

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

10-10-1999

ARMA lattice modeling for isolated word speech recognition.

Tracy Xiaoping Li
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Li, Tracy Xiaoping, "ARMA lattice modeling for isolated word speech recognition." (1999). *Electronic Theses and Dissertations*. 6924.
<https://scholar.uwindsor.ca/etd/6924>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

**ARMA LATTICE MODELING
FOR ISOLATED WORD SPEECH RECOGNITION**

by

Tracy Xiaoping Li

A Thesis

Submitted to the College of Graduate Studies & Research
through the Faculty of Engineering - Electrical & Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

October, 1999



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*

Our file *Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52599-6

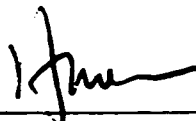
Canada

900226

© 1999 **Tracy Xiaoping Li**

All Rights Reserved. No part of this document may be reproduced, stored or otherwise retained in a retrieval system or transmitted in any form, on any medium or by any means without the prior written permission of the author.

Approved by:



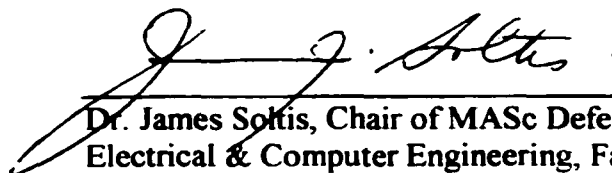
Dr. H. K. Kwan, Advisor,
Electrical & Computer Engineering, Faculty of Engineering



Prof. Philip H. Alexander, Internal Reader,
Electrical & Computer Engineering, Faculty of Engineering



Dr. Yung H. Tsin, External Reader,
School of Computer Science



Dr. James Soltis, Chair of MASc Defense,
Electrical & Computer Engineering, Faculty of Engineering

Abstract

In isolated word speech recognition system, it is important to develop an accurate model that estimates speech characteristics. The conventional speech modeling technique linear prediction (LP) method which is implemented by an all-pole Auto-Regressive (AR) model is found not good for all speech sounds. It can not handle the nasal sound, or phonemes that introduce zeros. The objective of this thesis is to apply a new Auto Regressive-Moving Average (ARMA) lattice model to speech feature extraction. To obtain the model parameters based on the Least Square Error (LSE) criterion, a lattice algorithm provides an efficient and robust approach. The reflection coefficients of the lattice model then are forwarded to pattern recognition stages which could be implemented by Vector Quantization (VQ) or Neural Network (NN) approaches. The experiments tested on an isolated word database indicate that the adopted lattice model provides significantly better performance on recognition accuracy than the LP model.

Dedicated with love
to my family.

Acknowledgments

I would like to express my sincere gratitude to my thesis advisor Dr. H. K. Kwan for his invaluable guidance, discussions and comments, and constant encouragement throughout the progress of this thesis. His insightful suggestion on the use of ARMA lattice modeling for isolated word speech recognition has been proved to be fruitful in the thesis. I would also like to thank him for going through the first and final drafts of the thesis in short notice and for making various suggestions to improve the presentation of the thesis. I would also like to thank my internal reader, Prof. P. H. Alexander, and my external reader, Dr. Y. H. Tsin from computer science for their suggestions, comments and advice for enhancing the thesis presentation.

I would like to sincerely thank all my friends, especially, Hui Ping, Wayne Chiang, Walter Jin and Jie Zhang for their encouragement and friendship in this journey. Thanks are also going to my dear fellows of the ISPLab and others, for their help to build a database for these experiments. Besides, special thank goes to Raymond Chan, who graduated from our laboratory two years ago, with his valuable work, save me much time on program writing.

Last but far from least, my very special thanks must be given to my family and particularly my parent for their wholehearted support, and for standing by me unconditionally.

Table of Contents

1	INTRODUCTION.....	1
1.1	SPEECH RECOGNITION	1
1.2	MODELING PROBLEMS.....	1
1.3	ABOUT THE THESIS.....	3
1.3.1	<i>The objectives</i>	3
1.3.2	<i>The organization</i>	3
2	SPEECH MODELING	5
2.1	WHAT IS SPEECH MODELING?	5
2.2	SPEECH PRODUCTION.....	6
2.2.1	<i>The mechanism of speech production</i>	6
2.2.2	<i>Principle components</i>	7
2.3	MODEL OF SPEECH WAVEFORM.....	8
2.3.1	<i>Tube model</i>	8
2.3.2	<i>LP model</i>	10
2.3.3	<i>Model breakdowns</i>	16
2.3.4	<i>Modifications on LP</i>	17
2.4	POLE-ZERO (ARMA) MODELING	18
2.4.1	<i>Introduction</i>	18
2.4.2	<i>Comparison between AR and ARMA modeling</i>	19
2.4.3	<i>Lattice method</i>	21
3	ARMA LATTICE MODELING ALGORITHM.....	25
3.1	INTRODUCTION	25
3.2	ARMA LATTICE MODELING.....	26
3.3	ARMA LATTICE ALGORITHM.....	29
3.3.1	<i>Introduction</i>	29

3.3.2	<i>Elementary modules</i>	29
3.3.3	<i>Definition of error fields</i>	32
3.3.4	<i>Starting blocks</i>	34
3.3.5	<i>Regular order update blocks</i>	35
3.3.6	<i>Order increments</i>	39
3.4	PROPERTIES OF LATTICE ALGORITHM	44
4	SPEECH RECOGNITION SYSTEM	47
4.1	INTRODUCTION	47
4.1.1	<i>Primary tasks</i>	47
4.1.2	<i>The data of speech recognition</i>	49
4.1.3	<i>Recognition approaches</i>	50
4.2	ISOLATED WORD SPEECH RECOGNITION SYSTEM	52
4.2.1	<i>Our task</i>	52
4.2.2	<i>Recognition system model</i>	54
4.3	PARAMETRIC REPRESENTATION	54
4.4	PATTERN TRAINING	58
4.5	PATTERN SIMILARITY DECISION	59
5	PATTERN RECOGNITION TECHNIQUES	61
5.1	INTRODUCTION	61
5.2	VECTOR QUANTIZATION	62
5.2.1	<i>Definition</i>	63
5.2.2	<i>Elements of VQ implementation</i>	64
5.2.3	<i>VQ training</i>	65
5.3	NEURAL NETWORK APPROACH	66
5.3.1	<i>Architecture</i>	66
5.3.2	<i>Learning algorithm</i>	67
5.3.3	<i>General considerations</i>	68

5.4	COMPARISON	69
6	EXPERIMENTS AND RESULTS.....	72
6.1	DATABASE COLLECTION.....	72
6.2	EXPERIMENTAL DESIGN.....	73
6.3	IMPLEMENTATION ISSUES	75
6.4	RESULTS AND DISCUSSIONS.....	76
6.4.1	<i>Speaker-dependent tests</i>	76
6.4.2	<i>Speaker-independent tests</i>	82
7	CONCLUSIONS	85
7.1	CONCLUSIONS.....	85
7.2	FUTURE WORK.....	87
8	REFERENCES.....	89
	APPENDIX	91

List of Tables

Table 3.1 Error fields	33
Table 3.2 Example: poles and zeros	41
Table 3.3 Numbers of lattice coefficients	43
Table 4.1 Vocabulary types and sizes	53
Table 5.1 Comparison between VQ and NN approach	69
Table 6.1 Results for speaker-independent recognition	76
Table 6.2 Results for speaker-independent recognition	82
Table 6.3 Recognition results for same feature vector dimension	84

List of Figures

Figure 2.1 (a) Human vocal system (b) functional components	6
Figure 2.2 (a) Schematic diagram of vocal system (b) Block diagram	9
Figure 2.3 Inverse filter $A(z)$ applied to (a) model response, (b) desired response	12
Figure 2.4 AR and ARMA modeling of speech sound /n/	20
Figure 2.5 Basic lattice structure	21
Figure 2.6 Illustration of modeling of speech sound /m/	24
Figure 3.1 Module A	30
Figure 3.2 Module B	31
Figure 3.3 Module blocks	32
Figure 3.4 I_AR	34
Figure 3.5 I_MA	35
Figure 3.6 AR_AR	36
Figure 3.7 MA_MA	37
Figure 3.8 AR_MA	38
Figure 3.9 MA_AR	39
Figure 3.10 ARMA lattice order increments	40
Figure 3.11 Minimal arrangement	41
Figure 4.1 Components of speech recognition application	48
Figure 4.2 Block diagram of recognition system	54
Figure 4.3 Block diagram of parametric representation	55
Figure 4.4 Blocking into overlapping frames	57
Figure 4.5 pattern training	59
Figure 4.6 Similarity decision system	59
Figure 5.1 Pattern recognition structure	62
Figure 5.2 Elements in VQ	64
Figure 5.3 Neural net architecture	67
Figure 6.1 Recognition using AR modeling	79

Figure 6.2 Recognition using ARMA lattice modeling	80
Figure 6.3 AR modeling vs ARMA lattice modeling	81
Figure 6.4 Speaker-dependent vs speaker-independent	83

List of Abbreviations

AR	Auto regressive
ARMA	Auto regressive moving average
LP	Linear prediction
LSE	Least square error
MA	Moving average
NN	Neural network
RLS	Recursive least squares
VQ	Vector quantization

Chapter 1

1 Introduction

1.1 *Speech Recognition*

Speech recognition consists of two major techniques: feature extraction and pattern recognition. The feature extraction attempts to discover characteristics of speech signals unique from one to others. Pattern recognition refers to the matching of features in such a way as to determine whether two sets of features are identical.

For extracting features, the speech waveforms have to be converted into some types of parametric representation: this is called speech modeling or analysis. It is important to develop an accurate model that estimates speech characteristics: it has large influence on the system ability.

1.2 *Modeling Problems*

In the past 30 years, the linear predictive (LP) technique [1] has been widely used in the area of speech modeling/analysis. By applying LP analysis to speech signals, we obtain

auto regressive (AR) all-pole digital filters. The basic idea behind linear predictive analysis is that a speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of the squared differences (over a finite interval) between the actual speech samples and linear predicted ones, a unique set of predictor coefficients can be determined. These coefficients are believed to contain the characteristics of speech and then could be used as the features information in recognition tasks. It is shown that such an all-pole type filter can be directly derived from an acoustic tube model of the vocal tract [2].

However in reality, the vocal tract model differs from the all-pole model in several respects. First, the LP analysis method can not handle the phonemes with zeros, e.g., nasals, if we take into account the nasal tract as a side branch, the mathematical representation of the model will include zeros in its transfer function. Secondly, those fricatives and plosives excited sounds are equal to have the posterior and anterior tubes in the vocal system: excitations between them introduce zeros. There may be no resonance, or resonance may be hidden by zeros. These sort of zero problems lead us to another modeling technique called auto regressive moving average (ARMA) modeling. This technique is designed for handling pole-zero type modeling problem: in other words, its transfer function includes not only poles but also zeros.

For ARMA modeling, a variety of techniques has been formulated theoretically to estimate the model parameters. However, the practical application of these techniques has been somewhat limited due to the relative complexity of the algorithms. After many years

efforts, some of the more recent work indicates that a relatively efficient ARMA modeling technique is now available [13][14]; it is called the lattice method. Based on a recursive structure, the lattice algorithm has many attractive properties, like numerical stability, computational efficiency and suitability for hardware implementation.

1.3 About The Thesis

1.3.1 The objectives

This thesis looks at ARMA modeling of speech with an efficient lattice algorithm. The work presented is trying to apply this lattice model to substitute for the conventional LP model in an isolated word speech recognition system, and therefore obtaining better system performance.

1.3.2 The organization

In Chapter 2, we discuss the speech modeling problem. An introduction is given on the acoustic model, the basic knowledge of all-pole (AR) and pole-zero (ARMA) modeling for the speech are also presented.

In Chapter 3, we give the detail of an efficient ARMA lattice algorithm. Based on geometric and projection theory, the ARMA modeling problem is solved by using recursive lattice structure. The formulae for computing lattice reflection coefficients are

given in this chapter. At the end of the chapter, the properties of the lattice algorithm are summarized.

In Chapter 4, we construct a system for isolated word speech recognition. It includes the step by step procedure for implementation.

In Chapter 5, the pattern recognition techniques are discussed - first introduce the concept of the popular Vector Quantization (VQ) techniques and then give the step by step algorithm of a Neural Network (NN) approach.

In Chapter 6, the experiments are designed for testing the pre-recorded database. The comparison results of using AR and ARMA lattice modeling methods are also presented.

In Chapter 7, the conclusions and some future directions on the subject are discussed.

Chapter2

2 Speech modeling

2.1 What is Speech Modeling?

Although the recognition systems vary a lot depend on their sub-components, the greatest common feature they share is perhaps named speech modeling or speech analysis. What is speech modeling? It is a process that converts speech waveform to some types of parametric representation. The objective of this technique is to develop an accurate model of the available time series speech data. This model then has certain spectral properties and is usually parametric, i.e. represented by a set of parameters.

Basically, modeling the speech sounds must take into consideration their method of production. They have to be based on the acoustic properties of sound production. In the recognition system, modeling plays important roles on the level of processing between the digitized acoustic waveform and the acoustic feature vectors. Their job is to extract distinct information from waveform.

2.2 Speech Production

2.2.1 The mechanism of speech production

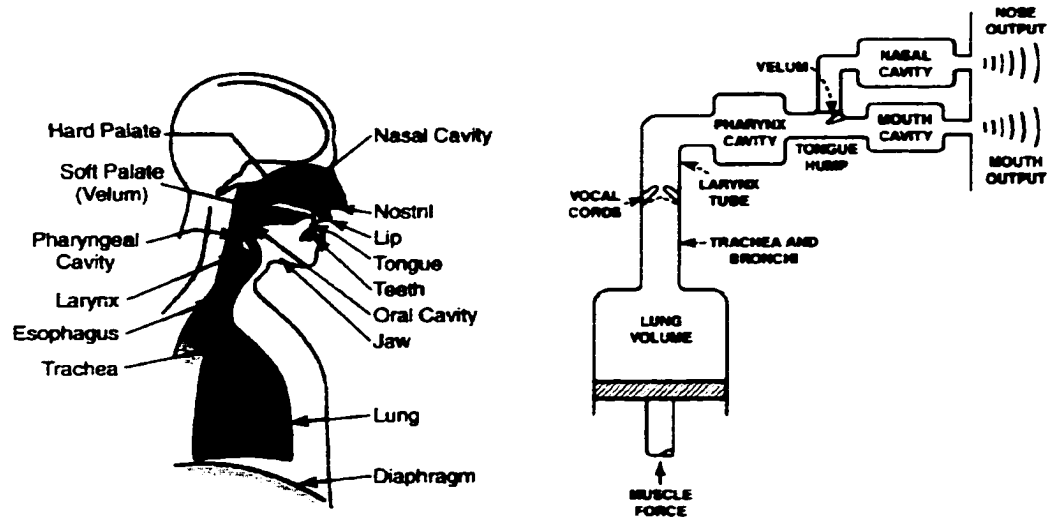


Figure 2.1 (a) Human vocal system (b) functional components

To study a model that is useful for speech processing, it is necessary to have knowledge about speech production. Figure 2.1(a) shows the human vocal system. Figure 2.1(b) shows a schematic of the functional components of the vocal system.

The process of producing speech sounds is as follows:

- lungs: fill with air
- contraction of rib cage forces air from the lungs into the trachea - the volume of air determines the amplitude of the sound
- trachea (windpipe): conveys air to the vocal tract. The vocal cords, at the top of the trachea, separate the trachea from the base of the vocal tract
- vocal tract

✓ consists of:

pharynx (throat)

mouth

nose

- ✓ the shape and size of the vocal tract vary by positioning the articulators: the tongue, teeth and lips
- ✓ the shape of the vocal tract determines the type of speech sound - e.g., the /a/ in "hat" vs the /i/ in "hit"

The figures place in evidence the important features of the human vocal system. The vocal tract begins at the opening between the vocal cords, or glottis, and end at lips. The vocal tract thus consists of the pharynx, the mouth and the nose. The tongue controls the vocal tract dividing it into two resonant cavities, the pharynx and mouth, which, in turn, determine the transmission characteristics of the vocal tract. The nasal cavity is an parallel with the mouth and can further modify the sound produced. The transmission characteristics of vocal tract is usually described by its resonant peaks in the spectrum known as formants, which shift in frequency as we changed the characteristics of the vocal tract.

2.2.2 Principle components

Two principle components of speech production are:

- A. Excitation - create a sound by setting the air in rapid motion
- B. Vocal tract - "shape" the sound

The excitation can be characterized as phonation, friction and plosive.

- ✓ **Phonation:** vibration of vocal cords. modes of vibration called resonance
- ✓ **Friction:** turbulent air flow
 - the excitation is set up by forcing air past a constriction at some point in the vocal tract
 - e.g., /f/ in "for": top teeth & bottom lip
 - e.g., /th/ in "thin": tongue & top teeth
 - can combine friction with phonation
 - e.g., /v/ as in "vote" top teeth & bottom lip as in /f/. combined with phonation
- ✓ **Plosive:** closure at some point in the vocal tract. followed by a release of air
 - e.g., /p/ as in "pot": closure at lips
 - can be combined with vocal cord vibration: phonation and plosive /b/ as in "boy": closure at lips closure as in /p/. combined with phonation

Speech produced by phonated excitation is called voiced; speech produced by phonated excitation plus friction is called mixed voiced; and speech produced by other types of excitation is called unvoiced.

2.3 Model of Speech waveform

2.3.1 Tube model

In studying the speech production process, it is helpful to abstract the important features of the physical system in a manner that leads to a realistic yet tractable mathematical

model. Figure 2.2(a) shows such a schematic diagram of the vocal system. For completeness the diagram includes the sub-glottal system composed of the lungs, bronchi and trachea. This sub-glottal system serves as a source of energy for the production of speech. Speech is simply the acoustic wave that is radiated from this system when air is expelled from the lungs and the resulting flow of air is perturbed by a constriction somewhere in the vocal tract.

