 32 kb/s ADPCM algorithms are similar to each other. Their power density distribution at low frequency is the same as that of the original conversation, and the distributions at high frequency are different when compared with that of the original one. This means that the reconstructed errors concentrate mainly at the high frequency end.

3. As the quantization levels decrease, the power spectrum of these reconstructed speech signals at the high frequency end differs greatly from that of the original speech signal.

7.3.4 Coherence Functions

We show plots of the coherence functions between the original conversation and reconstructed ones in Fig. 7.79, Fig. 7.80, Fig. 7.81 and Fig. 7.82. We see that:



(a) (b)

Figure 7.65: (a) Reconstruction of the conversation with the 32 kb/s linear ADPCM algorithm. Continuous curve: original speech signal; dashed curve: reconstructed speech signal. (b) The plot of the corresponding squared reconstruction error.

1. The coherence functions corresponding to the two kinds of 32 kb/s ADPCM algorithms are similar to each other. From 200Hz to 1,850Hz, the values of two coherence functions are over 0.9; from 1,850Hz to 3,000Hz, the values of two functions are over 0.7. However, in the higher frequency region (3,000Hz-3,500Hz), the value of the coherence function corresponding to the nonlinear algorithm seems to be higher than that of the linear algorithm.

2. It is apparent that there are some differences between the coherence function corresponding to the 24 kb/s nonlinear ADPCM algorithm and the coherence function corresponding to the 32 kb/s linear ADPCM algorithm. The frequency band on which the coherence value corresponding to the 24 kb/s nonlinear ADPCM algorithm is higher than 0.9, stretches from 200Hz to 1,850Hz. Between 1,850Hz to 3,500Hz, the average value is about 0.55. It seems that there are more higher frequency errors in the speech reconstructed by the 24 kb/s nonlinear ADPCM algorithm.

3. Using the 16 kb/s nonlinear ADPCM algorithm, between 200Hz to 1,400Hz, the average value of the coherence function between the original and reconstructed speech

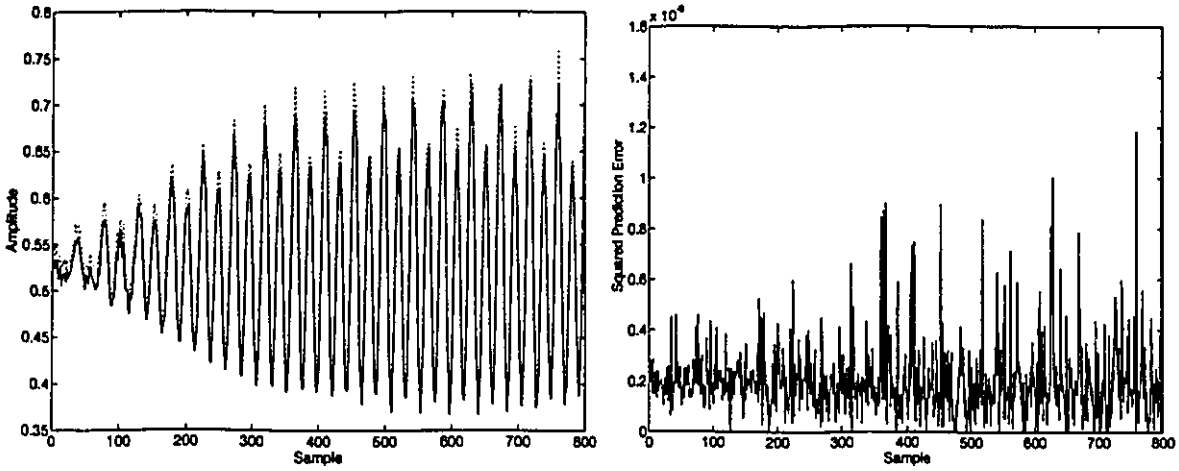


Figure 7.66: (a) Reconstruction of the conversation with the 32 kb/s nonlinear ADPCM algorithm. (b) A plot of the corresponding squared reconstruction error.

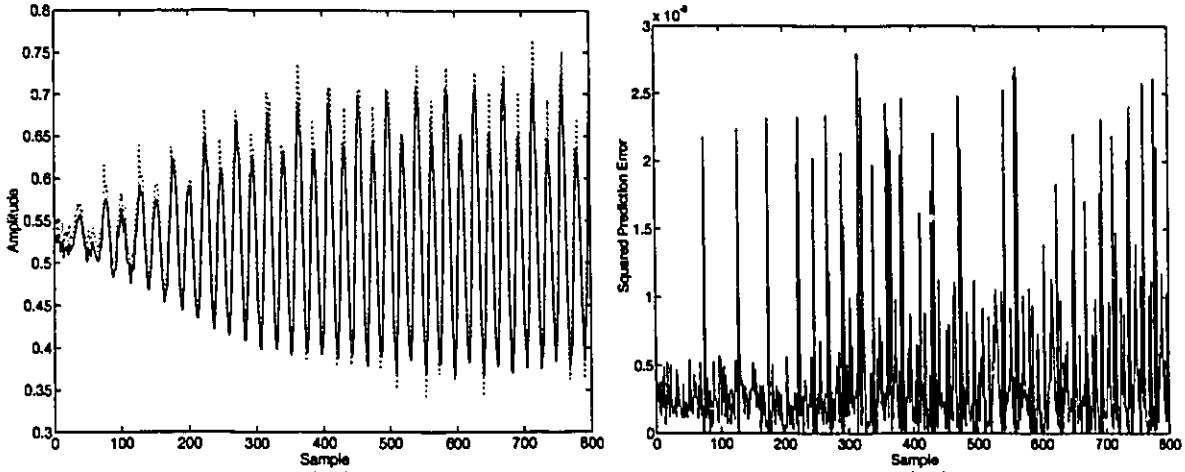
is about 0.9. Beyond that band, the average value is only about 0.25. Compared with that of the 32 kb/s linear ADPCM algorithm, speech reconstructed by the 16 kb/s nonlinear ADPCM algorithm exhibits a greater loss in information content.

7.3.5 Listening Tests

The final item of comparing the linear and nonlinear ADPCM algorithms is the listening quality of their reconstructed speeches. Several colleagues in our research laboratory, who were invited to listen these reconstructed conversation, gave the following comments:

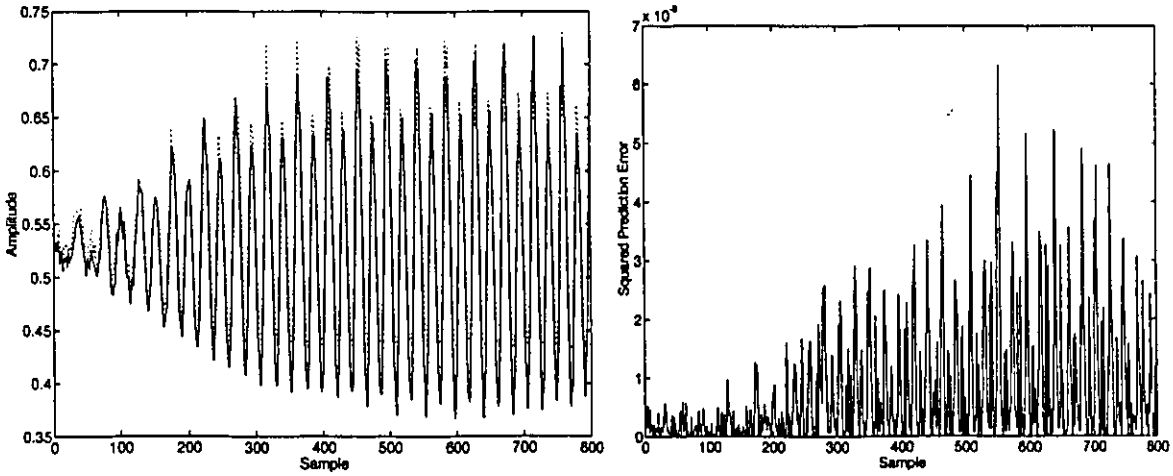
- The quality of the conversation reconstructed by the 32 kb/s nonlinear ADPCM algorithm is superior. There are few background noises and clicks in the reconstructed speech.

- There is a slight random background noise in the speech signal reconstructed by the decoder of the 32 kb/s linear ADPCM algorithm. The voice quality of this reconstructed conversation is more gentle as compared with that speech reconstructed by



(a) (b)

Figure 7.67: (a) Reconstruction of the conversation with the 24 kb/s nonlinear ADPCM algorithm. (b) A plot of the corresponding squared reconstruction error.



(a) (b)

Figure 7.68: (a) Reconstruction of the conversation with the 16 kb/s nonlinear ADPCM algorithm. (b) A plot of the corresponding squared reconstruction error.

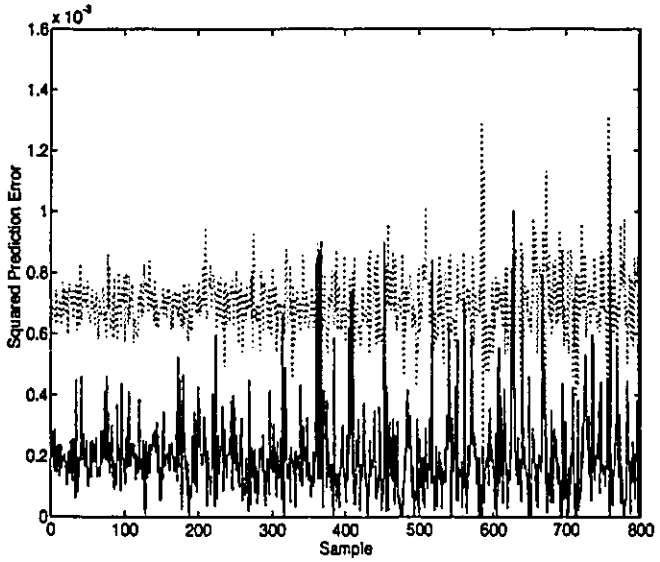


Figure 7.69: Comparison of squared reconstruction errors. Continuous curve: the plot of squared error using the nonlinear algorithm; dashed curve: the plot of squared error using the linear algorithm.

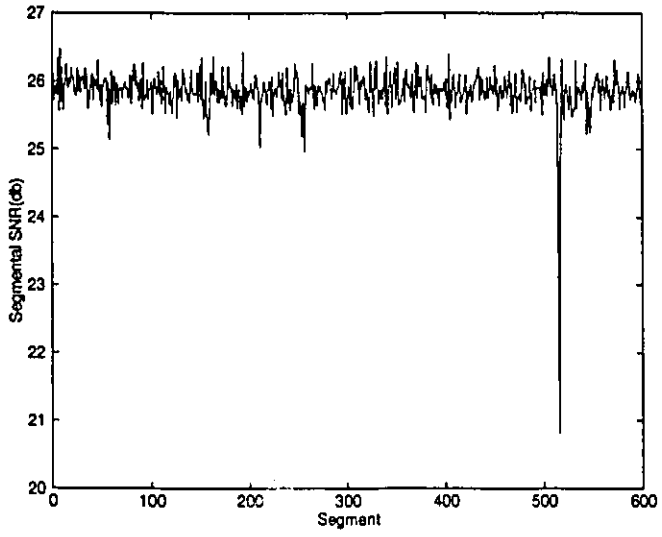


Figure 7.70: A segmental SNR track curve of the reconstructed conversation in using the 32 kb/s linear ADPCM algorithm.

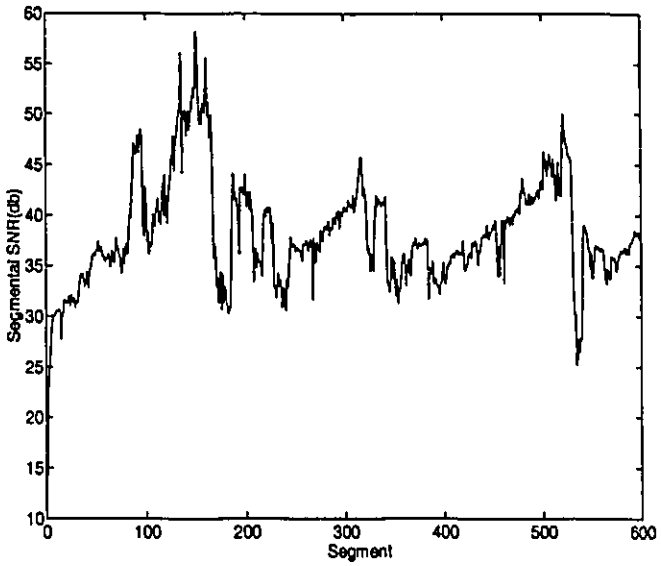


Figure 7.71: A segmental SNR track curve of the reconstructed conversation in using the 32 kb/s nonlinear ADPCM algorithm.

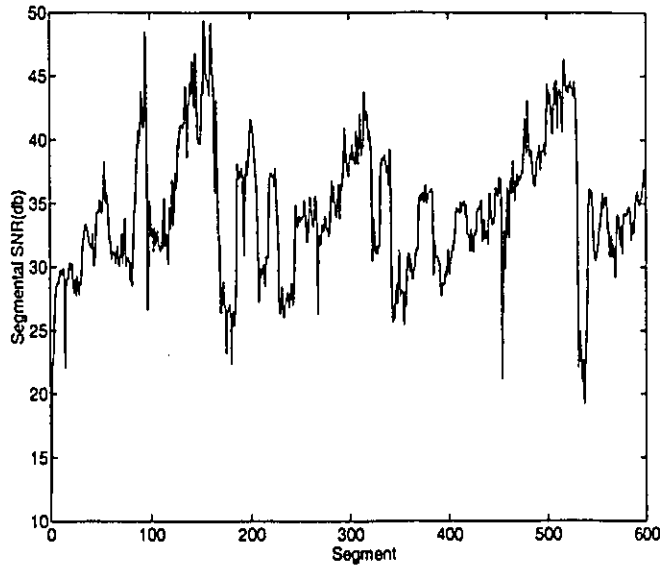


Figure 7.72: A segmental SNR track curve of the reconstructed conversation in using the 24 kb/s nonlinear ADPCM algorithm.