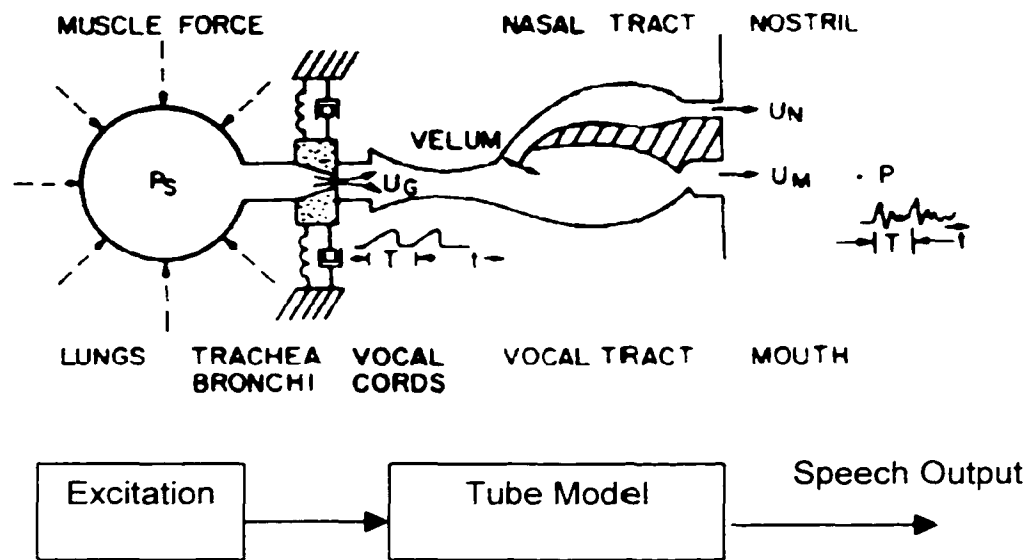


Figure 2.2 (a) Schematic diagram of vocal system (b) Block diagram

Figure 2.2 (b) shows a general block diagram that is representative of models that has been used as the basis for speech processing. These models all have in common that the excitation features are separated from the vocal tract features. The vocal tract is modeled as a uniform tube closed at the glottis end and open at the mouth end. It is accounted for by the time-varying linear system: the purpose is to model the resonance, the vibration of the vocal cord. The excitation generator creates a signal that is either a train of (glottal)

pulses, or randomly varying noise. The parameters of the source and the system are chosen so that the resulting output has the desired speech-like properties. In practice, this uniform tube model can be represented with an all-pole transfer function. In statistical terms, this kind of model which only has poles expressed by the denominator coefficients is also called an auto regressive (AR) model. The most famous AR modeling applied in speech processing area is based on linear prediction theory: so called LP model.

2.3.2 LP model

The all-pole LP model, as applied to speech, has been well understood for many years. The mathematical details and derivations will be omitted here: the interested reader is referred to the reference [1]. It is an analytically tractable model and provides good performance in many speech processing applications. Experience has shown that it also works well in some recognition applications. That is why it was so popular in the area of speech processing in the past 30 years.

The basic idea behind the LP model is that a given speech sample at time n , $y(n)$, can be approximated as a linear combination of the past N speech samples, such that

$$y(n) \approx a_1 y(n-1) + a_2 y(n-2) + \dots + a_N y(n-N) \quad (2.1)$$

where the coefficients a_1, a_2, \dots, a_N are assumed constant over the speech analysis frame.

We convert Eq. (2.1) to an equality by including an excitation term, $x(n)$, giving

$$y(n) = \sum_{k=1}^N a_k y(n-k) + x(n) \quad (2.2)$$

By expressing Eq. (2.2) in the z-domain we get the relation

$$Y(z) = \sum_{k=1}^N a_k z^{-k} Y(z) + X(z) \quad (2.3)$$

leading to the transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \sum_{k=1}^N a_k z^{-k}} = \frac{1}{A(z)} \quad (2.4)$$

We consider the linear combination of past speech samples as the estimated $\hat{y}(n)$, defined as

$$\hat{y}(n) = \sum_{k=1}^N a_k y(n-k) \quad (2.5)$$

We now form the prediction error, $e(n)$, defined as

$$e(n) = y(n) - \hat{y}(n) = y(n) - \sum_{k=1}^N a_k s(n-k) \quad (2.6)$$

The basic problem of linear prediction analysis is to determine the set of predictor coefficients, $\{a_k\}$, directly from the speech signal so that the spectral properties of the model match those of the speech waveform within the analysis window. By minimizing the sum of the squared differences (over a finite interval) between the actual speech samples and linear predicted ones, a unique set of predictor coefficients can be determined. There are many methods to determine these coefficients: the most commonly used include auto-correlation method, covariance method and lattice method.

Methods to determine coefficients

Consider the causal AR filter model.

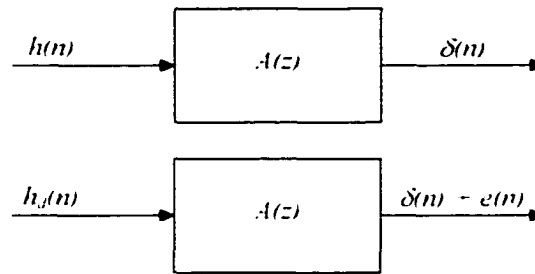


Figure 2.3 Inverse filter $A(z)$ applied to (a) model response, (b) desired response

$$H(z) = \frac{1}{A(z)} \quad (2.7)$$

$$\text{where } A(z) = a_0 + a_1 z^{-1} + \dots + a_N z^{-N} \quad (2.8)$$

then $H(z) A(z) = 1$, or, in time domain,

$$h(n) * a_n = \delta(n) \quad (2.9)$$

The FIR filter $A(z)$ is thus the inverse filter for $H(z)$, which whitens $h(n)$ to produce $\delta(n)$, as depicted in Figure 2.3(a). Since $A(z)$ has order N , we call it a support with N order. Using $2N - 1$ consecutive values for $h(n)$ beginning at $n = -N$, (2.9) provides $N - 1$ linear equations from which to solve for a_n ; i.e., given $h(0)$ through $h(N)$ with the fact that $h(n) = 0$, for $n < 0$, the convolution may be written in matrix form to produce the following $N - 1$ equations for a_n :

$$\begin{bmatrix} h_0 & 0 & \dots & 0 \\ h_1 & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_N & \dots & h_1 & h_0 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.10)$$

Nevertheless, given a desired or measured impulse response $h_d(n)$ that is not exactly AR and/or of greater than N^{th} order, we can only approximate it. Therefore, (2.9) becomes

$$h_d(n) * a_n = \delta(n) + e(n) \quad (2.11)$$

where $e(n)$ is the approximation error, and $A(z)$ is only an approximate inverse filter as illustrated in Figure 2.3(b). Similar to (2.9), we write (2.11) into matrix form; we have

$$\mathbf{H}_d \mathbf{a} = \boldsymbol{\delta} + \mathbf{e} \quad (2.12)$$

where

$$\mathbf{H}_d = \begin{bmatrix} h_{d0} & 0 & \cdots & 0 \\ h_{d1} & h_{d0} & & 0 \\ \vdots & & \ddots & \vdots \\ h_{dN} & h_{d,N-1} & \cdots & h_{d0} \\ \vdots & & & \vdots \\ h_{dL} & \cdots & \cdots & h_{d,N} \end{bmatrix}$$

$$\mathbf{a} = [a_0, a_1, \dots, a_N]^T$$

$$\boldsymbol{\delta} = [1, 0, \dots, 0]^T$$

$$\mathbf{e} = [e_0, e_1, \dots, e_N]^T$$

Note that we have assumed more data samples $h_d(n)$, for $n=0, 1, \dots, L$, than the minimum required (i.e., $L > N$) in order to approximate $h_d(n)$ closely over more than the minimum interval. Or, in other words, we can assume the available data for the approximation is up to N . Applying the least-square error criterion, we are about to minimize

$$E_e = \mathbf{e}^T \cdot \mathbf{e} = \sum_{n=0}^L e_n^2 \quad (2.13)$$

This is the standard problem in least-squares estimation with over-determined equations.

From (2.12), (2.13) becomes

$$\begin{aligned}
E_N &= (\mathbf{H}_d \mathbf{a} - \boldsymbol{\delta})^T (\mathbf{H}_d \mathbf{a} - \boldsymbol{\delta}) \\
&= (\mathbf{a}^T \mathbf{H}_d^T - \boldsymbol{\delta}^T) (\mathbf{H}_d \mathbf{a} - \boldsymbol{\delta}) \\
&= \mathbf{a}^T \mathbf{H}_d^T \mathbf{H}_d \mathbf{a} - 2 \mathbf{a}^T \mathbf{H}_d^T \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{\delta}
\end{aligned} \tag{2.14}$$

To minimize E_N , we put the corresponding partial derivatives to zero: i.e. in vector form.

$$\frac{\partial E_N}{\partial \mathbf{a}} = 0 \tag{2.15}$$

which gives

$$2 \mathbf{H}_d^T \mathbf{H}_d \mathbf{a} - 2 \mathbf{H}_d^T \boldsymbol{\delta} = 0 \tag{2.16}$$

or simply

$$\mathbf{H}_d^T \mathbf{H}_d \mathbf{a} = \mathbf{H}_d^T \boldsymbol{\delta} \tag{2.17}$$

Equation (2.17) is the desired normal equation for the LSE solution \mathbf{a} . Note that if we pre-multiply both sides of (2.12) by \mathbf{H}_d^T , we have

$$\mathbf{H}_d^T \mathbf{H}_d \mathbf{a} = \mathbf{H}_d^T \boldsymbol{\delta} + \mathbf{H}_d^T \mathbf{e} \tag{2.18}$$

Comparing (2.17) and (2.18), it implies

$$\mathbf{H}_d^T \mathbf{e} = 0 \tag{2.19}$$

Covariance method

If we denote

$$\Phi_N = \mathbf{H}_d^T \mathbf{H}_d \tag{2.20}$$

and note also that

$$\mathbf{H}_d^T \boldsymbol{\delta} = h_{d0} \boldsymbol{\delta} \tag{2.21}$$

Equation (2.17) becomes

$$\Phi_N \mathbf{a} = h_{d0} \delta \quad (2.22)$$

where Φ_N is the $(N+1) \times (N+1)$ symmetric covariance matrix. The parameters \mathbf{a} can be obtained by (2.22).

$$\mathbf{a} = h_{d0} \Phi_N^{-1} \delta \quad (2.23)$$

Yule-Walker Equation

As $L \rightarrow \infty$, the elements ϕ_{ij} of Φ_N approach the auto-correlation values $R(i-j)$ where

$$R(m) = \sum_{n=0}^{\infty} h_d(n) \cdot h_d(n+m) \quad (2.24)$$

Since $h_d(n)$ is real, $R(-m)=R(m)$. Therefore, Φ_N becomes the symmetric Toeplitz auto-correlation matrix:

$$R_N = \begin{bmatrix} R_0 & R_1 & \dots & R_N \\ R_{-1} & R_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_1 \\ R_{-N} & \dots & R_{-1} & R_0 \end{bmatrix} \quad (2.25)$$

Equation (2.22) becomes the famous Yule-Walker equation [4]

$$\mathbf{R}_N \mathbf{a} = h_{d0} \delta \quad (2.26)$$

And this leads to another famous method called autocorrelation method.

Both the covariance and autocorrelation methods consist of two steps: computation of a matrix of correlation values and solution of a set of linear equations. A variety of techniques can be applied to solve those linear equations in an efficient manner such as

the Cholesky decomposition solution for the covariance method, and Durbin's recursive solution for the autocorrelation equations. Another class of methods called lattice method has evolved in which the above two steps have in a sense been combined into a recursive algorithm for determining the linear predictor parameters. Once these coefficients obtained, they are forwarded to the pattern recognition stage. Since the model is derived from the vocal tract system, those coefficients are supposed to reflect the features of the resulting speech signals. LP model is believed match the signal spectrum as long as the model order is large enough. The advantage of using this model is it is easy for implementation, however, we wouldn't expect it to work well on all kinds of phonemes, e.g. nasals, or phonemes with zero properties.

2.3.3 Model breakdowns

However in reality, the vocal tract model differs from an all-pole AR model in several respects. The most crucial and well-known shortcoming of an all-pole model is in its assumption that during any voiced pronunciation the velum is always closed and the sound wave proceeds only through the oral cavity. So the influence of the nasal cavity is ignored in the assumption. It raises no big problem when non-nasal sounds are processed, but in the case of nasal sounds the mismatch of the model becomes severe. Since the nasal cavity plays the role of a resonance cavity during the nasal pronunciation, there appear some zeros in the transfer function. The zeros have the effect of suppressing the peaks in mid-frequency, flattening spectrum there. But such a flat spectrum of nasal sounds is not easily fitted by a finite number of poles of the existing all-pole modeling [1].

Besides, unlike phonation, the places of fricative, plosive and other excitation are actually inside the vocal tract itself. This could cause difficulties for models that assume an excitation at the bottom end of the vocal tract. In fact, those fricatives and plosives sounds generated somewhere inside the vocal tract (thus dividing the vocal tract into the front and back portions), introduce zeros. There may be no resonance, or the resonance may be hidden by zeros.

Therefore it is necessary to modify the all-pole transfer function into a pole-zero type which call for both poles and zero in transfer function so that to handle these zero problem missed by the all-pole model. For more general speech analysis, as carried out, for example, within speech recognition, a variety of important cues is provided by information about spectral zeros.

2.3.4 Modifications on LP

At the beginning of attempts to apply a pole-zero model, many methods were still developed based on the original all-pole model: only small modifications were made for some special purposes: this is to avoid bringing in more computation burden, so the parameters for the zero part somehow can be obtained easily from the known parameters of the poles [5]. In the meantime, there are other researchers developed a new acoustic model taking into account the nasal tract as well as the oral tract and finally led to a pole-zero transfer function [6].

However, at present, more and more researchers focus on directly applying pole-zero analysis method to speech signals. 20 years ago, this seemed not a feasible choice due to the huge computation on the determination of parameters for both poles and zeros. But at present, except for the standard methods for solving the parameters, some other more efficient and powerful techniques have been developed and used in this area. Other encouragement comes from the big improvement on the technology of high speed computers. All these facts made pole-zero modeling eventually become a competitive technique for speech analysis.

2.4 Pole-Zero (ARMA) Modeling

2.4.1 Introduction

In the previous discussing of basic model for speech production and the characteristics of the vocal tract, we commented that the vocal tract could be reasonably approximated in terms of a rational transfer function represented by poles and zeros. The poles corresponding to the vocal tract resonance and the zeros introduced due to such effects as coarticulation and coupling between the vocal tract and nasal cavity.

The general transfer function of a pole-zero model is as follows:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^q b_k z^{-k}}{1 - \sum_{k=1}^p a_k z^{-k}} \quad (2.27)$$

where a and b denote the model parameters corresponding to poles and zeros respectively.

Like all-pole AR models, the pole-zero model is called the autoregressive moving average (ARMA) model in statistical term. This is more generally used for a modeling problem other than a signal processing problem. In fact, all different types of model can be derived from Eq. (2.27), depending on the nature of the numerator and denominator coefficients. Models for which the coefficients a are zero are called moving average (MA) models. Those for which the b coefficients are zeros are called autoregressive (AR) models. They are the special cases of the ARMA models. Because modeling the speech signal is equivalent to applying a digital filter in the signal processing, they can be called all-zero (MA) filters, all-pole (AR) filters and pole-zero (ARMA) filters too. For the purpose of simplicity, we will only use term MA, AR and ARMA in the following parts.

2.4.2 Comparison between AR and ARMA modeling

Figure 2.4 shows the comparison between all-pole modeling and pole-zero modeling for nasal sound /n/ in time domain. Here are a 12th order AR model and a 6/6 ARMA model: standard methods, autocorrelation and Prony, are used to determine the parameters respectively. The sum of the squared error between the real signal and the modeled signal of are 3.8309 for AR modeling, and 0.7963 for ARMA modeling (35 samples).

In the figure, it is easy to see that the ARMA modeled signal is closer to the original one, that is to say that the ARMA model is more accurate than the AR model in representing

the signal. Similar results can be found for other types of sound; of course error differences might not be that big like the sounds with zero properties. However, overall, it again verifies that ARMA model is the more generally applicable model for characterizing the speech signal, if one does not consider the difficulty of obtaining the model coefficients.

The practical application of ARMA modeling techniques has been somewhat limited due, perhaps, to the relative complexity of the model fitting algorithm. This is a bottleneck problem of using ARMA model. Fortunately, more recent research work indicates that the lattice algorithm, an efficient technique available for ARMA modeling, could be one feasible choices.

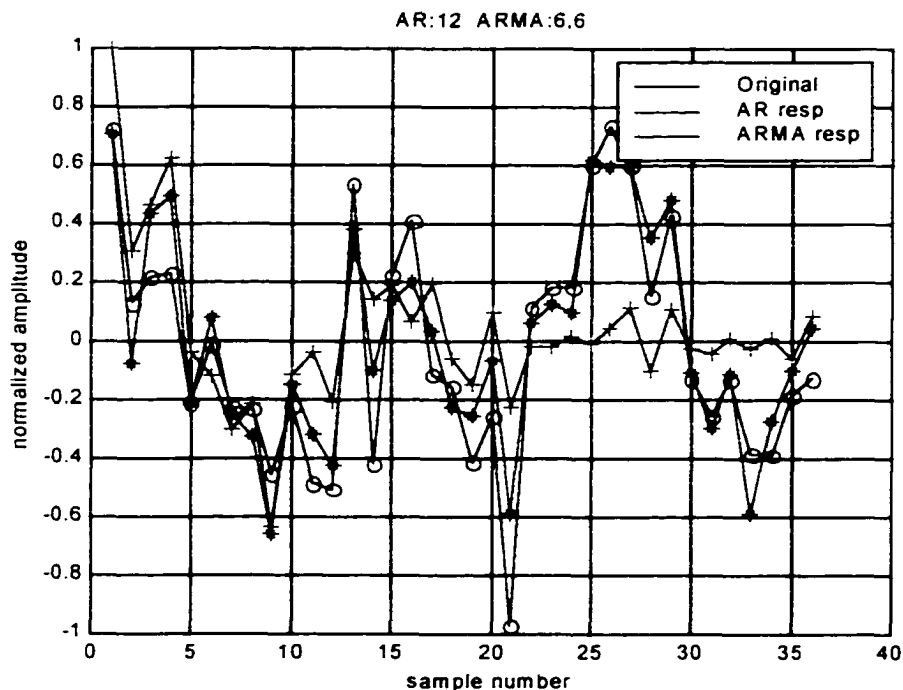


Figure 2.4 AR and ARMA modeling of speech sound /n/

processing type and based on the Levinson algorithm which is related to stationary covariance matrices [1]. The computation is performed in two steps. First, the sample covariance matrix of the data is computed, using various types of windowing. Then, reflection coefficients are computed by cross-correlating the forward and backward errors propagating in the lattice [8]. When the sample covariance matrix has a Toeplitz structure (the so-called autocorrelation method), the resulting lattice model is the optimal least-squares predictor of the observed process.

Recently, new lattice structures were developed for implementing the least-square prediction [9][10]: the algorithms are capable of computing reflection coefficients directly from the data. It is recursive both in time and in model order. The reflection coefficients are updated as each new data point becomes available. The recursive nature of the algorithm makes it ideally suited for real-time estimation.

The use of the lattice form realizations in linear system modeling problems has played an increasingly important role in many areas of application, for example, in speech processing, spectral estimation, and system identification. The many nice structural properties of lattice forms such as modular structure, good control of numerical stability, and much reduced computational complexity of modeling algorithm make them attractive candidates for hardware implementation.

Here is the comparison of using the lattice technique and the direct-form AR and ARMA modeling for speech signal. Figure 2.6(a) depicts the spectrum of the nature nasalized constant /m/.

Two strong antiresonances are clearly evident, one is at 650 Hz and the other at 3.2 KHz. Figure 2.6(b) shows the spectrum resulting from a 12th order AR modeling.. It is evident that this spectrum accurately reflects the presence of the resonances but not the antiresonances. In general, increasing the order of the AR modeling will not improve the modeling of the antiresonances [6].

Figure 2.6(c) and (d) represent the spectrum obtained from an ARMA (12,10) model. In Figure 2.6(c), the model is obtained using direct-form Prony's method [7] and in Figure 2.6(d), the model is obtained using a RLS based lattice algorithm [7]. In comparing the results for these two methods, it seems clear that the use of RLS based lattice modeling leads to more accurately characterized the antiresonances.

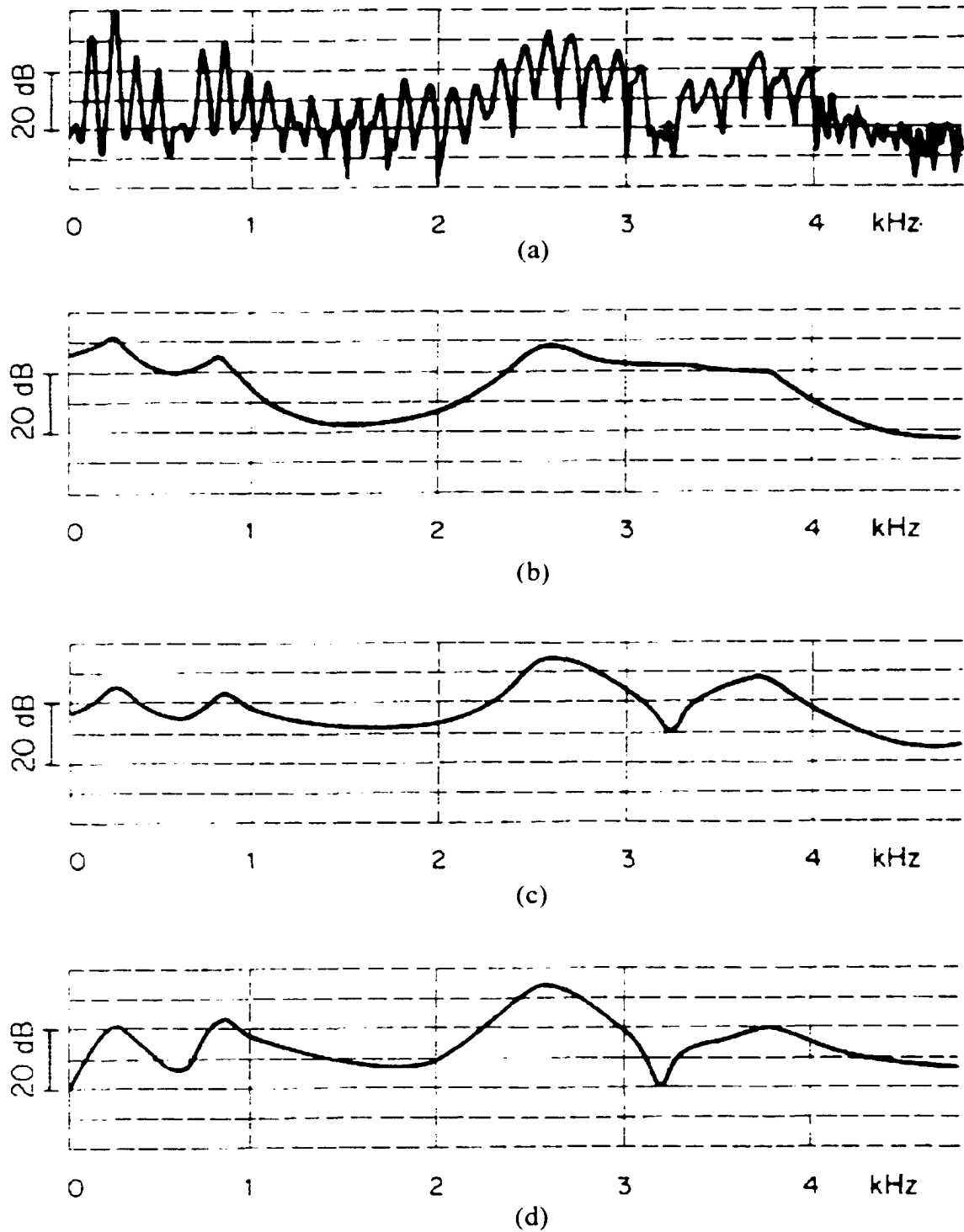


Figure 2.6 Illustration of modeling of speech sound /m/

Chapter 3

3 ARMA Lattice Modeling Algorithm

3.1 Introduction

The earliest form of ARMA lattice modeling algorithm was proposed by Lee, Friedlander and Morf [11] who formulated the algorithm based on geometric approach and projection approach. A similar approach was used by Parker and Lim [12] to develop a multi-channel AR lattice algorithm. It is shown that the two-channel case can be adapted for the purpose of ARMA system modeling. However, both algorithms are restricted to the case where the numerator and denominator polynomials of the optimal model are of the same order. In attempts to remove such restriction, Miyanaga, Nagai and Miki [10] developed a lattice algorithm which allows arbitrary arrangement of AR type and MA type update recursions. Further improvements in this direction are made by Kwan and Lui [13] [14], who have derived the algorithm which exhibits a high degree of simplicity and regularity in computational architecture and made it a competitive candidate for implementation.

3.2 ARMA Lattice Modeling

A linear causal time varying ARMA model may be characterized by an equation which expresses the model output at time n , $y_n(n)$, as a linear combination of the previous outputs $y(n-1), \dots, y(n-p)$ and the current and previous inputs $x(n), \dots, x(n-q)$, i.e.,

$$y(n) = \sum_{k=1}^p a_k y(n-k) + \sum_{k=0}^q b_k x(n-k) \quad (3.1)$$

Implementing the model corresponding to this difference equation leads to a transversal filter realization [9], where the a_k and the b_k are the filter coefficients. For lattice realization, $y_n(n)$ is expressed as linear combination of some auxiliary variables which can be expressed as a linear combination of the variables $y(n-1), \dots, y(n-p)$ and $x(n), \dots, x(n-q)$. Although these structures would have different sets of filter coefficients, they would all be equivalent in the sense that for a given input they would generate the same output (assuming no quantization or round off errors).

In the ARMA modeling problem, we are given an input sequence $\{x(n)\}$ and an output sequence $\{y(n)\}$ and we would like to find a linear time varying ARMA filter (determine the filter coefficients), whose output $\{\hat{y}(n)\}$ is as close as possible to $\{y(n)\}$.

In a speech application, the systems we are trying to model are slowly varying in time so that the ARMA filter coefficients may be considered to be constant over any interval of length L . In this case, the modeling problem at time n may be formulated as follows. At each time i within the interval $[n-L+1, n]$, an ARMA model with constant coefficients generates an output according to the difference equation

$$\hat{y}_n(i) = \sum_{k=1}^p a_{n,k} y(i-k) + \sum_{k=0}^q b_{n,k} x(i-k) \quad (3.2)$$

Our goal then is to find the set of filter coefficients which minimize the mean square error over the interval $[n-L+1, n]$, i.e., minimize

$$MSE(n) = \sum_{i=n-(L-1)}^n |e_n(i)|^2 = \sum_{i=n-(L-1)}^n |y(i) - \hat{y}_n(i)|^2 \quad (3.3)$$

Note that if we define the L-dimensional error vector e_n as

$$e_n = [e_n(n), \dots, e_n(n-L+1)]^T \quad (3.4)$$

then the mean square error, $MSE(n)$, can be viewed as the Euclidean norm of the error vector, e_n , in an L dimensional Hilbert space. If we also define the vectors

$$\begin{aligned} y_n &= [y(n), \dots, y(n-L+1)]^T \\ \hat{y}_n &= [\hat{y}_n(n), \dots, \hat{y}_n(n-L+1)]^T \\ x_n &= [x(n), \dots, x(n-L+1)]^T \end{aligned} \quad (3.5)$$

then we can write the error vector as

$$e_n = y_n - \hat{y}_n = y_n - \left(\sum_{k=1}^p a_{n,k} y_{n-k} + \sum_{k=0}^q b_{n,k} x_{n-k} \right) \quad (3.6)$$

Since \hat{y}_n is a linear combination of the vectors y_{n-1}, \dots, y_{n-p} and x_n, \dots, x_{n-q} then it lies in the subspace generated by these vectors:

$$\hat{y}_n \in \mathfrak{R}_{n,p,q} = Sp\{y_{n-1}, \dots, y_{n-p}, x_n, \dots, x_{n-q}\} \quad (3.7)$$

It is well known that the error vector e_n is minimized when \hat{y}_n equals the projection of y_n onto the subspace $\mathfrak{R}_{n,p,q}$ [1]. If we denote this projection by $P_{n,p,q}(y_n)$ then

$$\hat{y}_n = P_{n,p,q}(y_n) = \sum_{k=1}^p a_{n,k} y_{n-k} + \sum_{k=0}^q b_{n,k} x_{n-k} \quad (3.8)$$

where $a_{n,k}$ for $k=1, \dots, p$ and $b_{n,k}$ for $k=0, \dots, q$ are the projection coefficients. It is important to keep in mind of the fact that $P_{n,p,q}(y_n)$ is the minimum error norm solution, independent of the basis selected for $\mathfrak{R}_{n,p,q}$. This projection, however, may be realized or evaluated in many different ways depending upon the choice of basis for $\mathfrak{R}_{n,p,q}$.

The AR modeling problem is a special case of the ARMA modeling problem where $q=0$. In this case the mean square error is minimized with the projection

$$\hat{y}_n = P_{n,p}(y_n) \text{ the projection of } y_n \text{ onto } \mathfrak{R}_{n,p} \quad (3.9)$$

where $\mathfrak{R}_{n,p} = \text{Sp}\{y_{n-1}, \dots, y_{n-p}\}$.

In light of this geometric interpretation of the modeling problem, it has been shown that the ARMA lattice filter is characterized by two design rules [2][3], namely

- 1) It is realized in terms of an order increasing orthogonal basis of $\mathfrak{R}_{n,p}$, $r_{n,0}, \dots, r_{n,p-1}$, which is generated by the Gram-Schmidt orthogonalization procedure: $r_{n,i} = y_{n-i+1} - P_{n,i}(y_{n-i+1})$
- 2) It evaluates the projection error $e_{n,p}$ rather than the projection $P_{n,p}(y_n)$ and it does so in an order increasing manner, i.e. it first evaluates $e_{n,1}$, then $e_{n,2}$, etc. up to $e_{n,p}$.

An ARMA lattice filter was developed in [13][14] for ARMA modeling problem. The ARMA lattice filter was derived by defining six prediction errors. A summary of the algorithm is given in the next section.

3.3 ARMA Lattice algorithm [13][14]

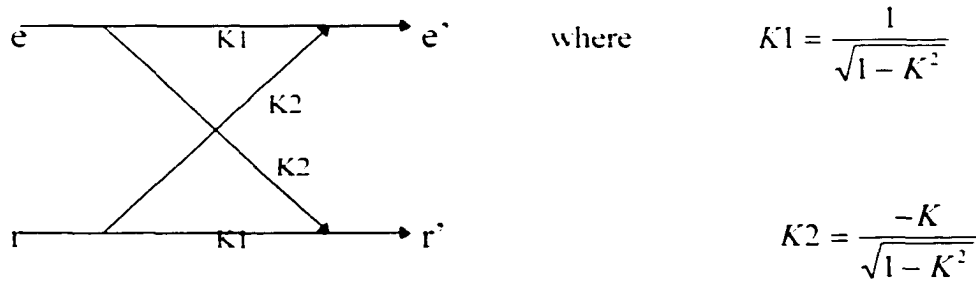
3.3.1 Introduction

The algorithm adopted here is an ARMA lattice analysis method for signals. To obtain order update recursions for a lattice model, two elementary modules and two starting blocks and four regular blocks are formed. Reflection coefficients inside each lattice stage can be calculated as model parameters.

3.3.2 Elementary modules

Module A

The structure of module A is shown in Figure 3.1. e and r are forward error and backward error respectively. K_1 and K_2 are reflection coefficients. they can be expressed by a single coefficient K according to the equations listed below. The relations between the inputs and the outputs of the module A then can be reflected by this single coefficient K .

**Figure 3.1 Module A**

In other words,

$$\begin{bmatrix} e' \\ r' \end{bmatrix} = \frac{1}{\sqrt{1-K^2}} \begin{bmatrix} 1 & -K \\ -K & 1 \end{bmatrix} \begin{bmatrix} e \\ r \end{bmatrix} \quad (3.10)$$

K is given by $K = \langle e, r \rangle$. It is the dot product, also known as correlation, between vector e and r . The correlation estimation formula used here is given by:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ \cdot \\ e_t \end{bmatrix} \quad \text{and} \quad r = \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ r_t \end{bmatrix} \quad (3.11)$$

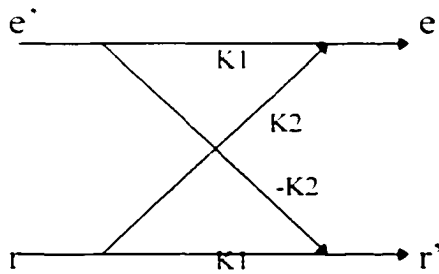
then,

$$\langle e, r \rangle = \frac{1}{L} \cdot \sum_{i=1}^L e_i \cdot r_i \quad (3.12)$$

The value of L is taken the length of the frame. Once the module is defined, with given inputs, coefficient K and the outputs can be obtained.

Module B

After some algebraic manipulation, with path of $\mathbf{e}-\mathbf{e}'$ revised direction, module A will become module B which takes input \mathbf{e}' and \mathbf{r} and returns \mathbf{e} and \mathbf{r}' . And the definition is shown in Figure 3.2



$$\text{where } K1 = \sqrt{1 - K^2} .$$

$$\text{and } K2 = K .$$

Figure 3.2 Module B

Same as module A, instead of using $K1$ and $K2$, a single reflection coefficient K is given by:

$$\frac{K}{\sqrt{1 - K^2}} = \frac{\langle r, u \rangle}{\langle e', u \rangle} \quad (3.13)$$

where \mathbf{u} is related to \mathbf{e}' by the fact that \mathbf{e}' is the error associated with the prediction of the random variable \mathbf{u} .

The operation of module B is described by the following equation:

$$\begin{bmatrix} e \\ r' \end{bmatrix} = \begin{bmatrix} \sqrt{1 - K^2} & K \\ -K & \sqrt{1 - K^2} \end{bmatrix} \cdot \begin{bmatrix} e' \\ r \end{bmatrix} \quad (3.5)$$

and the calculation of the reflection coefficient is

$$\text{from } \frac{K}{\sqrt{1-K^2}} = \frac{\langle r, u \rangle}{\langle e', u \rangle}, \quad \text{we get } K = \text{sign}\left(\frac{\langle r, u \rangle}{\langle e', u \rangle}\right) \cdot \sqrt{\frac{\left(\frac{\langle r, u \rangle}{\langle e', u \rangle}\right)^2}{1 + \left(\frac{\langle r, u \rangle}{\langle e', u \rangle}\right)^2}}$$

These two modules will be used as elementary blocks to construct other regular blocks. For simplicity, the modules will be treated as blocks with inputs and outputs shown below:

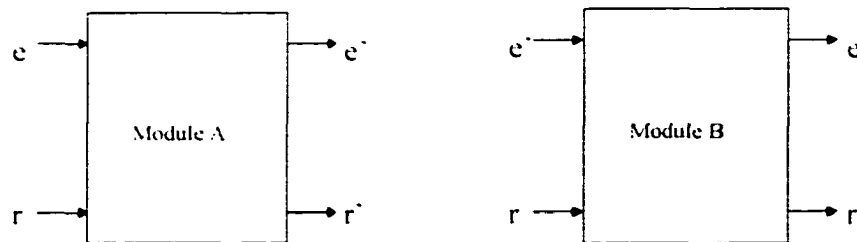


Figure 3.3 Module blocks

3.3.3 Definition of error fields

Before we introduce the starting blocks and regular blocks, we have to give the definition of error fields which the lattice algorithm operates on. They are inputs and outputs of each block. The lattice blocks involved in this algorithm manipulate 4 out of 6 error fields. They are listed in the following Table 3.1.