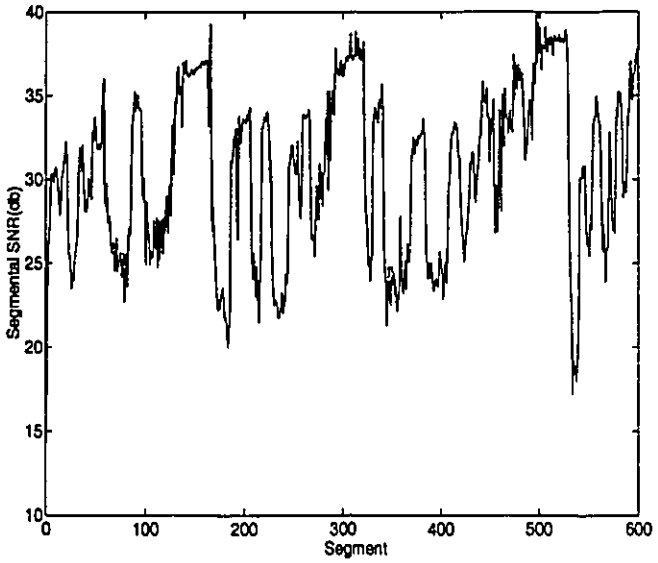


Figure 7.73: A segmental SNR track curve of the reconstructed conversation in using the 16 kb/s nonlinear ADPCM algorithm.

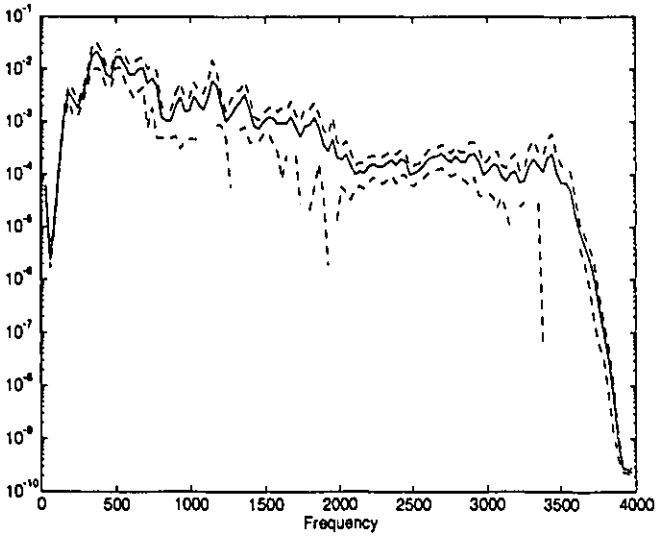


Figure 7.74: Power spectrum of the original conversation. Continuous curve: typical result; dashed curves: upper and lower bounds representing 95 percent confidence.

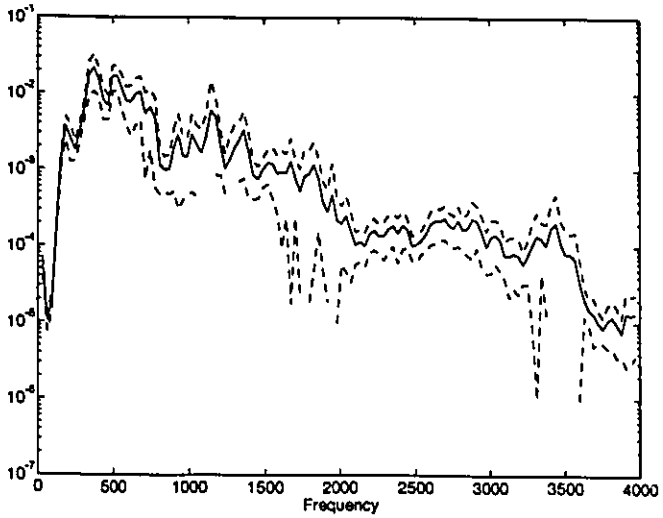


Figure 7.75: Power spectrum of the conversation reconstructed by the 32 kb/s linear ADPCM algorithm.

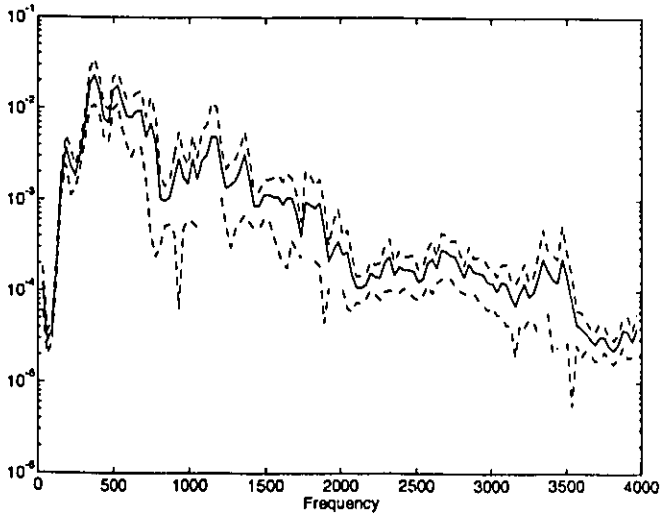


Figure 7.76: Power spectrum of the conversation reconstructed by the 32 kb/s nonlinear ADPCM algorithm.

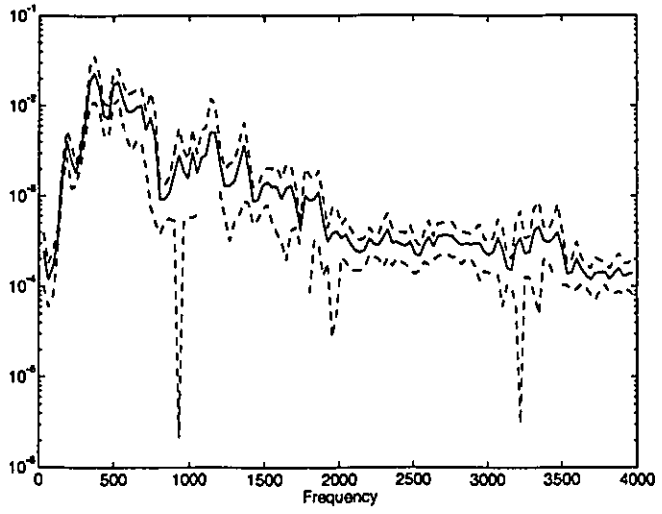


Figure 7.77: Power spectrum of the conversation reconstructed by the 24 kb/s nonlinear ADPCM algorithm.