Table 3.1 Error fields

Symbol	Definition
e_x	forward error of x
ϵ_y	extended forward error of y
τ_x	extended backward error of x
r_y	backward error of y
r_x	backward error of x
τ_y	extended backward error of y

The error fields in the table form two groups, one for AR order update, the other for MA order update, each one consists of 4 error fields, i.e.,

$$\begin{pmatrix} e_x \\ \epsilon_y \\ \tau_x \\ r_y \end{pmatrix} \text{ for AR lattice like AR_AR, AR_MA.}$$

$$\begin{pmatrix} e_x \\ \epsilon_y \\ r_x \\ \tau_y \end{pmatrix} \text{ for MA lattice like MA_AR, MA_MA.}$$

The first two are the same for both sets: the last two will be changed in outputs when the order update needs to shift from AR to MA or from MA to AR.

3.3.4 Starting blocks

There are two starting blocks, namely I_AR and I_MA , respectively for AR and MA order update in the next stage.

I_AR

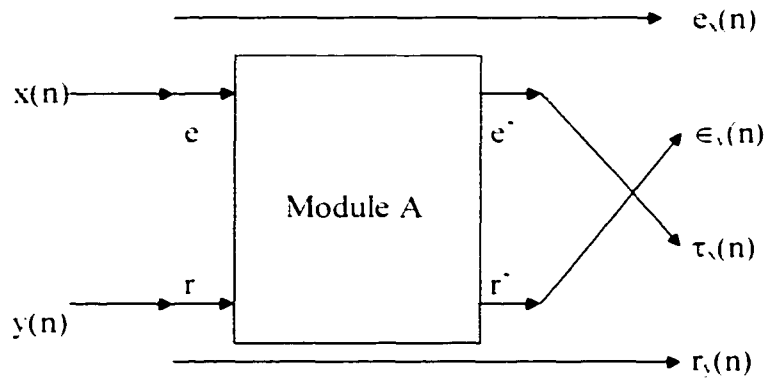


Figure 3.4 I_AR

Look at the diagram of block I_AR , it is constructed by a module A. The output error fields are for the AR order update in the next stage. The reflection coefficients of this block is simply the coefficients of module A. With given inputs of $x(n)$ and $y(n)$, the reflection coefficients and the error fields can be calculated by equations (3.10) and (3.12) and (3.13).

I_MA

Similar to I_AR , the starting block I_MA is also constructed by a module A. However its output error fields are for MA order update: one could see the last two errors are changed.

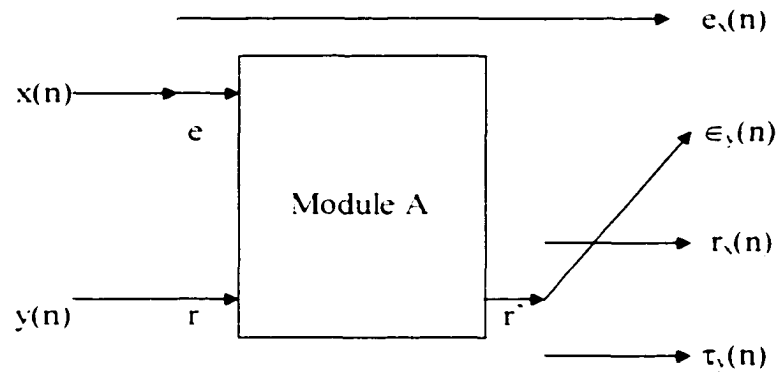


Figure 3.5 I_MA

3.3.5 Regular order update blocks

Once we have the first block of the lattice model to be either I_AR or I_MA, the next task is to connect it with other order update blocks for AR or MA order increment until we have got the model order that we want. In this algorithm, there are four such blocks are developed, namely, AR_AR, MA_MA, AR_MA, MA_AR. The first part of the names stands for the input error fields for order update, and the second part is for the output error fields for connection of the next stage; e.g., the block AR_AR has the input error fields of AR order update type: after this block, AR order will be increased by 1, and since the output error fields of this block are for AR type too, the next stage that can be connected must have the identical input error fields, say AR_AR or AR_MA.

AR_AR

The block diagram of AR_AR is shown in Figure 3.6. The error field on the left side is of order p (for AR order), q (for MA order). After this lattice block, the AR order will increase, i.e. $p \rightarrow p+1$. But for simplicity, the notation is not shown.

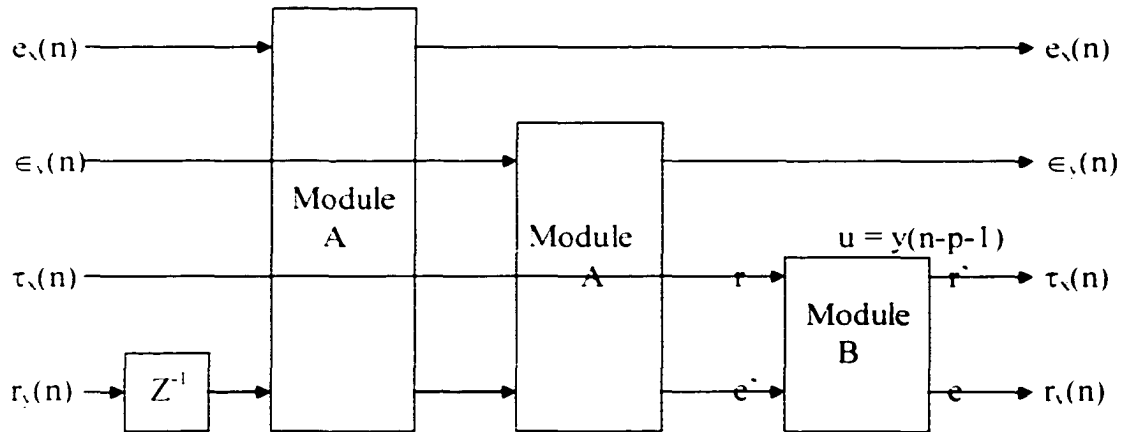


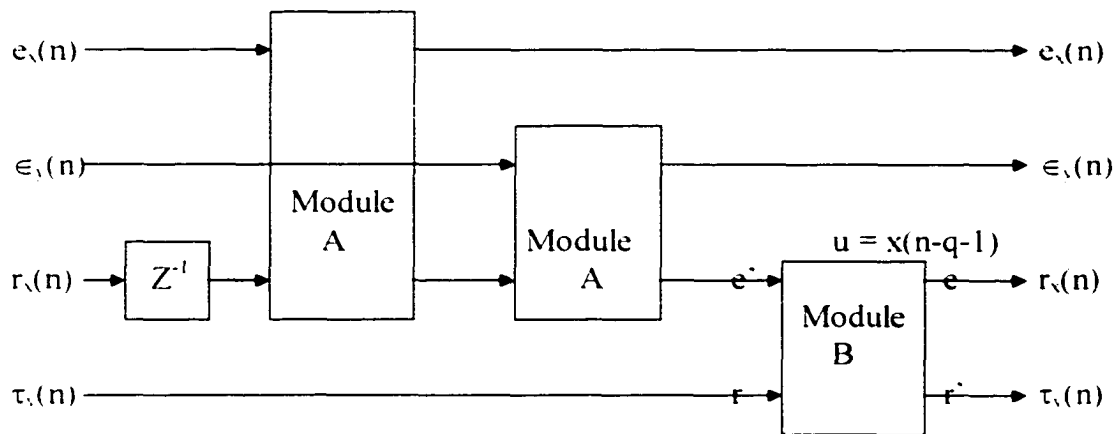
Figure 3.6 AR_AR

- * note that Module B is upside down.
- ** Module A is symmetrical, it could be upside down as long as the signal paths are not twisted.
- *** u is required in calculating the reflection coefficient.

The left side is $\begin{pmatrix} e_v^{(p,q)} \\ \epsilon_v^{(p,q)} \\ \tau_v^{(p,q)} \\ r_v^{(p,q)} \end{pmatrix}$ while the right side is $\begin{pmatrix} e_v^{(p+1,q)} \\ \epsilon_v^{(p+1,q)} \\ \tau_v^{(p+1,q)} \\ r_v^{(p+1,q)} \end{pmatrix}$

MA_MA

Shown below, the block diagram of block MA_MA in Figure 3.7, the same as AR_AR: the error field on the left side is of order p, q . After this lattice block, the MA order will increase, i.e. $q \rightarrow q+1$.

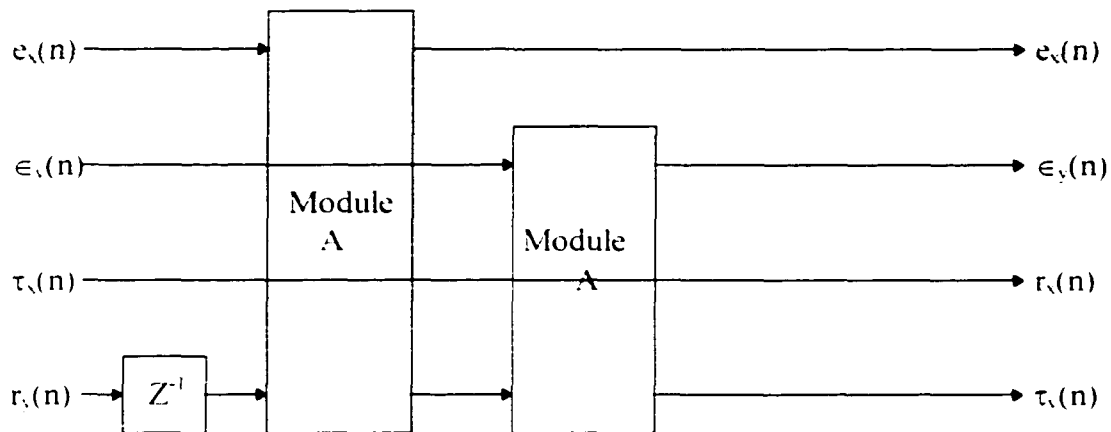
**Figure 3.7 MA_MA**

- * note that Module B is NOT upside down in this MA_MA block.
- ** Module A is symmetrical, it could be upside down as long as the signal paths are not twisted.
- *** u is required in calculating the reflection coefficient.

The left side is $\begin{pmatrix} e_v^{(p,q)} \\ \epsilon_v^{(p,q)} \\ r_v^{(p,q)} \\ \tau_v^{(p,q)} \end{pmatrix}$ while the right side is $\begin{pmatrix} e_v^{(p,q+1)} \\ \epsilon_v^{(p,q+1)} \\ r_v^{(p,q+1)} \\ \tau_v^{(p,q+1)} \end{pmatrix}$

AR_MA

In this lattice block, AR_MA, shown below in Figure 3.8, the output error fields are changed to **MA** format. The error field on the left side is of order p, q . After this lattice block, the **AR** order will increase, i.e. $p \rightarrow p+1$. And the 3rd and 4th output error fields are changed in meaning.

**Figure 3.8 AR_MA**

The left side is $\begin{pmatrix} e_v^{(p,q)} \\ e_v^{(p,q)} \\ \tau_v^{(p,q)} \\ r_v^{(p,q)} \end{pmatrix}$ while the right side is $\begin{pmatrix} e_v^{(p+1,q)} \\ e_v^{(p+1,q)} \\ r_v^{(p+1,q)} \\ \tau_v^{(p+1,q)} \end{pmatrix}$

MA_AR

The last block in this lattice algorithm is called MA_AR, its block diagram is shown in Figure 3.9. Again the error field on the left side is of order p, q . After this lattice block,

the MA order will increase, i.e. $q \rightarrow q+1$. The output error fields have changed to AR compatible format.

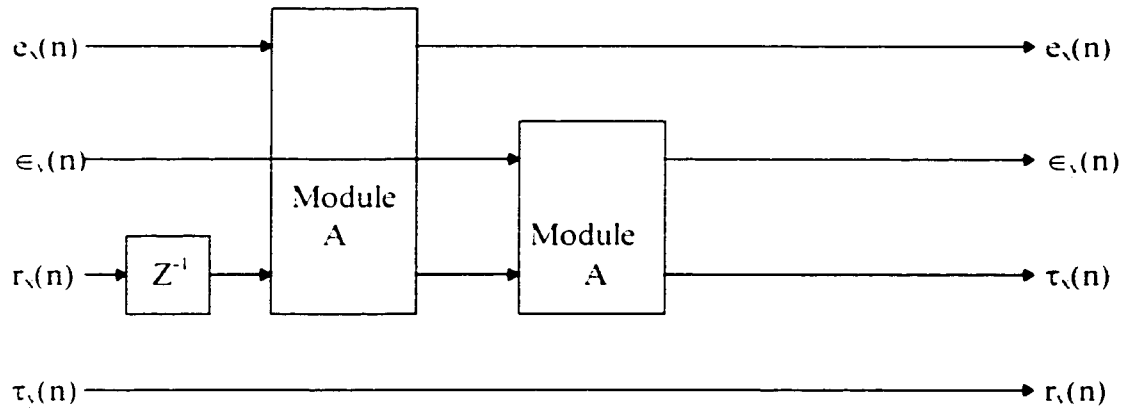


Figure 3.9 MA_AR

The left side is $\begin{pmatrix} e_v^{(p,q)} \\ \epsilon_v^{(p,q)} \\ r_v^{(p,q)} \\ \tau_v^{(p,q)} \end{pmatrix}$ while the right side is $\begin{pmatrix} e_v^{(p,q+1)} \\ \epsilon_v^{(p,q+1)} \\ \tau_v^{(p,q+1)} \\ r_v^{(p,q+1)} \end{pmatrix}$

3.3.6 Order increments

To build an ARMA lattice model, as we mentioned before, it should always start from one of the starting blocks. Other regular order update blocks are then connected according to the rules of error fields compatibility. The following figure shows the “micro” structure of an ARMA lattice model:

Basically, ARMA model with any orders can be obtained in such a way as long as the inputs and outputs of adjacent blocks are identical. e.g. starts from I_AR, so the next block's input must in same format of the I_AR's output. That is to say, either AR_MA or AR_AR can follow the I_AR block. Same thing happens for all the blocks here. If the second block is AR_MA, then the third one must be MA_AR or MA_MA, then one by one to continue.

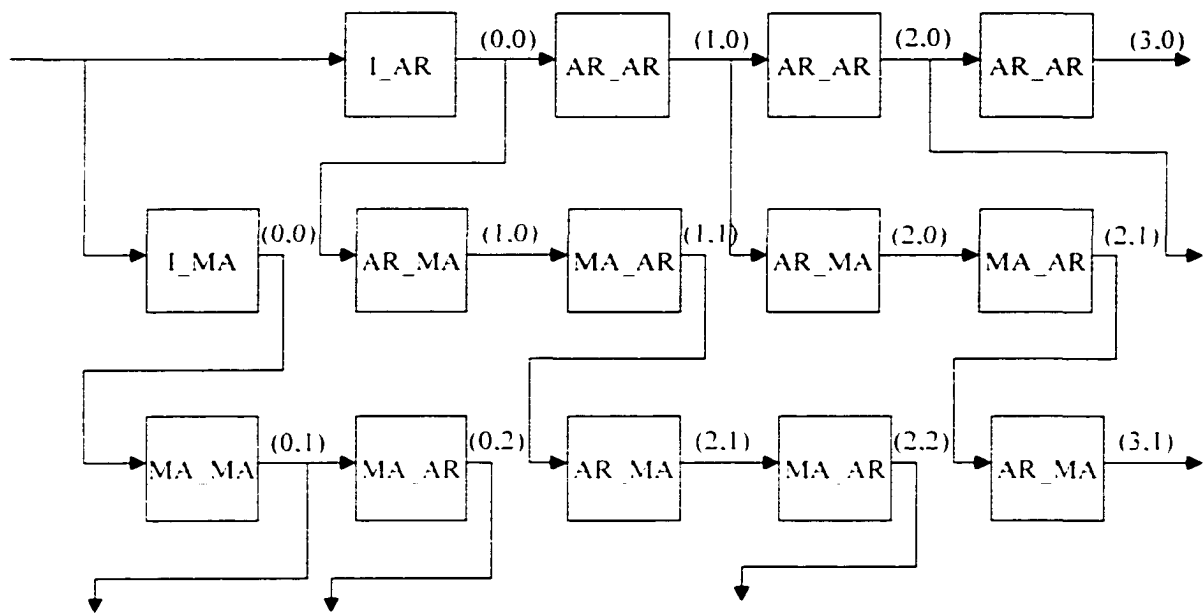


Figure 3.10 ARMA lattice order increments

In the figure, we notice that for any ARMA lattice with order of (p, q) , there might be not only one way to build it. They could include different types of basic blocks or have different sequences. e.g., ARMA $(2,1)$ could be formed as I_AR- AR_MA- MA_AR- AR_MA or I_AR- AR_AR- AR_MA- MA_AR or I_MA- MA_AR- AR_AR- AR_AR and so on. In all kinds of possible connections, there is one special arrangement so-called minimal, i.e., the order update is in the way of AR_MA, MA_AR alternating. The following is an example of such an arrangement.

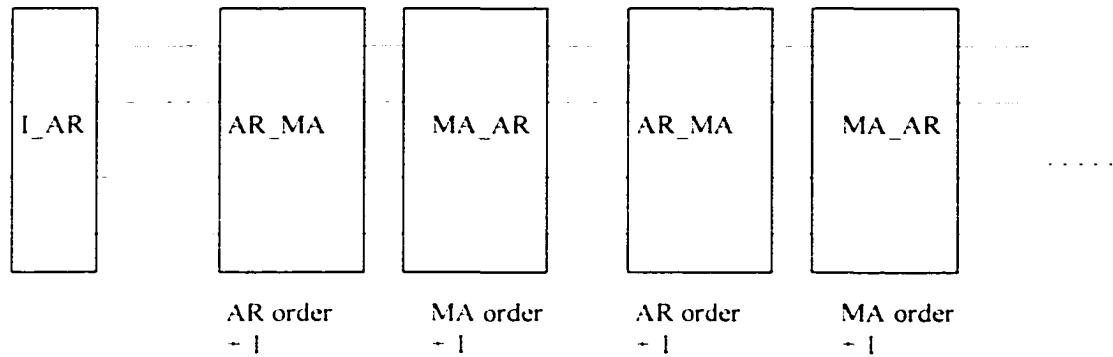


Figure 3.11 Minimal arrangement

Since being arranged in this way could cause the one of the reflection coefficients for each regular order update block AR_MA and MA_AR become to zero, the total number of coefficient required for the whole modeling is though reduced. Here is a numerical example. The system which needs to be modeled has poles and zeros shown as below

Table 3.2 Example: poles and zeros

	r	θ
Zeros	0.70	$\pm 10^\circ$
	0.80	$\pm 40^\circ$
	0.80	$\pm 54^\circ$
	0.85	$\pm 87^\circ$
	0.85	$\pm 93^\circ$
	0.70	$\pm 115^\circ$
	0.60	$\pm 150^\circ$
Poles	0.60	$\pm 12^\circ$
	0.90	$\pm 23^\circ$
	0.75	$\pm 72^\circ$
	0.96	$\pm 72^\circ$
	0.90	$\pm 94^\circ$
	0.96	$\pm 108^\circ$
	0.90	$\pm 132^\circ$

The impulse response sequence then can be generated for the modeling. A lattice model of order (8,8) is used. Thus there are 17 blocks inside it, starting from I_AR, followed by 8 AR_MA and MA_AR pairs. The corresponding coefficients are obtained as follow:

$$K_0 = 0.512129$$

Block 1 AR-MA	$K_{11} = 0.000000$, $K_{12} = -0.957997$
Block 2 MA_AR	$K_{21} = 0.000000$, $K_{22} = 0.092432$
Block 3 AR-MA	$K_{31} = 0.000000$, $K_{32} = -0.751382$
Block 4 MA_AR	$K_{41} = 0.000000$, $K_{42} = -0.095171$
Block 5 AR-MA	$K_{51} = 0.000000$, $K_{52} = 0.258404$
Block 6 MA_AR	$K_{61} = 0.000000$, $K_{62} = 0.448894$
Block 7 AR-MA	$K_{71} = 0.000000$, $K_{72} = -0.643009$
Block 8 MA_AR	$K_{81} = 0.000000$, $K_{82} = 0.182304$
Block 9 AR-MA	$K_{91} = 0.000000$, $K_{92} = 0.484343$
Block 10 MA_AR	$K_{101} = 0.000000$, $K_{102} = 0.441427$
Block 11 AR-MA	$K_{111} = 0.000000$, $K_{112} = -0.311877$
Block 12 MA_AR	$K_{121} = 0.000000$, $K_{122} = 0.095687$
Block 13 AR-MA	$K_{131} = 0.000000$, $K_{132} = 0.106936$
Block 14 MA_AR	$K_{141} = 0.000000$, $K_{142} = -0.496944$
Block 15 AR-MA	$K_{151} = 0.000000$, $K_{152} = -0.991207$
Block 16 MA_AR	$K_{161} = 0.000000$, $K_{162} = -0.377064$

So this gives us the idea of how to build an ARMA lattice model with minimum reflection coefficients. If the AR order is higher than the MA order, one can start from the I_AR block and follow by AR_AR blocks. The total number of the AR_AR blocks needed can be computed by subtraction of AR order from MA order. After all the AR_AR blocks, the AR_MA and MA_AR pairs are then arranged until it get to the order desired. In the contrast, if the MA order is higher than AR order, we can start from I_MA block and then followed by MA_MA blocks and AR_MA and MA_AR pairs. Table 3.3 shows the number of coefficients required for the ARMA lattice model with different orders. They are all arranged under the consideration of minimal arrangement.

Table 3.3 Numbers of lattice coefficients

	AR : 2	AR : 4	AR : 6	AR : 8
MA : 2	5	15	21	27
MA : 4	15	9	15	21
MA : 6	21	15	13	19
MA : 8	27	21	19	17

For convenience, we recall the number of coefficients needed for each block as follows:

Block Type: I_AR I_MA AR_AR MA_MA AR_MA MA_AR

Coefficients: 1 1 3 3 1 1

So, e.g.: ARMA(8,4) includes 1 I_AR, 4 AR_AR, 4 AR_MA, and 4MA_AR, so the total number of reflection coefficients required is $1*1+4*3+4*1+4*1=21$.

From the table, we found that within the same model length, the minimum total number of coefficients is always obtained when the AR order equals to the MA order. This

property has very special meaning when choosing the model order for the recognition application.

As we know, a 12th order AR model is the typical choice for the speech coding and recognition applications. Since the dimension of feature vectors after modeling is same as the dimension of code vectors in the next pattern recognition stage, we have to keep the new ARMA model in the same model length as the conventional AR model so that it will not add computation burden in following codebook training and testing procedure. Based on this consideration, we would rather choose ARMA (6,6) (13 lattice coefficients) than ARMA (8,4) (21 lattice coefficients) when we want the model length to be, e.g., around 12.

Besides the algorithm we proposed here, there are number of other lattice algorithms developed based on the recursive least square criterion. Comparing with those algorithms, our method has an advantage that, as we discussed before, with special arrangement the number of coefficients required for the model can be reduced. Moreover, since it only takes $(5+10N)$ multiplications and $(3+6N)$ additions (N is the model length, see in Appendix), compare with other algorithms, our method obtain more computation saving [16].

3.4 Properties of Lattice Algorithm

The lattice algorithm that we discussed in the previous section also has number of other desirable properties which are shared by most of lattice algorithms. In this section we

consider these properties and compare them with the corresponding direct-form algorithm.

Computational Requirements

The computational complexity of the traditional direct-form algorithms for ARMA modeling is proportional to N^2 (model length) [7]. In contrast, the lattice algorithm described in the previous section has a computational complexity that is proportional to N . It is computationally efficient.

Numerical properties

The lattice algorithm is numerically robust. First, it is numerically stable. The term *numerically stable* means that the output estimation error from the computational procedure is bounded when a bounded error signal is introduced at the input. Besides, all the reflection coefficients are bounded between -1 and 1 . Second, the numerical accuracy of the optimum solution is also relatively good compared to other direct-form algorithms. They are less sensitive to roundoff errors and coefficient quantizations [17].

Implementation Consideration

As we have observed, the lattice model structure is highly modular and allows for the computations to be pipelined. Because of the high degree of modularity, the lattice algorithm is suitable for implementation in VLSI.

As a result of this advantage in implementation and the desirable properties of stability, excellent numerical accuracy, and computational efficiency, we anticipate that in the near future, more and more modeling will be implemented by the lattice approach.

Chapter 4

4 Speech Recognition System

4.1 Introduction

4.1.1 Primary tasks

Speech recognition is generally used as a human-computer interface for other software. When it functions effectively in this role a speech recognition system performs three primary tasks (see Figure 4.1):

<i>Preprocessing</i>	converts the spoken input into a form the recognizer can process
<i>Recognition</i>	identifies what has been said
<i>Communication</i>	sends the recognized input to the software/hardware systems that need it

In order to understand what these tasks entail, the technology focus begins with a description of the data that speech recognition systems must handle. It describes how

speech is produced (called articulation), examines the stream of speech itself (called acoustics), and then characterizes the ability of the human ear to handle spoken input (called auditory perception). Once this groundwork has been laid the technology focus examines the demands of preprocessing in detail. The discussion is followed by this introduction.

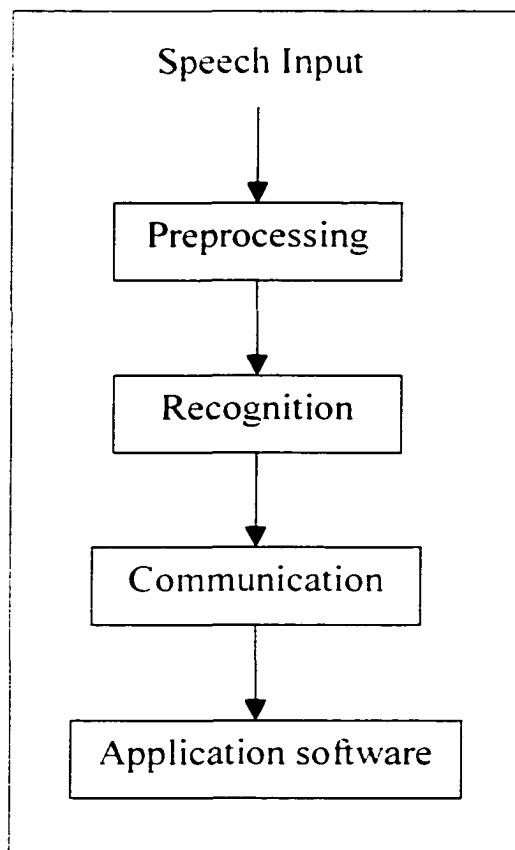


Figure 4.1 Components of speech recognition application

Preprocessing, recognition and communication should be invisible to the users of a speech recognition interface. The end user sees them indirectly as accuracy and speed of the system. Accuracy and speed are tools that users call upon to evaluate a speech recognition interface.

4.1.2 The data of speech recognition

The information needed to perform speech recognition is contained in the stream of speech. For humans, that flow of sounds and silences can be partitioned into discourses, sentences, words, and sounds. Speech recognition systems focus on words and sounds that distinguish one word from another in a language. Those sounds are called phonemes. The words "seat," "meat," "beat," and "cheat" are different words because, in each case, the initial sound ("s," "m," "b," and "ch") is recognized as a separate phoneme in English. The ability to differentiate words with distinct phonemes is as critical for speech recognition as it is for human beings.

There are a number of ways speech can be described and analyzed. The most commonly used approaches are *articulation*, *acoustics* and *auditory perception*. These three approaches offer insights into the nature of speech and provide tools to make recognition more accurate and efficient.

The articulation is concerned with how phonemes are produced. The focus of it is on the vocal (tract) apparatus (structure) of the throat, mouth and nose where the sounds of speech are produced. And ARPABET, a national system, was established by speech researchers to classify or label those phonemes.

Although articulation provides valuable information about how speech sounds are produced, a speech recognition system can not analyze the movements of the mouth.

Instead, the data source for speech recognition systems is the stream of speech itself. Like all sound streams, speech is an analog signal: a continuous flow of sounds waves and silence. Use an acoustic-based model to analyze speech data is still the most popular and reliable method for the recognition task.

The ability of the human auditory processing system suggests that an auditory-based speech recognition system would be superior to systems based on acoustics and signal processing. Unfortunately, our understanding of human speech perception is incomplete. Although research found significant improvements in recognition accuracy when coding based upon auditory models is used in conventional recognition systems, full utilization must await results of a great deal of additional research.

4.1.3 Recognition approaches

The recognizer in the system performs its primary function: to identify what the user has said. The three competing recognition technologies found in commercial speech recognition systems are:

- Pattern recognition
- Acoustic-phonetic recognition
- Stochastic processing

These approaches differ in speed, accuracy, and storage requirements.

The pattern recognition represents speech data as sets of feature/parameter vectors called patterns or templates. Thus the fundamental of this approach is pattern training. The units

being trained can be phrases, words or sub-word units, and then stored as reference patterns / templates. The spoken input are organized into patterns prior to performing recognition, then one compare the input with stored patterns and find the best match. Provide a good training set is very important to this approach. Pattern recognition was the dominant recognition methodology in 1950's and 1960's. In 1980's the new dynamic time warping (DTW) algorithm was introduced as a fast, robust and accurate one which was then widely used even today. Pattern recognition performs very well with small vocabularies of phonetically distinct items but has difficulty making the fine distinctions required for larger vocabulary recognition and recognition of vocabularies containing similar-sounding words (called confusable words). Since it operates at word level there must be at least one stored pattern/template for each word in the application vocabulary.

Unlike pattern recognition, acoustic-phonetic recognition functions at the phoneme level. Theoretically, it is an attractive approach to speech recognition because it limits the number of representations that must be stored to the number of phonemes needed for a language. For English, that number is around forty, no matter how large the application vocabulary. The basic techniques of this approach are feature analysis (i.e. measurement of invariant of sounds), segmentation of the feature contours into consistent group of features and labeling of the segmented features so as to detect words, sentences. Acoustic-phonetic recognition supplanted pattern recognition in the early 1970's. However it was not fully developed due to poor understanding of basic knowledge on some aspects. This technology combined with the development of powerful but inexpensive computing hardware, has led to renewed interest by researchers.

The term stochastic refers to the process of making a sequence of non-deterministic selections from among sets of alternatives. They are non-deterministic because the choices during the recognition process are governed by the characteristics of the input and not specified in advance. Like pattern recognition, stochastic processing requires the creation and storage of models of each of the items that will be recognized. But stochastic processing involves no direct matching between stored models and input. Instead, it is based upon complex statistical and probabilistic analyses which are best understood by examining the network-like structure like HMM (hidden markov model). Researchers began investigating using HMM for speech recognition in the early 1970's, but this method did not gain widespread acceptance for commercial system until the late 1980's. However, by the 1990's HMM had become the dominant approach to continuous speech recognition.

4.2 Isolated Word Speech Recognition System

4.2.1 Our task

According to the vocabulary type and size, the recognition application can be divided into groups shown in Table 4.1:

Table 4.1 Vocabulary types and sizes

Type	Isolated words (≤ 2)	*
	Connected words (> 2)	
	Continuous words	
Size	small vocabulary (< 20)	*
	Moderate vocabulary (around 100)	
	Large vocabulary (around 1000)	

Our recognition task is to recognize the isolated words (called discrete utterances) which are probably used for voice command of a software application. An isolated word system operates on a single word at a time. This kind of recognition task is generally more difficult when vocabularies are large or have many similar-sounding words. Our database consists of 10 English words: it belongs to the small vocabulary size.

To build a system for handling this small vocabulary size and isolated word recognition task, the appropriate data analysis method could be acoustic signal modeling. The modeling techniques (e.g. LP), have been used in this area since the 1960's with many successful applications. In our application, we would use ARMA lattice modeling due to its superiorities that have been discussed in Chapter 3. As to recognition part, basically, as we introduced in the previous section, the pattern recognition is the best and economic choice for this task, since it performs very well with small vocabularies of phonetically distinct items.

4.2.2 Recognition system model

This is the basic pattern recognition system model (Figure 4.2). Input $S(n)$ is analyzed based on some parametric model (ARMA lattice model) to give the test pattern T , then compared to a pre-stored set of reference patterns $\{R_v\}$. Using a pattern classifier (i.e. a similarity procedure) the pattern similarity scores are then sent to a decision algorithm and then choose the best transcription of the input speech and output the index of reference.

The reasons for using this block diagram model are because it is easy to implement in either software or hardware, and it works well in practice.

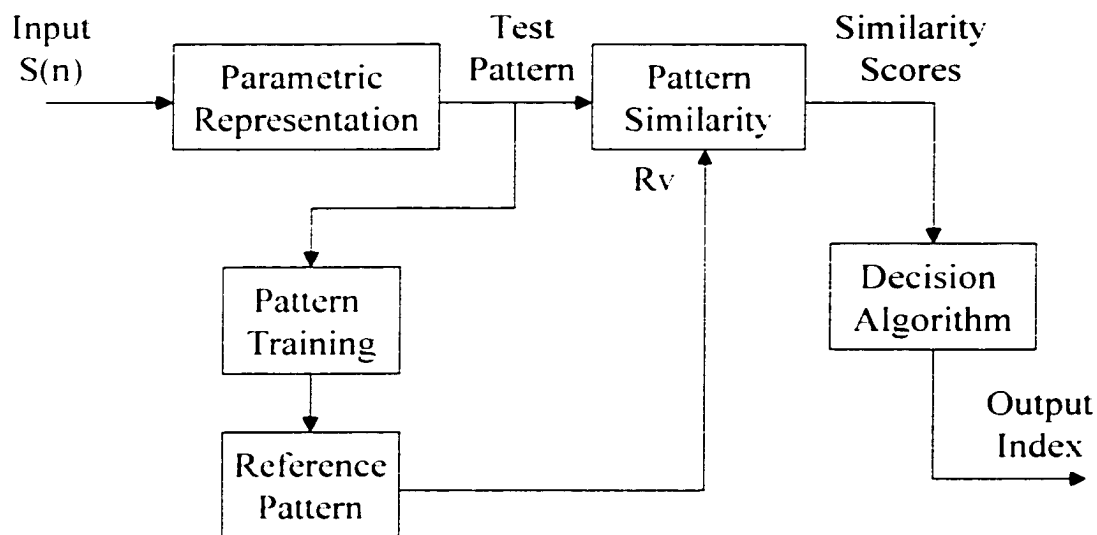


Figure 4.2 Block diagram of recognition system

4.3 Parametric Representation

Parametric representation is the front-end processor that has been widely used in speech recognition systems. Figure 4.3 shows its block diagram.

Basically, parametric representation includes two parts: preprocessing and ARMA lattice analysis. (If using the LP model, the second part changes to LP analysis accordingly)

A typical parametric representation procedure starts with the digitization (i.e. sampling) of a continuous waveform at a sampling rate that is at least twice the highest significant frequency of the waveform. The remaining part of this analog-to-digital conversion is quantization of the discrete data. Many of voice recording devices or softwares complete this part automatically. The resulting data are then preprocessed digitally by filtering, windowing or other combination of various techniques so that the desired information can be extracted or enhanced. In the following, we will give the detail description of each block in Figure 4.3.