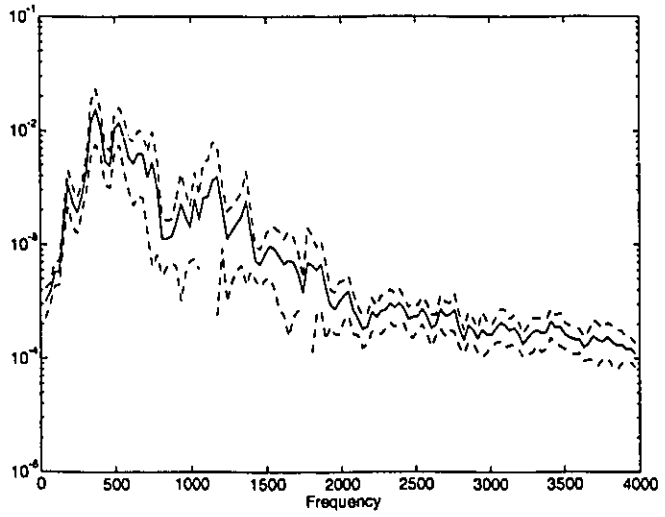


Figure 7.78: Power spectrum of the conversation reconstructed by the 16 kb/s nonlinear ADPCM algorithm.

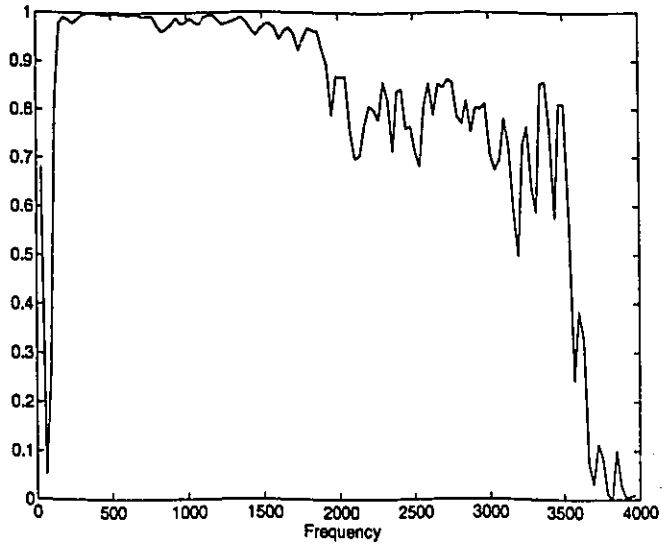


Figure 7.79: Coherence function between the original conversation and reconstructed one by the 32 kb/s linear ADPCM algorithm.

the nonlinear algorithm. This may be due to the inclusion of the *Synchronous Coding Adjustment* in the linear system, which decreases the fluctuation of segmental SNR track curve. The quality score of the speech signals reconstructed by using both nonlinear and linear 32 kb/s ADPCM algorithm is estimated to be over 3.5.

- Although the conversation reconstructed by the 24 kb/s nonlinear ADPCM is still clear, there are apparent clicks and noise in the decoded speech. The estimated quality score of this speech is about 3.0.

- In addition to the clicks and whistles, there is background noise in the conversation reproduced by the 16 kb/s nonlinear ADPCM algorithm. The quality score of this reconstructed conversation is estimated to be somewhat higher than 2.5.

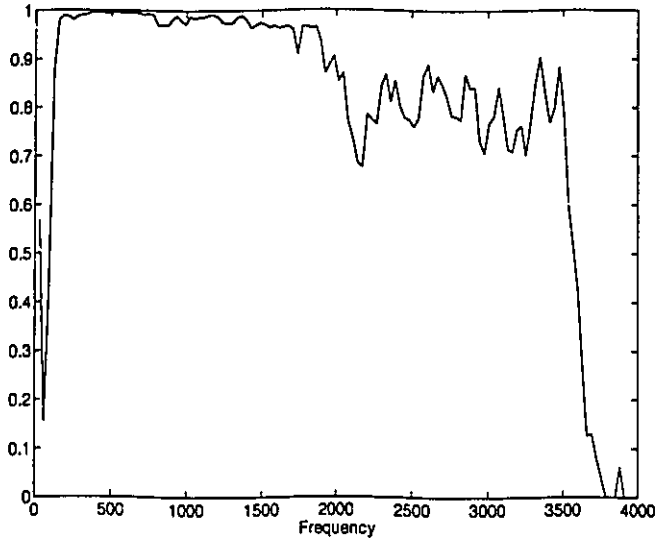


Figure 7.80: Coherence function between the original conversation and reconstructed one by the 32 kb/s nonlinear ADPCM algorithm.

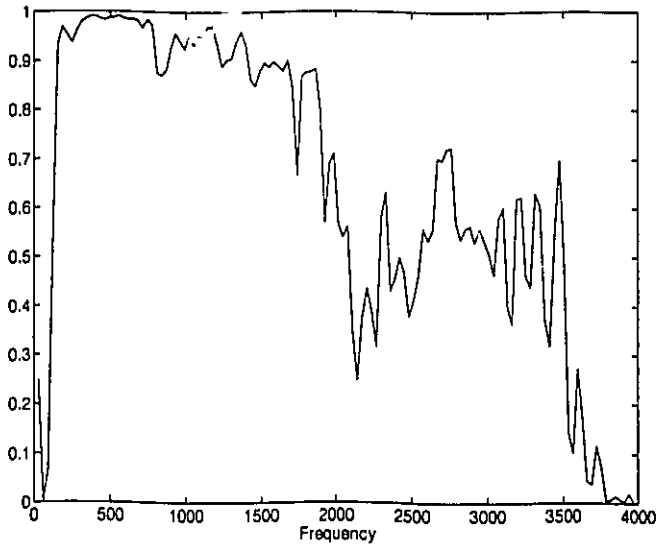


Figure 7.81: Coherence function between the original conversation and reconstructed one by the 24 kb/s nonlinear ADPCM algorithm.

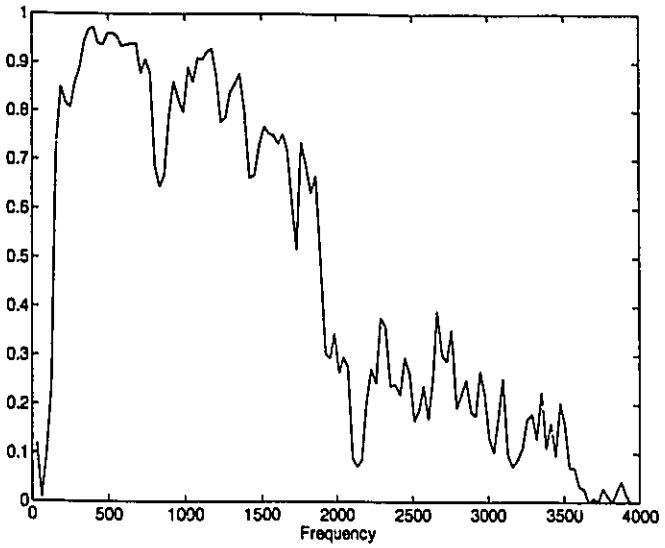


Figure 7.82: Coherence function between the original conversation and reconstructed one by the 16 kb/s nonlinear ADPCM algorithm.

7.4 Discussion

7.4.1 Processing Delay of the Nonlinear ADPCM

In principle, the processing delay of a linear ADPCM is caused by the linear predictor. The linear predictor is an infinite impulse filter, which is composed of two structures, a six-order section that models zero points in the input signal and a two-order section that models pole points in the input signal. The function of the linear predictor is described as

$$\hat{s}(n+1) = a_1(n)s(n) + a_2 s(n-1) + \mathbf{b}^T(n)\hat{\Delta}(n)$$

where $\mathbf{a}(n) = [a_1(n), a_2(n)]^T$ and $\mathbf{b}(n)$ are coefficient vectors, and $\hat{\Delta}$ is the inverse-quantized transmitted signal vector.

Therefore, in using the linear predictor, there is a two sampling-unit delay between the input signal sample and corresponding output sample. The two same predictors are used in the encoder and decoder of the ADPCM system. The total processing delay between an

original speech sample and the reconstructed speech sample at the decoder output is four sampling units. With 8,000Hz sampling frequency, each sampling unit is $0.125ms$. In the linear ADPCM, the processing delay of one signal sample is over 0.6ms.

In addition to using the infinite impulse filter as the linear subsection, the adaptive predictor of the nonlinear ADPCM system also use the Pipelined Recurrent Neural Network as the nonlinear subsection. The PRNN introduces more processing delay in the reconstructed speech signal.

According to the mathematical model of PRNN system presented in Section 4.3, there is a $p + M - 1$ sampling-unit delay between an input sample and the corresponding output one; p is the number of external inputs on each module and M is the number of modules in the PRNN. The total processing delay of a PRNN-based predictor consists of the processing delay in its nonlinear subsection plus the processing delay in its linear subsection, which works oout to be $p + M - 1 + 2$ sampling units.

With two PRNN-based predictors used, one in the encoder and the other in the decoder of a nonlinear ADPCM system, the total processing delay is $2p + 2M + 2$ sampling units.

The nonlinear ADPCM system used in our subjective tests consists of four modules ($M = 4$), and each module has four external inputs ($p = 4$). With a sampling frequency of $8,000Hz$, the total processing delay produced by the nonlinear ADPCM is about $2.25ms$. This processing delay is higher than the processing delay produced by the linear ADPCM ($0.125ms$).

Ideally, for telephone communications the processing delay should be zero for ADPCM, and $25ms$ for adaptive sub-band coding [64]. In the ADPCM studies considered in this chapter, the processing delay of $2.25ms$ for our nonlinear system is larger than that for the corresponding linear system, but it is considered to be small enough for practical use.

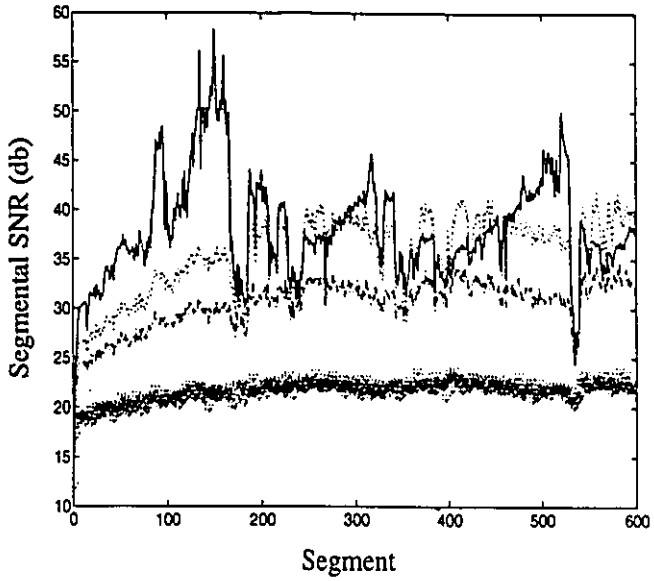


Figure 7.83: The segmental SNR tracks of the conversation reconstructed by the 32 kb/s nonlinear ADPCM algorithm with different noisy inputs

7.4.2 Sensitive to noise in the input speech

In Section 6.3, we mentioned that the stability of the on-line training and processing procedure of the pipelined recurrent neural network is sensitive to the presence of input noise and calculation error. The presence of noise may cause divergence of the nonlinear predictor. For a complete assessment of the performance of the new nonlinear ADPCM system, we therefore need to evaluate the effect of noise in the input signal on system performance.

In evaluating the sensitivity of our nonlinear ADPCM to noise in the input speech signal, a series of subjective tests are performed on the nonlinear ADPCM with different noisy test speech signals. These noisy test speech signals consist of the "pure" four-person conversation and white noise of different levels. The signal-to-noise ratio (SNR) values of the resulting speech signals are estimated to be $30dB$, $25dB$ and $20dB$ respectively. The indices of evaluation are:

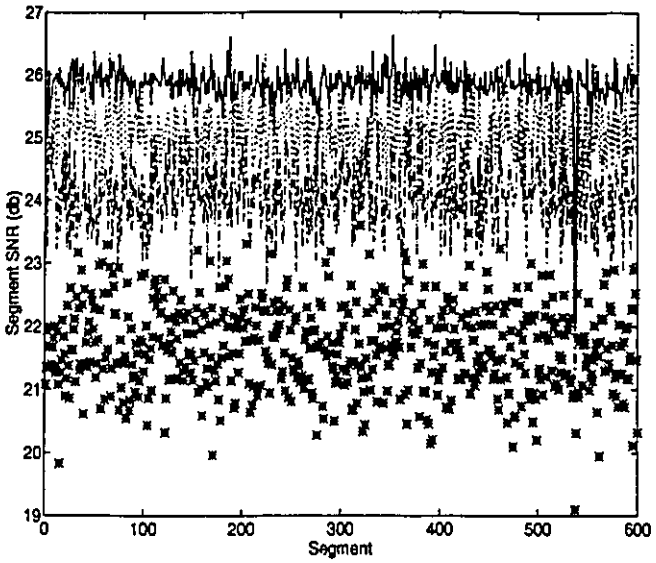


Figure 7.84: The segmental SNR tracks of the conversation reconstructed by the 32 kb/s linear ADPCM algorithm with different noisy inputs

1. *Segmental output Signal-to Noise Ratio tracks.*

2. *Listening test.*

For comparison, the same subjective tests are made on the *AT&T* linear ADPCM system.

Figure 7.83 shows the segmental SNR tracks of these noisy conversation signals reconstructed by the 32 kb/s nonlinear ADPCM. Figure 7.84 describes the corresponding ones produced by the linear ADPCM system with the same inputs. The solid lines describe the segmental tracks of the reconstructed signals by nonlinear and linear 32kb/s ADPCM in two figures for the case of noiseless speech signal. The dotted lines present the segmental tracks of reconstructed speech signals for the two systems for $30dB$ SNR. The dashdot lines are the segmental tracks corresponding to $25dB$ SNR. Finally, the star lines describe the segmental SNR tracks of the reconstructed speech signals in the two systems for $20dB$ SNR.

From these two figures, we may make the following observations:

- As the noise power in the input speech signal is increased, the level of output segmental SNR tracks is progressively reduced.

- In using the linear ADPCM system, the average segmental values of the solid track, dotted track, dashdot track, and the star track are $26.02dB$, $25.5dB$, $24.2dB$, and $21.6dB$ respectively.