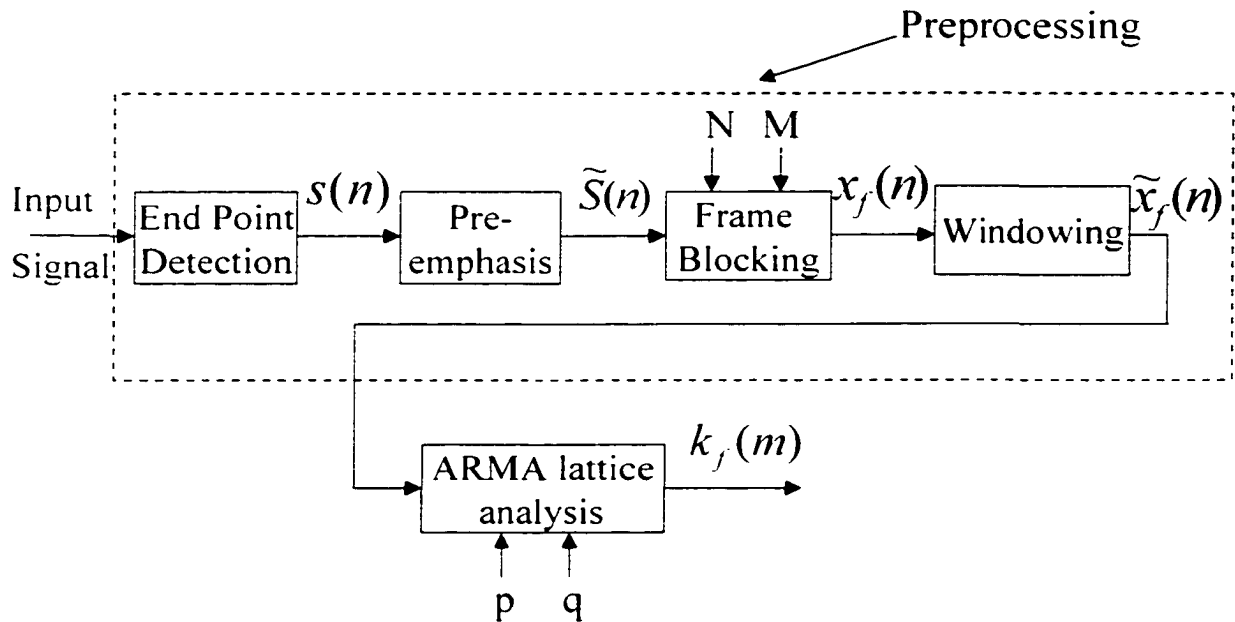


Figure 4.3 Block diagram of parametric representation

1. **Endpoint Detection**---The goal of endpoint detection is to separate acoustic events of interest from silences and background noises. A threshold could be set up for this purpose. A simple detector functions in the way: when the input signal is found to have three continuous samples above the threshold or under the threshold, the first one above and first one under the threshold could be thought as the start point and the end point of the utterance.

2. **Pre-emphasis**---The speech signal $s(n)$, is put through a first order ($a=0.95$) FIR filter to spectrally flatten the speech signal. Perhaps the most widely used pre-emphasis network is the following system:

$$H(z) = 1 - az^{-1} \quad (4.1)$$

In this case, the output of the pre-emphasis network is related to the input to the network by the difference equation

$$\tilde{s}(n) = s(n) - as(n-1) \quad (4.2)$$

The most common value for a is around 0.95 [18].

3. **Frame Blocking**---In this step the pre-emphasized speech signal is blocked into frames of N samples, with adjacent frames being separated by M samples. In practice, N should be as small as possible to reduce the total computation load. Figure 4.4 illustrates the blocking into frames for the typical case in which $M=(1/3)N$ [18].

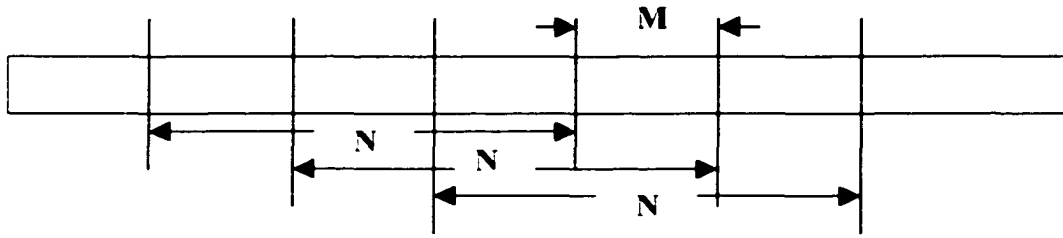


Figure 4.4 Blocking into overlapping frames

The first illustrated frame consists of the first N speech samples. The second frame begins M samples after the first frame, and overlaps it by $N-M$ samples. Similarly, the third frame begins $2M$ samples after the first frame and overlaps it by $N-2M$ samples. If we denote the f^{th} frame of speech by $x_f(n)$, and there are L frames within the entire speech signal, then

$$x_f(n) = \tilde{s}(Mf + n) \quad (4.3)$$

where $n=0,1,\dots,N-1$ and $f=0,1,\dots,L-1$

Typical values for N and M are 240 and 80 when the sampling rate of the speech is 8 kHz. These correspond to 30-msec frames, separated by 10 msec.

4. **Windowing**--- The next step in the preprocessing is to window each individual frame so as to minimize the signal discontinuities at the beginning and the end of each frame. The concept here is identical to the one discussed with regard to the frequency domain interpretation of a short-time spectrum, to use the window to taper the signal to zero at the beginning and the end of each frame. If we define the window as $w(n)$, $0 \leq n \leq N-1$, then the result of windowing is the signal

$$\tilde{x}_f(n) = x_f(n)w(n) \quad (4.4)$$

A typical window used here is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1 \quad (4.5)$$

5. **ARMA lattice analysis**---This is the last step for the parametric representation. With the pre-known order p and q , the reflection coefficients for each frame then can be computed. e.g. if select $p = q = 6$, there would be 13 coefficients generated from the lattice analysis.

4.4 Pattern Training

Pattern recognition processes consist of training and testing. Pattern training is by which the representative sound patterns are converted into the reference patterns for use by the pattern similarity decision. The training methods include casual training, robust training and clustering training. Among them, the clustering training, which means large number of versions of each vocabulary entry are trained to create one reference pattern, is usually used for speech reference pattern training [18]. The training is shown in Figure 4.5, when put the reflection coefficients of each frame (1 to L) from ARMA lattice analysis as input of the training procedure: the codebook CB is designed to minimize the distortion between input vectors and reference vectors.

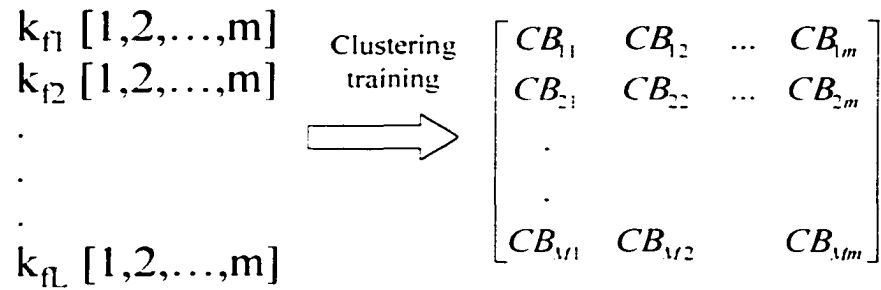


Figure 4.5 pattern training

4.5 Pattern Similarity Decision

After the training procedure, suppose we have N codebooks for N utterance classes. The following is a recognition similarity decision system shown in Figure 4.6. After computing the distortion between the input utterance and all reference codebooks, the utterance is then recognized as the one which has minimum distortion. The index of that reference codebook will be output as the final recognition result.

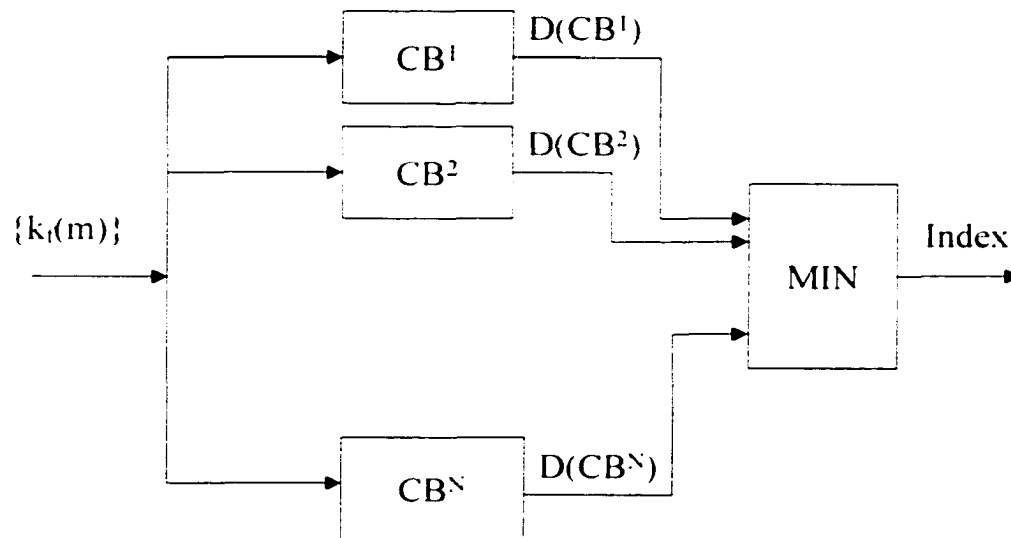


Figure 4.6 Similarity decision system

During the operation, the N codebooks are used to test the unknown utterance. The input comes in frame by frame. The resulting N average distortion scores D are obtained by the following equation:

$$D(CB^{(i)}) = \frac{1}{L} \sum_{t=1}^L d(k_t, \hat{k}_t^{(i)}), i = 1, 2, \dots, N \quad (4.6)$$

where $\hat{k}_t^{(i)}$ satisfying

$$\hat{k}_t^{(i)} = \arg \min d(k_t, CB_{i,t}^{(i)}) \quad (4.7)$$

Then the utterance is recognized as class K if

$$D(CB^{(K)}) = \min D(CB^{(i)}) \quad (4.8)$$

K is the index of the codebook as which input utterance recognized.

Chapter 5

5 Pattern Recognition Techniques

5.1 Introduction

The key question in pattern recognition is how to generate the reference patterns so that the test patterns can be compared with them to determine their similarity. Depending on the different techniques, it can be done in a variety of ways.

As we mentioned in Chapter 4, pattern recognition processes consist of training and testing. During training, the pattern template of each known word must be created. Each template consists of a set of features extracted from spoken utterances. The exact form of template will depend on the nature of the pattern recognition algorithm used. During testing, patterns of unknown words are compared with those templates; the decision is then made by specific rules. The basic structure of the speech pattern recognition process is shown in Figure 5.1.

Many different types of pattern recognition techniques can be used in speech recognition system, such as vector quantization and neural network techniques.

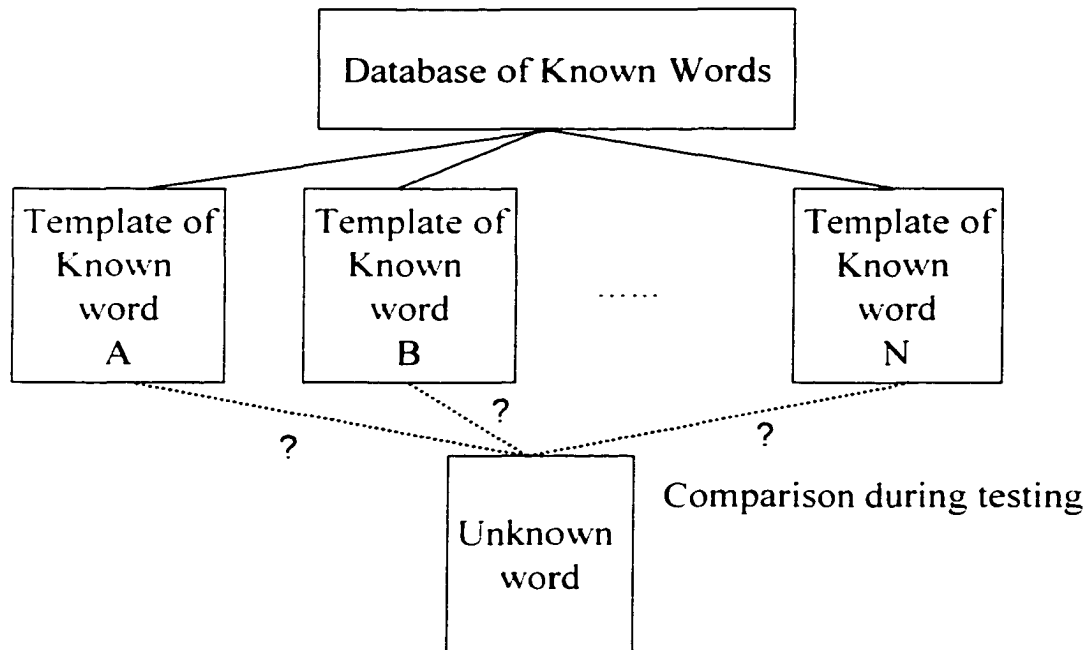


Figure 5.1 Pattern recognition structure

5.2 Vector Quantization

Vector quantization (VQ) [19] is an effective method of segregating data into clusters and determining the centroids of those clusters. VQ reduces a set of L m -dimensional vectors into a codebook of M centroid vectors where $L \gg M$.

VQ was originally designed for speech transmission systems to reduce the bandwidth of signals. Instead of transmitting all the bits necessary to represent the m -dimensional vector, only the codebook entry number of the centroid closest to the vector would need to be transmitted. Thus, a sequence of codebook entry numbers could be transmitted to represent an entire utterance.

5.2.1 Definition

A vector quantizer Q of dimension m and size M is a mapping from a vector in m -dimensional Euclidean space, R^m , into a finite set C .

$$Q: R^m \rightarrow C$$

where Q is the vector quantizer and R is the Euclidean space, and

C is called codebook and is defined by $C = (y_1, y_2, \dots, y_M)$, $y_i \in R^m$, where y_i is called code vector, i is the index of the code vector.

Associated with every code vector is a partition (cell) in set of R^m . The i^{th} cell which is associated with y_i is defined by

$$R_i = \{x \in R^m : Q(x) = y_i\},$$

for which to be a cell, it naturally follows that

$$\bigcup_i R_i = R^m \text{ and } R_i \cap R_j = \emptyset \text{ for } i \neq j.$$

so that the cells form a partition of R^m .

The above basic definitions have described the components in a vector quantizer. Figure 5.2 shows a 2-D VQ, with $M=6$ code vectors in the codebook. The input X is in 3rd cell, and therefore, will be quantized to vector y_3 with index $i=3$.

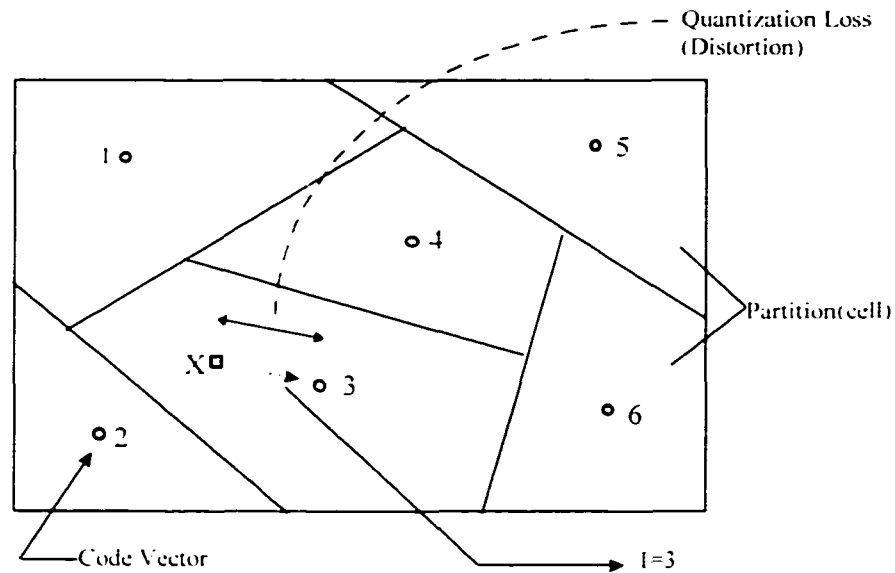


Figure 5.2 Elements in VQ

5.2.2 Elements of VQ implementation

To build a VQ codebook and implement a VQ-based recognition procedure, we need the following:

1. A large set of feature vectors which form a training set. The training set is used to create the optimal set of code vectors for representing the feature variability observed in the training set. We require the sufficient number of training vectors to be much greater than the number of code vectors (at least several times the code vectors), so as to be able to find the best set of code vectors in a robust manner. In practice, it has been found that the number of training vectors should be at least ten times the number of code vectors to train a VQ codebook that works reasonably well [18].

2. A measure of similarity, or distance, between a pair of vectors so as to be able to cluster the training set vectors as well as to associate or classify arbitrary input vectors into unique codebook entries.
3. A centroid computation procedure. On the basis of the partitioning that classifies the training set vectors into M clusters, we choose the M code vectors as the centroid of each of the M clusters.
4. A classification procedure for arbitrary speech input vectors that chooses the code vector closest to the input vector and uses the codebook index as the resulting recognition result. This is often referred to as the nearest-neighbor labeling or optimal encoding procedure. The classification procedure is essentially a quantizer that accepts, as input, a speech vector and provides, as output, the codebook index of the code vector that best matches the input.

5.2.3 VQ training

The way in which a set of training vectors can be clustered into a set of M code vectors is the following (this procedure is known as the generalized Lloyd algorithm):

1. **Initialization:** Arbitrarily choose M vectors as the initial set of code words in the codebook.
2. **Nearest-neighbor search:** For each training vector, find the code vector in the current codebook that is closest and assign that vector to the corresponding cell (associated with the closest code word).

3. **Centroid update:** Update the codeword in each cell using the centroid of the training vectors assigned to that cell.
4. **Iteration:** Repeat steps 2 and 3 until the average distance falls below a present threshold.

From the above procedure, we see the drawback of this method. The minimum distance search is an exhaustive search algorithm which walks through all the code vectors in the codebook. Moreover, centroid update is computation consuming too.

5.3 Neural Network approach

Artificial neural networks (ANN) are computational models that attempt to emulate the human brain by a topology that resembles interconnected nerve cells. NNs are capable of doing many different jobs, such as classification, associative memory and clustering. The main drawback of neural networks is their long training time. Although knowledge about neural networks is still in an early stage, their learning power on the application to speech recognition is significant.

5.3.1 Architecture

Introduced by Kohonen [20], the architecture for the net that can be used to cluster a set of m -dimension training vectors X into M clusters Y is shown below. W is called the weight vector that is associated with the input and the output neural.