- In using the nonlinear ADPCM system, the average segmental value of dotted track (input SRN being $30dB$) is $34.5dB$. As the the number of segments is larger than 200, the output segmental SNR track (dotted lines) of the nonlinear ADPCM tends toward the solid track, with an average value of $38.2dB$. This behavior is not found in the segmental tracks of the linear system. The implication here is that the learning and generalizing ability of a neural network help the nonlinear system to gradually recover its normal performance when the input speech signal is only slightly noisy.

- As the noise in the input speech signal is increased further, the performance of the nonlinear ADPCM system appears to break down quickly. The average value of the dashdot segmental track (input SNR being $25dB$) is $29.04dB$. When the input SNR is $20dB$, the average output segmental value is dramatically reduced down to $20.5dB$. This means that the nonlinear ADPCM system is notoriously sensitive to the presence of a strong noise composed in the input signal.

Evaluation of the listening test for reconstructed speech signals is summarized as follows:

- When the noise level in the input speech is relatively low (e.g. its SNR is on the order of $30dB$), the quality of conversation reconstructed by the nonlinear 32 kb/s ADPCM remains fairly good. There is some random background noise and clicks in the reconstructed conversation. However, it is not difficult to understand the conversation.

The quality score may be 2.

- When the noise power level is high, quality of the reconstructed speech by the nonlinear ADPCM is poor. There is strong background noise and apparent clicks in the reconstructed conversation. In particular, when the noisy input has $20dB$ SNR, the reconstructed conversation is not at all clear. Listeners are annoyed by the presence of a strong background noise. The quality score is lower than 1.

- For the linear 32kb/s ADPCM system, the quality of conversations reconstructed is gradually decreased as the noise power level in the signal is increased. For an input SNR of $30dB$, the reconstructed conversation is still clear, but the background noise is quite apparent. The quality score of this reconstructed conversation is lower than 2. When the input SNR is reduced further down to $25dB$ or $20dB$, the reconstructed conversations are not at all clear. The quality of the corresponding reconstructed conversations is assessed to be poor.

7.5 Summary

In this chapter, the performance of the nonlinear ADPCM algorithm is evaluated with subjective tests. The conclusion drawn from the evaluation may be summarized as follows:

- Subjective tests with three entirely different speech signals show that nonlinear adaptive differential pulse-code modulation provides a promising approach for high-quality speech communication at 32 kb/s bits rate.

- By using nonlinear prediction in the encoder and decoder of the ADPCM system, we have been able to produce a performance superior to that attained by the linear version of the system, designed by $AT\&T$ Bell Labs. This improvement in performance was achieved in spite of the fact that the linear system includes many auxiliary parts

designed to exploit known characteristics of speech signals, and the fact that these auxiliary parts were absent from the nonlinear system.

- There is a slight click and background noise in the speech reproduced by the 24 kb/s or 16 kb/s nonlinear ADPCM system. These imperfections are caused mainly by quantization noise. If some auxiliary parts (such as the *synchronous coding adjustment*) are included in the nonlinear system for decreasing the accumulated noise, it should be possible to further improve the performance of the nonlinear ADPCM system.

- The nonlinear ADPCM system appears to be notoriously sensitive to noise in the input speech signal. As the noise power level is slight, the quality of the reconstructed speech signal remains fairly good, and is better than the quality of the speech reconstructed by the linear ADPCM system. However, as the noise power level in the input speech signal is made higher and higher, quality of the reconstructed speech by the nonlinear ADPCM system is sharply degraded.

Chapter 8

Summary of the Thesis

8.1 Summary of this Research

The objective of this thesis is the investigation of the nonlinear adaptive prediction of nonstationary signals whose generation is governed by a nonlinear dynamical mechanism, and its application to speech communications. We concentrated our investigation on dynamical neural networks, the relation between the neural networks and traditional adaptive filters, and the application of neural networks to real-time communications.

First, we asked ourselves three questions: 1) What kind of neural networks are suited for adaptive signal processing? 2) How can an adaptive neural network predictor be designed? 3) Can a neural network predictor be used for the application of real-time communications?

After having done a thorough review of the principles and applications of neural networks, we found that the recurrent neural network could provide a dynamical nonlinear model for nonstationary signals. The *real-time recurrent learning (RTRL) algorithm* provides an on-line (continuous) training method for recurrent neural networks. A practical limitation of the RTRL algorithm is that its computation complexity increases exponentially

with the number of neurons in the network.

Based on the engineering principle of divide and conquer, and the biological principle of neural network modules, a pipelined structure of recurrent neural networks (PRNN) is proposed. The features of PRNN are:

- The PRNN is composed of M separate modules, each of which is designed to perform nonlinear adaptive filtering on an appropriately delayed version of the input signal vector.
- The modules of the PRNN are identical, each designed as a fully recurrent neural network with a single output neuron.
- Information flow into and out of the modules proceeds in a synchronized fashion.

In the thesis, the mathematical model and the real-time learning algorithm of the PRNN are discussed. The pipelined recurrent neural network enhances the computational ability and stability of the conventional recurrent neural network.

In order to study the nonlinear adaptive prediction of nonstationary signals, a computationally efficient scheme is described. The complete prediction consists of two subsections, one of which performs a nonlinear mapping from the input space to an intermediate space with the aim of linearizing the input signal, and the other subsection performs linear mapping from the new space to the output space. The nonlinear subsection consists of a pipelined recurrent neural network, and the linear section consists of a conventional tapped-delay-line filter. The nonlinear predictor, called PRNN-based adaptive predictor, exhibits the following characteristics.

- The neural network-based predictor learns to adapt to statistical variations of the incoming time series while, at the same time, the prediction is going on.

- The adaptation of the nonlinear prediction follows a self-organized learning method, with the desired signal being derived from the incoming time series.

The signal processing capability of this new adaptive prediction has been demonstrated by studying the one-step adaptive prediction of speech signals, and the adaptive prediction of sea clutter. For these applications, it is shown that this nonlinear adaptive prediction outperforms the traditional linear adaptive scheme in a significant way. In general, the nonlinear predictor provides 3 to 5 dB more prediction gain than linear ones. However, an important attribute of the new neural network-based adaptive predictor is its high computational efficiency. This attribute makes it possible to use dynamical neural networks in the real-time speech communications.

Turing to the task of communication, the speech waveform coding has been studied in detail. In the traditional Adaptive Differential Pulse Code Modulation (ADPCM), an adaptive linear predictor of a sufficiently high order is used to account for the nonstationary nature of speech signals. In this thesis, a new nonlinear adaptive predictor has been successfully applied to the encoder and decoder of an Adaptive Differential Pulse-Code Modulation system. The nonlinear adaptive predictor originates from the adaptive PRNN-based predictor. The nonlinear subsection is still a pipelined recurrent neural network; the linear subsection is an infinite impulse response filter. However, the choice of a suitable neural network predictor architecture for such an application is constrained by the fact that the predictor has to perform its learning in an on-line fashion as the speech signal is being continuously processed, and that the stability of the encoder and decoder of ADPCM should be assured.