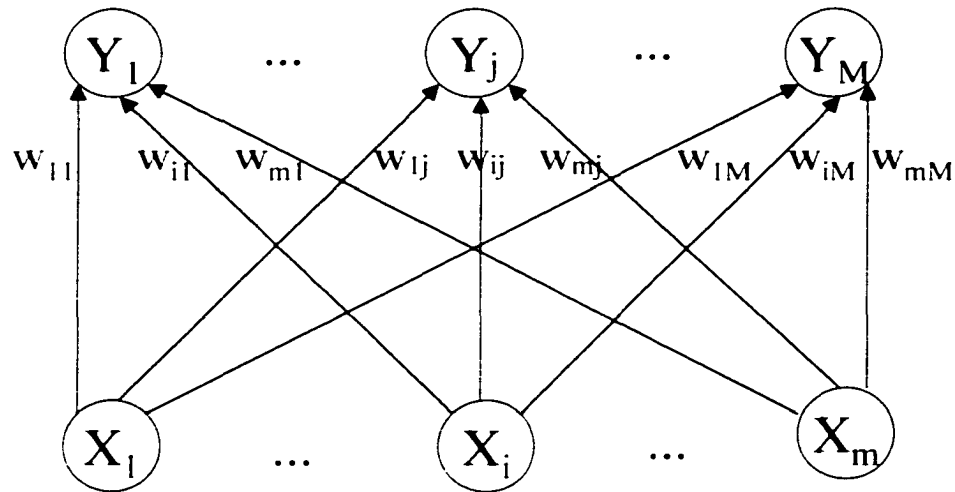


Figure 5.3 Neural net architecture

5.3.2 Learning algorithm

The motivation for the algorithm is to find the winner output unit that is closest to the input vector. Euclidean measure is used for the distance measurement. It is a pattern classification method in which each output unit represents a particular class. The weight vector for an output unit is often referred to as a reference vector (code vector) for the class that the unit represents.

The following is the nomenclature we use in the algorithm

X	training vector
T	correct class for the training vector
W_j	weight vector for j th output unit
$\ X - W_j\$	Euclidean distance between input vector and weight vector for j th output unit

Algorithm

- Step 0 initialize weight vectors and learning rate $\alpha(0)$
- Step 1 while stopping condition is false. do step 2-6
- Step 2 for each training input vector X . do step 3-4
- Step 3 Find j so that $\|X-W_j\|$ is a minimum
- Step 4 update W_j as follows:
- $W_j(t+1)=W_j(t)+\alpha(t)[X(t)-W_j(t)]$
- Step 5 reduce learning rate
- Step 6 test stopping condition

5.3.3 General considerations

Initialization of weight vectors

The simplest method of initializing the weight vectors is to take the first M training vectors and use them as weight vectors. The remaining vectors are then used for training [21]. Another simple method is to assign the initial weights randomly.

Stopping rules

The stopping condition in the algorithm may specify a fixed number of iterations or the learning rate reaching a sufficiently small value.

It often happens that the neural network algorithms “overlearn”, i.e., when learning and test phases are alternated, the recognition accuracy is first improved until an optimum is

reached: after that, when learning is continued, the accuracy starts to decrease slowly. A possible explanation is that when the weight vectors become very specifically tuned to the training data, the ability of the algorithm to generalize for new data suffers from that. It is therefore necessary to stop the learning process after some optimal number of steps, say, 50 to 200 times the total number of the weight vectors (depending on the particular algorithm and learning rate). Such a stopping rule can only be found by experience, and it also depends on the input data [21].

Learning rate

The learning rate α is a slowly decreasing function of time (or training epochs). Kohonen [20] indicates that a linearly decreasing function is satisfactory for practical computations. At the beginning it should stay above 0.1. For good statistical accuracy, α should be maintained during the convergence phase at a small value (on the order of 0.01 or less) for a fairly long period of time, which is typically thousands of iterations.

5.4 Comparison

Table 5.1 Comparison between VQ and NN approach

	VQ	NN approach
1	Euclidean distance measurement	Euclidean distance measurement
2	Computation consuming centroids calculation	Simple update formula
3	No convergency control parameter. It is based on Lloyd Iteration.	Learning rate parameters
4	May fall in local minima	Embedded relaxation process which reduces the chance of locking in local minima

Table 5.1 shows the comparison between vector quantization and neural network approach. It is obvious that the structures of VQ and NN approach are very similar. They all need to train a codebook based on the inputs. However, the NN approach seems has more control on the training process, e.g., the learning rate and the number of iterations.

The learning rate α is an important parameter, with its decreasing during the learning procedure, the impact by those further iterations is getting smaller and smaller. The phenomenon is similar to simulate annealing which is a stochastic relaxation technique. The reason of that is to reduce the chance of falling into local minima. This stochastic relaxation technique has been introduced to some improved method of VQ. However, it is already embedded in the NN approach.

VQ is a computational intensive algorithm. The centroids calculation as well as the minimum Euclidean distance search are exhaustive calculations. Although NN approach involves the same distance measure, it has a simple update formula for the new weight vectors.

Besides the above considerations, the neural network approach gets lots of research support for efficient hardware implementations such as, it can readily implement a massive degree of parallel computation, because a neural net is a highly parallel structure of simple, identical, computational elements. Moreover, it intrinsically possesses a great deal of robustness or fault tolerance. Since the "information" embedded in the neural network is "spreaded" to every computational element within the network, this structure

is inherently among the least sensitive of networks to noise or defects within the structure.

Chapter 6

6 Experiments and Results

6.1 Database Collection

In the previous sections, we have already built up a system for recognizing isolated words. To test this system, we need to have a database which contains the isolated word we are going to recognize.

The pre-recorded database that we used for the experiment is specially designed for our isolated word speech recognition application. It is a collection of speech recordings which is accessible in computer readable form (*.wav format). The following is the description of the database:

Linguistic contents

- isolated English words and names:
Call, Hangup, No, Yes, Halima, Hari, John, Tracy, Walter, and Wayne
- each word is repeated 10 times by each speaker

Numbers and types of speakers

- 10 speakers, 5 females, 5 males

Recording conditions

- speech recorded under the awareness of speakers
- in quiet lab/office environment
- read speech recording
- single-channel

Recording equipment

- microphone: table-top microphone
- recording device/processor: sound recorder

Sampling rate

- 8 kHz

6.2 Experimental Design

Two kinds of experiments were designed to test the recognition system: speaker-dependent test and speaker-independent test. Basically, some recognition systems require speaker enrollment---a user must provide samples of his or her speech before using them, whereas other systems are said to be speaker-independent, in that no enrollment is necessary. Simply speaking, speaker-dependent applications must use voice samples from the same group of speakers for both training and testing. And speaker-independent

applications must use voice samples from different groups of speakers (for both cases, none of the individual sample were used for both training and testing). Apparently, the speaker-independent case could be more difficult and the recognition accuracy is lower than for the speaker dependent one.

As we described in the previous section, our database contains 10 English words with each repeated 10 times by 10 speakers. So for each word, there are 100 samples from 10 different speakers. For speaker-dependent tests, the samples of each word were divided into two sessions. Session 1 contained 60 samples from 10 different speakers (6 samples per speaker) and session 2 took the remaining 40 samples, they were non-overlapping sessions. In the experiments, session 1 was used as training data, whereas session 2 was for testing. Since the training data and testing data come from the same 10 speakers, the tests then are said to be speaker-dependent.

For speaker-independent tests, we divided database into two sessions in another way. This time, for each word session 1 consisted of 60 samples from 6 different speakers (10 samples per speaker) and sessions 2 had the remaining 40 samples form the other 4 speakers. The training was then done on session 1 and testing was done on session 2. Since we use other speakers voices to test the pre-trained system, this is called speaker-independent.

6.3 Implementation Issues

Before showing the results, some implementation issues are described below. In the parametric representation, prior to any analysis, the speech signal (sampling rate is 8 kHz) is pre-emphasized by a first order filter $1-0.95z^{-1}$. The analysis was done over frames of 30 ms duration (240 samples). The overlap between frames was 20 ms. For the LP analysis, the autocorrelation method was used to get traditional 12th order AR model coefficients. The lattice method discussed in Chapter 3 was used to calculate the reflection coefficients of dimension 13 for the ARMA (6,6) model. The reasons why we choose 6/6 as the order of the ARMA model have already been explained in Chapter 3.

In pattern recognition process, the NN classifier (as described in Chapter 5) was trained using 12-dimension AR feature vectors and 13-dimension ARMA feature vectors. The codebooks for each word were then generated from session 1. In some experiments the codebook sizes were chosen of 16, 32 and 64 for the comparison. According to the stopping rule discussed in Chapter 5, the total number of iterations should be 50~200 times of codebook size, so in our experiments, the training process will stop at 1000, 2000 and 3500 epochs for codebook size of 16, 32, and 64 respectively. And the learning rate was determined by $\alpha(t) = 0.1 - \frac{0.1 - 0.01}{T}t$, T is the total number of iterations and t is the index of iterations. Once the codebooks had been generated, the recognition results were then obtained from testing session 2 on those pre-stored codebooks.

6.4 Results and Discussions

6.4.1 Speaker-dependent tests

The Table 6.1 shows the experimental results for speaker-dependent tests. For both ARMA lattice and AR (LP) modeling, three groups results are obtained for codebook size of 16, 32 and 64 respectively.

Table 6.1 Results for speaker-independent recognition

	ARMA			AR		
	16	32	64	16	32	64
Call	34/40	37/40	38/40	30/40	33/40	34/40
Yes	32/40	36/40	37/40	32/40	34/40	34/40
No	33/40	34/40	36/40	26/40	30/40	30/40
Hangup	38/40	39/40	40/40	26/40	36/40	36/40
Halima	32/40	37/40	39/40	30/40	35/40	36/40
Hari	32/40	36/40	37/40	29/40	32/40	34/40
John	34/40	38/40	38/40	26/40	26/40	30/40
Tracy	36/40	37/40	39/40	30/40	34/40	35/40
Walter	34/40	39/40	40/40	28/40	33/40	36/40
Wayne	33/40	36/40	38/40	23/40	26/40	30/40
Overall	338/400	369/400	382/400	280/400	319/400	335/400

From the table, we can see that for both AR and ARMA methods, the recognition rates improved by increasing the codebook size. The AR models could be more sensitive to the changes of the codebook size due to relatively larger variations in LP coefficients.

Here is an example of recognizing a word sample "no". The left side shows results obtained using ARMA lattice modeling with a codebook of size 32, whereas the right side shows the results using the same modeling with a codebook of size 64. Since there are more code vectors to represent input signals in 64 case, the values for the distortion between the input and the codebooks are reduced.

*** no.wav:

distance with no	is 0.357394
distance with halima	is 0.610437
distance with hangup	is 0.676647
distance with hari	is 0.764919
distance with john	is 0.439352
distance with call	is 0.489176
distance with tracy	is 0.847844
distance with walter	is 0.503466
distance with wayne	is 0.6002
distance with yes	is 0.964889

*** It is

no

*** no.wav:

distance with no	is 0.3442
distance with halima	is 0.510472
distance with hangup	is 0.64445
distance with hari	is 0.7414
distance with john	is 0.395938
distance with call	is 0.464484
distance with tracy	is 0.748699
distance with walter	is 0.449677
distance with wayne	is 0.523466
distance with yes	is 0.830009

*** It is

no

The following is the comparison between the ARMA lattice modeling (left side) and AR modeling (right side): the codebook sizes are 64 for both cases. Since the AR model has more variations in LP coefficients, the distortion measured are much bigger than those of the ARMA model. The results also show the test using AR modeling has not recognized "no" correctly: this might be due to the poor capability of AR modeling for the words with nasal sounds.

*** no.wav:

distance with no	is 0.3442
distance with halima	is 0.510472
distance with hangup	is 0.64445
distance with hari	is 0.7414
distance with john	is 0.395938
distance with call	is 0.464484
distance with tracy	is 0.748699
distance with walter	is 0.449677
distance with wayne	is 0.523466
distance with yes	is 0.830009

*** It is

no

*** no.wav:

distance with no	is 3.39076
distance with halima	is 4.27658
distance with hangup	is 4.3432
distance with hari	is 4.29334
distance with john	is 3.52464
distance with call	is 3.02287
distance with tracy	is 4.78638
distance with walter	is 3.86665
distance with wayne	is 3.83143
distance with yes	is 4.82915

*** It is

call

To see the results more clearly, three comparison figures are generated from the above table.

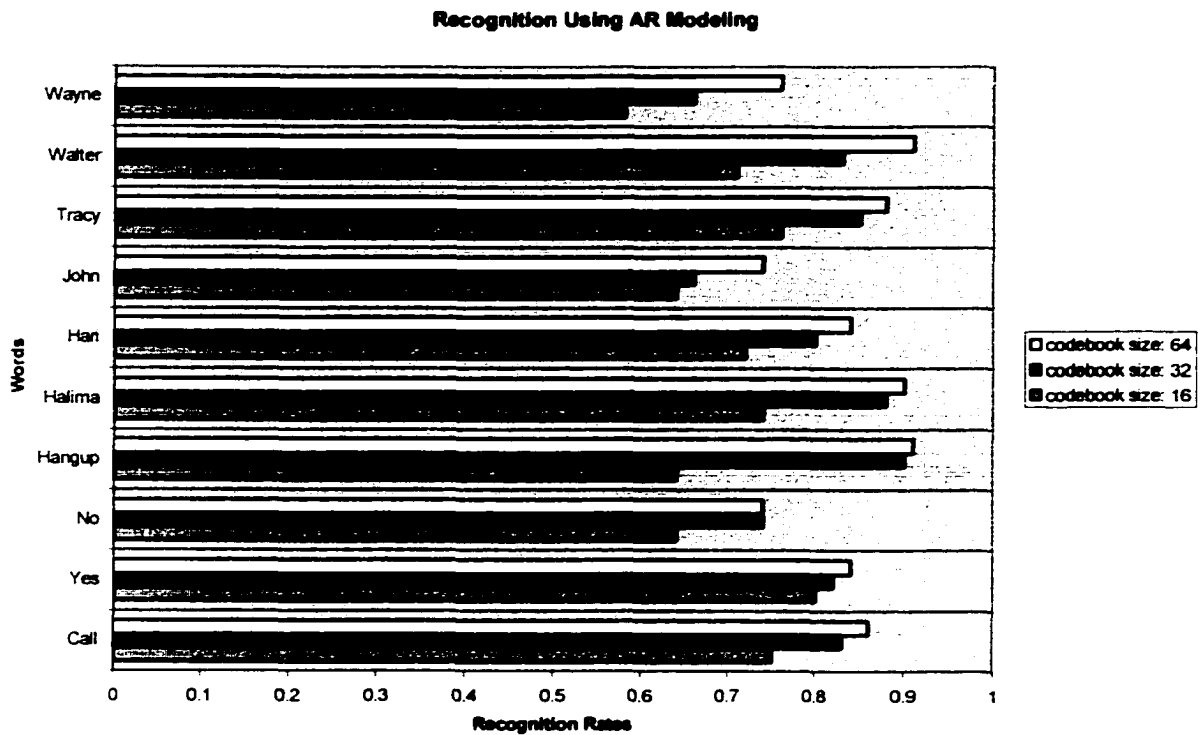


Figure 6.1 Recognition using AR modeling

Figure 6.1 only gives the performance of the AR modeling for speaker-dependent recognition. From the figure, compare the results from codebook size of 16 and 32, the longer utterances, longer words, e.g. halima, hangup, walter, which contains more phonetic information are found take more benefits of increasing codebook size than other short ones. This is probably because they need more code vectors to represent their features. One exception is for the hangup in ARMA modeling (see in Figure 6.2), the results shows no big difference between codebook 32 and 16; I guess this is because this word is so distinctive in its nasal pronunciation when compared with other words, so it is easy to be recognized.

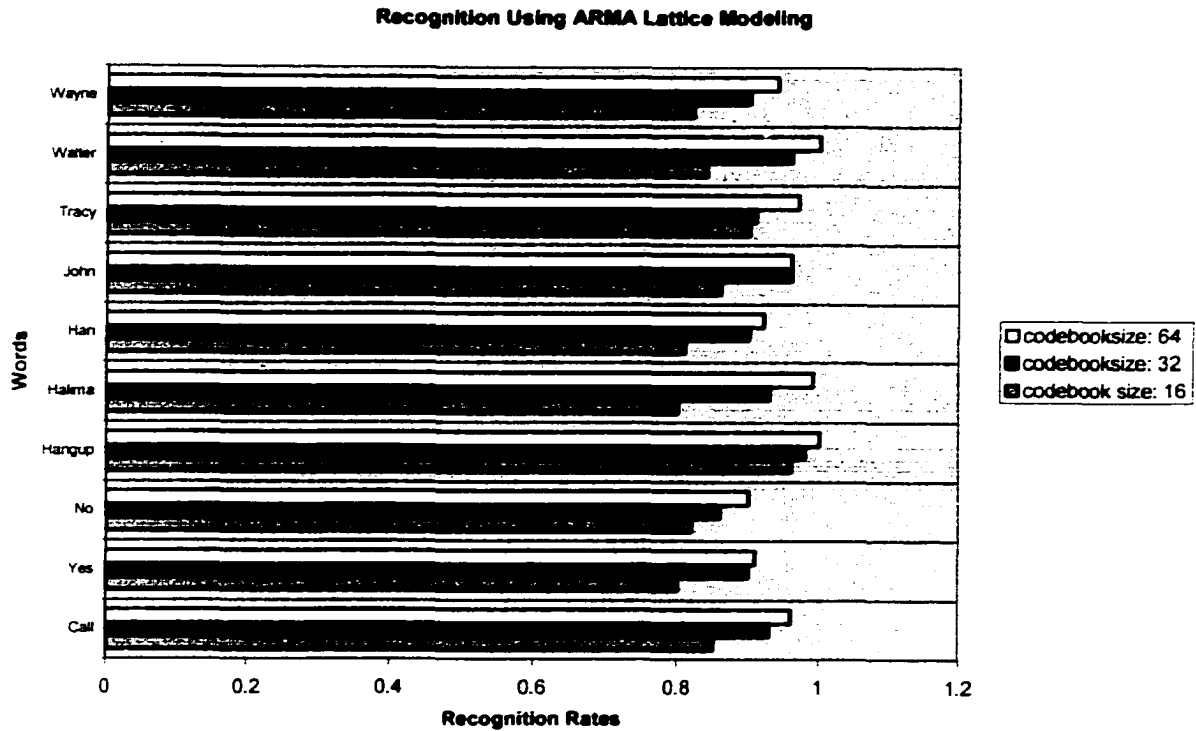


Figure 6.2 Recognition using ARMA lattice modeling

Although the utterances especially longer ones get significant improvement on recognition rate with number of code vectors increased to 32, most of the recognition rate can not get such improvement as number of the code vectors continue to jump from 32 to 64. That implies the codebook size must be closer and closer to a limit. After a certain point, they are large enough for all utterances, and the recognition rates will not improve any more with the size increasing. In practice, we always want the codebook size as small as possible with the acceptable performance. First of all, it could save storage; second, it will reduce the computation for the similarity decision. On the other hand, with larger codebooks, the speech can be better characterized, but at significant computational expense. So it is always necessary to find a balance (trade-off) between these two issues. F. K. Soong reported error rates of 20% for codebooks of size 4, 10% for size 8, and 2%

for size 64 for identification based on utterances of 10 digits in a noise-free environment [22]. In fact, 64 is the largest size that has ever been found in literature on isolated word speech recognition applications. The experimental results shown here also give a proof that 64 could be an appropriate choice. We notice that all the recognition rates for codebook size of 64 are higher than 90% for the ARMA case.