In keeping with the nonlinear predictor, all of ADPCM systems (including 4-, 8- or 16-level non-uniform adaptive quantizer) are redesigned. Correspondingly, a new nonlinear

adaptive pulse-code modulation algorithm is developed. The performance of the new algorithm is evaluated with subjective tests. Test results show that nonlinear ADPCM provides a promising approach for the high-quality speech communication at low bits rates. The weakness of the nonlinear ADPCM are the longer processing delay and the notoriously sensitivity to noise in the input speech. We have compared the performance of this nonlinear algorithm to that attained by the linear ADPCM algorithm from several aspects, such as some evaluations in the time domain, in the frequency domain, and listening tests. Speech experiments show that the nonlinear Adaptive Differential Pulse-Code Modulation is able to produce a superior performance.

8.2 Contributions of this Research

The theoretical investigation, deduction, and subsequent simulation performance in this thesis provide several important contributions to the field of the theory of neural networks, nonlinear adaptive prediction of nonstationary signals, the application of neural networks to real-time speech communications. These contributions[1] are briefly summarized here:

1. A new pipelined recurrent neural network (PRNN) is designed. Its continuous learning algorithm makes a breakthrough in the on-line (continuous) training of recurrent neural networks in a computationally efficient manner.

2. A new scheme for the nonlinear adaptive prediction of nonstationary signals is proposed. The nonlinear neural network-based filter learns to adapt to statistical variations of the incoming time series while, at the same time, the information-processing operation is going on.

[1] In addition to the contributions described in this thesis, a nonlinear modified Kalman filtering was proposed. The new algorithm augments the standard Kalman filtering with a model mismatch function. It is effective in decreasing estimation errors and overcoming the divergence problem in Kalman filtering.

3. The new nonlinear neural network-based adaptive predictor has been successfully applied to Adaptive Differential Pulse Code Modulation (ADPCM) of speech communications.

4. Publication of several conference papers and journal papers which report on the research results [8, 38, 48, 49, 54, 56, 73].

8.3 Conclusion

In this thesis, highlights of the results of our study on the dynamical neural network, the nonlinear adaptive prediction of nonstationary signals and their applications on the speech communications were presented. The research results have given exact and positive answers to the three questions posed previously.

As always, any attempt to steer a study along all possible paths, while at the same time exhausting all courses of action is futile, especially with limited time and resources. It is important therefore, to focus on a few issues, with enough depth so that meaningful results may emerge, leaving all the other issues for another time, or another researcher. However, it is only after our methods are validated in the real world, that we may be invited to take a bow.

Bibliography

[1] L. Rabiner, *and* B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] Shuzo Satito, *and* Kazuo Nakata, *Fundamentals of Speech Signal Processing*, ACADEMIC PRESS, 1985.

[3] J. K. Baker, "Stochastic Modeling as a Means of Automatic Speech Recognition", *Ph.D dissertation*, Computer Science Department, Carnegie Mellon University, 1975.

[4] T. Brent, "Nonlinear prediction of speech", in *Int'l Conference on Acoustics, Speech, and Signal Processing*, (Toronto, Canada), pp. 425-428, 1991.

[5] N. Tishby, "A dynamical systems approach to speech processing", in *Int'l Conference on Acoustics, Speech, and Signal Processing*, (New Mexico, USA), pp. 365-368, 1990.

[6] R. E. Kalman, "A new approach to linear filtering and predict on problems", *J. Basic Eng., Trans. ASME*, pp. 35-45, March 1960.

[7] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall–2nd Edition, 1991.

[8] L. Li, *and* S. Haykin, "Modified Kalman filtering with an Optimal Target Function", in *Int'l Conference on Acoustics, Speech and Signal Processing*, (San Francisco, USA) March, 1992.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation,* Macmillan, 1994.

[10] W. S. McCulloch, *and* W. H. Pitts, "A logical calculus of the ideas imminent in nervous activity", *Bulletin of Mathematical Biophysic,* Vol. 5, pp. 115-133, 1943

[11] J. Hopfield, "Neural networks and physical systems with emergent collective computational ability", *Proceedings of National Academy of Sciences,* Vol. 79, pp. 2554-2558, 1982.

[12] D. O. Hebb, *The Organization of Behavior,* New York: Wiley, 1949.

[13] J. A. Anderson, "A simple neural network generation an interactive memory", *Mathematical Bioscience,* Vol. 14, pp. 197-220, 1972.

[14] T. Kohonen, *Self-Organization and Associative Memory,* New York: Springer-Verlag, 1987.

[15] T. Kohonen, "Automatic formulation of topological maps of patterns in a self-organizing system", *Scandinavia Conference on Image Analysis,* (Helsinki), pp. 214-220, 1981.

[16] R. Linsker, "Self-organization in a perceptional network", *Computer,* Vol. 21, pp. 105-117, 1988.

[17] Dan Hammerstrom, "Working with neural networks", *IEEE Spectrum,* July 1993.

[18] R. S. Sutton, "Temporal credit assignment in reinforcement learning", *Ph.D Dissertation,* University of Massachusetts, Amherst, 1984.

[19] J. Hopfield, "Neural networks as physical systems with emergent collective computational, abilities", *Proceeding of the National Academy of Science 79,* 1982.

[20] A. Lapedes, *and* R. Farber, "A self-optimizing, nonsymmetrical neural net for content Addressable memory and patter recognition", *Physica D, Vol. 22, pp. 247-259*

[21] A. B. Herve, *and* Nelson Morgan, *Connectionist Speech Recognition A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[22] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinationrial environment", *Proceedings of IEEE First International Conference on Neural Network*, pp. 609-618, 1987.

[23] F. J. Pineda, "Dynamics and architecture for neural computation", *Journal of Complexity*, Vol. 4, pp. 216-245, 1986.

[24] R. Rohwer, *and* B. Forrest, "Training time-dependence in neural networks", *Proceedings of the IEEE First International Conference on Neural Network*, pp. 701-708, 1987.

[25] C. A. Mead, *and* M. Mahowald, "A silicon model of early visual processing", In *Computational Neuroscience* (E.L.Schwartz, ed), pp. 331-339. Cambridge, MA: MIT Press, 1987.

[26] Ken-Ichi Funahashi, and Yuichi Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks", *Neural Networks*, Vol. 6, pp. 801-806, 1993.

[27] Ken-Ichi Funahashi, "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, Vol. 2, pp. 183-191, 1989.

[28] A. J. Robinson, *and* F. Fallside, "A recurrent error propagation speech recognition system", *Computer Speech and Language*, Vol. 2, pp. 259-274, 1991.

[29] A. F. Atiya, "Learning on a general network", In*Neural Information Processing Systems* (D.Z.Anderson,ed), pp. 22-30, New Tork: American Institute of Physics, 1988.

[30] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks", *Neural Computation*, Vol. 1, pp. 263-269, 1991.

[31] M. Sato, "A learning algorithm to teach spatio-temporal patterns to recurrent neural networks", *Biological Cybernetics*, No. 62, pp. 259-263, 1990.

[32] C. Thierry, "A method for improving the real-time recurrent learning algorithm", *Neural Networks*, Vol. 6, pp. 807-821, 1993.

[33] L. B. Leerink, *and* M. A. Jabri, "Temporal difference learning applied to continuous speech recognition", *In Applications of Neural Networks to Telecommunications*, (Hilsdale, NJ), pp. 252-258, 1993.

[34] T. Jerome, R. Douglas, *and* L.E. Atlas, "Recurrent neural networks and robust time series prediction", *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 240-253, March, 1994.

[35] G. Kechriotis, E. Zervas, *and* E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization", *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 267-278, March, 1994.

[36] N. S. Jayant, *and* P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

[37] D. Gabor, W. P. L. Wilby, *and* R. Woodcock, "An universal non-linear filter, predictor and simulator which optimizes itself by a learning process", *Proceedings of the IEE*, Vol. 108, pp. 422-438, 1961.

[38] S. Haykin, *and* L. Li, "Real-time nonlinear adaptive prediction of nonstationary signal", *CRL Report*, Communications Research Laboratory, McMaster Univ., No. 276, Oct. 1993.

[39] W. Bernard, *and* A. Lehr. Michel, "30 years of adaptive neural networks: perceptron, madaaline, and backprogation", *Proceedings of the IEEE*, Vol. 78, pp. 1415-1442, September 1990.

[40] S. W. Kuffler, J. G. Nicholls, *and* A. R. Martin, *From neuron to brain*, Second Edition, Sinauer Associates Inc, 1984.

[41] C. Mead, "Neuromorphic electronic system", *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1629-1636, 1990.

[42] M. C. Thomas, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition", *IEEE Transaction on Electronic Computers*, Vol. EC-14, pp. 326-334, 1965.

[43] M. B. Priestley, *Non-Linear and Non-Stationary Time Series Analysis*, ACADEMIC PRESS, 1989.

[44] J. C. Houk, "Learning in modular networks", in *Int'l Workshop on Adaptive and learning Systems*, (Yale University), pp. 80-84, May, 1992.

[45] David C. Van Essen, *and* Charles H. Anderson, "Information processing in the primate visual system: an integrated systems perspective", *Science*, pp. 419-423, Jan, 1992.

[46] R. J. Williams, *and* D. Zipser, "Gradient-based learning algorithms for recurrent connectionist networks", *Technical Report NU-CCS-90-9*, Northeastern University, Boston, 1990.

[47] R. J. Williams, *and* D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, Vol. 2, pp. 270-280, 1987.

[48] L. Li, *and* S. Haykin, "A cascaded recurrent neural networks for real-time nonlinear adaptive filtering", in *Int'l Conference on Neural Networks*, (San Francisco, USA), pp. 857-862, 1993.

[49] S. Haykin, *and* L. Li, "Nonlinear adaptive prediction of nonstationary signals", accepted by and to be published on *IEEE Transactions on Signal Processing*, 1994.

[50] N. I. Johnson, *Continuous Univariate Distributions*, Houghton Mifflin Company, Boston, 1970.

[51] L. Wu, *and* M. Niranjan, "On the design of nonlinear speech predictor with recurrent nets", in *Int'l Conference on Acoustics, Speech and Signal Processing*, (Australia), Vol. II, pp. 529-532, April, 1994.

[52] R. J. Williams, *and* J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent neural network trajectories", *Neural Computation*, Vol. 2, pp. 490-501, 1990.

[53] K. Theomas, "The innovation approach to detection and estimation theory", *Processing of the IEEE*, Vol. 58, No. 5, May, 1970.

[54] S. Haykin, *and* L. Li, "Pipelined recurrent neural networks for adaptive prediction of nonstationary signals", in *Int'l Workshop on Adaptive and Learning Systems*, (Yale University), June, 1994.

[55] B. S. Atal, "Speech analysis and synthesis by linear prediction of the speech wave", *The Journal of the Acoustical Society of America*, Vol. 50, No. 2, pp. 637-655, 1971.

[56] S. Haykin, *and* L. Li, "16kb/s adaptive differential pulse code modulation of speech", in *Int'l Workshop on Applications of Neural Networks to Telecommunications*, (Princeton, NJ), pp. 132-138, 1993.

[57] B. S. Atal, *and* M. R. Schroeder, "Adaptive predictive coding of speech signals", *Bell syst. Tech.J.*, 49-8, 1970.

[58] N. Benvenuto, "The 32-kb/s ADPCM coding standard." *AT&T Technical Journal*, September/October,Volume 65, Issue 5, 1986.

[59] Chris Rowden(ed), *Speech Processing*, McGRAW-HILL Book company, 1992.

[60] N. Takao, "A CCITT standard 32 kbit/s ADPCM LSI codec", *IEEE Transaction on ASSP*, Vol. 35, No. 2, February 1987.

[61] B. S. Atal, "Predictive coding of speech at low bit rates", *IEEE Transaction Communications, Vol. 30, No. 4.* April 1982

[62] S. Lim Jae, *and* V. Alan, "All-pole modeling of degraded speech", *IEEE Transaction on ASSP*, Vol. 26, No. 2, June 1978.

[63] F. Takens, "In dynamical systems and turbulence", *Lecture notes in Math*, (SPRINGER, Berlin), 1981.

[64] N. S. Jayant, "Coding speech at low bit rates," *IEEE Spectrum*, pp. 58-63, August, 1986.

[65] T. Nishitani, S. Aikon, T. Araseki, K. Ozawa, R. Maruta, "A 32 kb/s toll quality ADPCM codec using a single chip signal processor," in *Int'l Conference on Acoustics, Speech, and Signal Processing*, (Paris, France), Vol 2, pp. 960-963, April, 1982.

[66] D. Cointot, "A 32 kb/s ADPCM coder robust to channel errors," in *Int'l Conference on Acoustics, Speech, and Signal Processing*, (Paris, France), Vol. 2, pp. 964-967, April, 1982.

[67] Carver Mead, "Neuromorphic electronic system", *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1629-1636, 1990.

[68] D. Millar, *and* P. Mermelstein, "Prevention of predictor mistracking in ADPCM coders", in *Int'l Conference on ICC'84*, 1984.

[69] D. W. Petr, "32 kb/s ADPCM-DLQ coding for network applications", *IEEE GLOBECOM'82 Conference Proceedings*, (Miami, Florida), Vol. 1, pp. 239-243, Nov. 1982.

[70] C. Richard, "The algorithm of 32 kb/s ADPCM", personal communication, 1994.

[71] N. Kitawaki, and H. Nagabuchi, "Quality assessment of speech coding and speech synthesis system", *IEEE Communications Magazine*, Vol. 26, pp. 36-44, 1988

[72] International Telecommunication Union, "5-, 4-, 3- and 2-bits sample embedded adaptive differential pulse code modulation (ADPCM)" *CCITT G.727*, Geneva, 1990.

[73] S. Haykin, *and* L. Li , "Modified kalman filtering", *IEEE Transaction on Signal Processing*, Vol. 42, pp. 1239-1242, May, 1994.