Figure 6.3 presents the comparison of two modeling methods. With the same codebook size, 64, ARMA lattice modeling proves remarkably better in the recognition performances. Especially for the words with nasal sound like no, john, wayne, the superiority of ARMA method is apparently proved.

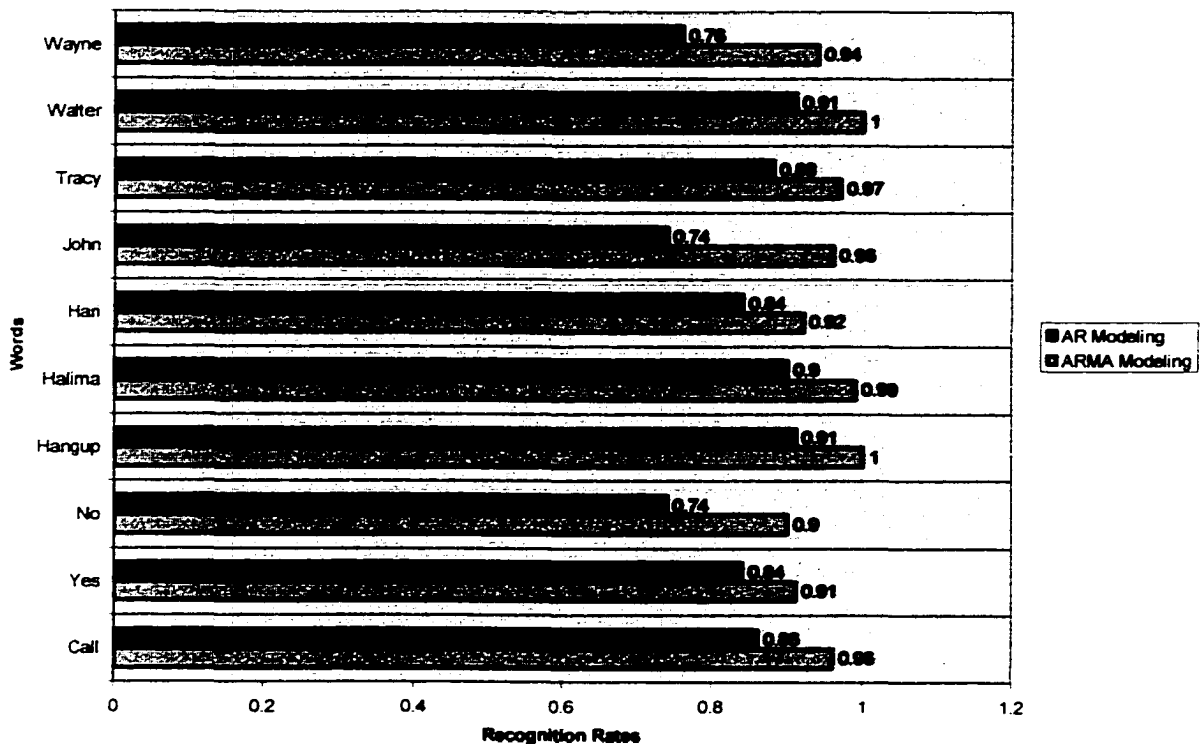


Figure 6.3 AR modeling vs ARMA lattice modeling

It is also interesting to notice that the longer utterances, e.g., hangup, halima and walter, tend to be easier to be recognized due to their rich and distinct phonetic contents. In the opposite, shorter utterances are easier to be buried by other utterances with partly similar sounds.

6.4.2 Speaker-independent tests

In Table 6.2, the speaker-independent experiments show similar results as the speaker-dependent one, except their recognition rates degraded due to the testing of different speakers.

Table 6.2 Results for speaker-independent recognition

	ARMA			AR		
	16	32	64	16	32	64
Call	25/40	31/40	33/40	19/40	24/40	29/40
Yes	29/40	32/40	35/40	19/40	26/40	30/40
No	25/40	30/40	32/40	18/40	25/40	27/40
Hangup	34/40	36/40	38/40	22/40	28/40	32/40
Halima	29/40	35/40	39/40	20/40	30/40	34/40
Hari	30/40	34/40	35/40	25/40	29/40	33/40
John	33/40	36/40	36/40	20/40	27/40	31/40
Tracy	30/40	33/40	37/40	23/40	28/40	32/40
Walter	30/40	35/40	37/40	24/40	29/40	33/40
Wayne	32/40	35/40	36/40	21/40	27/40	30/40
Overall	297/400	337/400	358/400	211/400	273/400	281/400

In pattern recognition, the speaker variability is typically modeled using statistical technique applied to a large amount of data. To properly train the codebook, the training data should span the range as broad as possible on different speakers, including ranges in age, accent, gender, speaking rate, levels and other variables [18]. This is to ensure the system's applicability to a wide range of speakers.

We show a comparison between the speaker-dependent recognition and speaker-independent recognition in Figure 6.4. The overall accuracy of all 10 words for both speaker-dependent and independent cases are shown below. Although all the speaker-independent cases have lower rates compared with the dependent ones, the degradation for the ARMA lattice modeling is not as much as the AR modeling. This implies that the features extracted from ARMA lattice modeling characterized the speech signal more effectively than the AR modeling. In some degree ARMA lattice representation might emphasize perceptually important speaker-independent features of the signal, and deemphasize speaker-dependent characteristics.

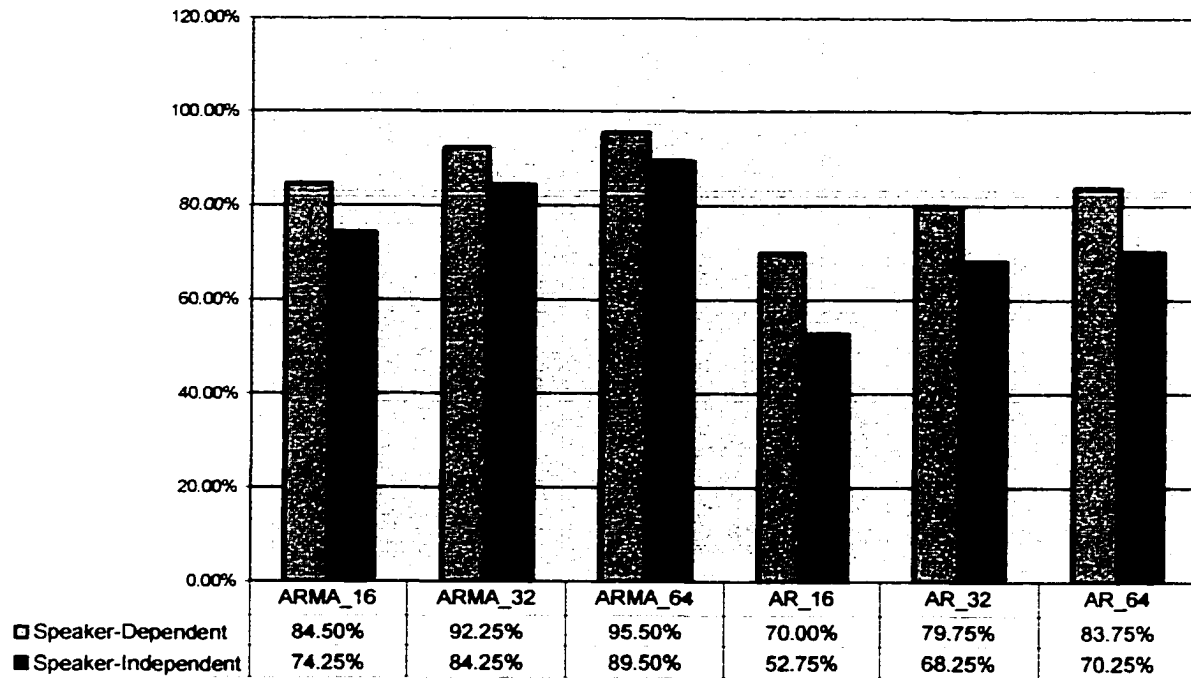


Figure 6.4 Speaker-dependent vs speaker-independent

To make it fair, we also investigate a 13th order AR model and AR lattice model in our experiments, then the feature vector dimension is the same for both AR and ARMA models. The lattice reflection coefficients can be obtained from LP coefficients using a set of recursion equations (Durbin's algorithm) [23]. The following table lists their overall recognition accuracy for the case of codebook size 64.

Table 6.3 Recognition results for same feature vector dimension

	AR(12)	AR(13)	AR Lattice(13)	ARMA Lattice(6,6)
Speaker- Dependent	83.75%	84.00%	84.25%	95.50%
Speaker- Independent	70.25%	70.25%	70.50%	89.50%

From the table, no improvement is found for speaker-independent recognition when AR order increased by 1. And the 13th order AR lattice model has slight improvements on recognition performance compared with its traditional method. The reflection coefficient is said to be a useful feature domain for speech recognition. Look at the results from all the above modelings, with the same feature vector dimension. ARMA lattice model shows again its superiority on the recognition performance.

Chapter 7

7 Conclusions

7.1 Conclusions

Speech recognition is a difficult problem, largely because of variability associated with speech signals: thus acoustic feature extraction becomes a very important part for a recognition application. The system attempts to model the signal in such a way that, at the level of signal representation, an accurate modeling technique that is able to extract more and important features from signals is developed.

In stead of using conventional LP model to analyze the isolated word speech signal, a new ARMA lattice model was investigated due to its ability of modeling the zeros. The number of operations in this lattice algorithm compares favorably to those in the corresponding direct form pole-zero modeling algorithm. And with the special arrangement, the number of coefficients required in the modeling is reduced too. These advantages make it an efficient method for speech analysis. Moreover, the advances in computer technology and hardware improvement have also directly influenced the model

choice. Since the availability of fast computation has enable large scale processes run within a short amount of time, system speed is not a serious problem anymore.

Two kinds of experiments, speaker-dependent and speaker-independent recognition, have been designed to test the system using the proposed ARMA lattice modeling and neural network pattern recognition technique. With the same codebook size, ARMA (6,6) lattice modeling proves remarkably better in the recognition performances than 12th order AR (LP) modeling in both experiments. The recognition using ARMA lattice modeling with 64 code vectors has the highest overall recognition accuracy of 95.5%.

Comparing all the results of AR and ARMA lattice modeling, the ARMA lattice modeling shows the superiority especially for recognizing the words with nasal sound like no, john, wayne. Their accuracy is about 19% higher than using AR modeling. This proves the capability of taking care of sounds with zeros properties for ARMA modeling. To make the comparison fair, we even tested AR and ARMA models with same coefficients dimension (13). The results of ARMA lattice modeling show a similar accuracy of 11% higher than other AR modeling in speaker-dependent recognition.

In conclusion, ARMA lattice modeling is more suitable than the AR (LP) method for the isolated word speech recognition applications. With its benefit in the hardware implementation, we anticipate this modeling technique will be popular in the area of speech recognition.

7.2 Future Work

The future work of our study of ARMA lattice modeling for speech recognition may be focus on two directions, further investigation on the proposed modeling techniques and using of public database to make recognition results comparable.

In the developing of an appropriate model, model order has to be determined accordingly. To provide guidelines to aid in the choice of the LP order for practical implementation, researchers performed a series of investigation [18]. However, when we constructing ARMA lattice model, the model order is simply chosen to make the vector length as same as typical LP model for the purpose of keeping low computation load. Further experimental evaluation of that value should be done to obtain more evidential support. Moreover, up to this point we have discussed ARMA lattice model mainly in terms of difference equation and recursive correlation formula; i.e., in terms of time domain representations. To make this technique complete, from the view of theory, the frequency domain interpretation of ARMA lattice modeling should be also investigated.

In other side, although we already have had reasonable results to verify the effectiveness of our system, there are some other external parameters that can affect the recognition performance, including the characteristics of the environmental noise and the type and the placement of the microphone. As seen in the experiments, we trained and tested our system using locally collected data and had not been very careful in selecting the training and testing sets. As a result, it was very difficult to compare performance across systems. Nowadays, much effort has gone into the development of large speech corpora for system

development, training and testing. Use of these corpora in future helps to compare the components of a recognizer in a more meaningful way.

Another direction of future work is probably the robust recognition. In our experiments, the data used for both training and testing are supposed to be clean speech. However, most practical recognition applications have to work under more difficult situations, say, the training and testing conditions are different. To solve this problem, improvements can be made at both feature extraction level and pattern recognition level. In the LP analysis, a standard method is to convert the LP coefficients to cepstral coefficients which have been shown to be a more robust, reliable feature set for speech recognition than the LP coefficients [18]. This raises our guess about the possibility of any conversion on ARMA lattice coefficients, therefore to get a more robust system.

8 References

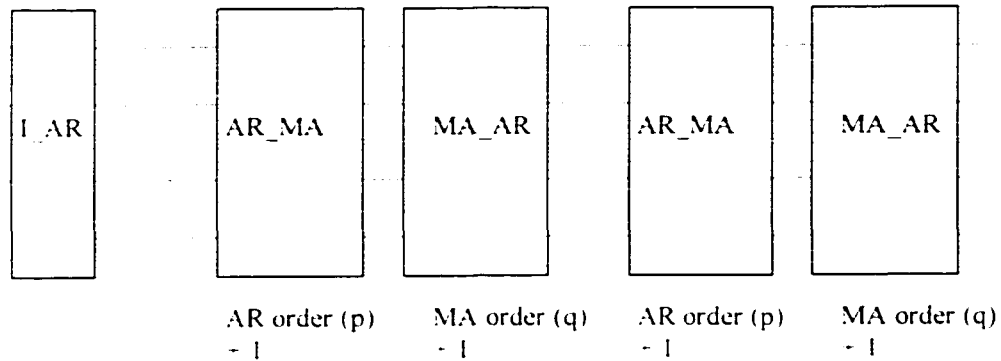
- [1] J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, 1976.
- [2] L. R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signal*, Prentice Hall, Englewood Cliffs, NJ, 1978.
- [3] D. Lee, M. Morf, and B. Friedlander, "Recursive square-root ladder estimation algorithm," *IEEE Tran. Acoust., Speech, Signa Processing*, vol. ASSP-29, pp.627-641, 1981.
- [4] J. L. Flanagan, *Speech Analysis Synthesis and Perception*, 2nd Edition, New York: Wiley, 1968.
- [5] K. T. Assaleh and R. J. Mammone, "New LP-derived features for speaker identification," *IEEE Trans. on Speech and Audio Proc.*, vol. 2, pp. 630-638, Oct. 1994.
- [6] I.-T. Lim and B. G. Lee, "Lossless pole-zero modeling of speech signals," *IEEE Trans. on Speech and Audio Proc.*, vol. 1, no. 3, pp. 269-276, 1993.
- [7] J. G. Proakis, C. M. Rader, F. Ling, and C. L. Nikias, *Advanced Digital Signal Processing*, Macmillan Publishing Company, New York, 1992.
- [8] J. Makhoul, "Stable and efficient lattice methods for linear prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 423-428, Oct. 1977.
- [9] E. Karlsson and M. H. Hayes, "ARMA modeling of time-varying systems with lattice filter," *IEEE ICASSP '86*, Tokyo, pp. 2335-2338, April 1986.
- [10] Y. Miyanaga, N. Nagai and N. Miki, "ARMA digital lattice filter based on new criterion," *IEEE. Trans. Circuits and Systems*, vol. CAS-34, pp. 617-628, June 1987.
- [11] D. Lee, B. Friedlander and M. Morf, "Recursive ladder algorithms for ARMA modeling," *IEEE. Trans. Automatic Control*, vol. AC-27, pp. 753-764, August 1982.
- [12] Y. C. Lim and S. R. Parker, "On the synthesis of lattice parameter digital filter," *IEEE. Trans. Circuits and Systems*, vol. CAS-31, pp. 593-601, July 1984.
- [13] H. K. Kwan, and Y. C. Lui, "Normalized ARMA levison algorithm," *IEEE Trans. Circuits Syst.*, vol CAS-36, no. 3, pp. 383-386, March 1989.

-
- [14] H. K. Kwan, and Y. C. Lui, "Minimal normalized lattice structure for ARMA digital filter realization." *Proceeding of IEEE International Conference on Acoustic, Speech and signal processing*, vol. 3, pp. 1629-1632, May 1991.
- [15] Y. Miyanaga, H. Watanabe, N. Miki and N. Nagai, "A fast calculation algorithm for a parameter estimation of autoregressive and moving average model." *Trans. I. E. C. E.*, J66-A, 10, pp. 1000-1007, Oct. 1983.
- [16] E. Karlsson and M. H. Hayes, "Least squares ARMA modeling of linear time-varying systems: Lattice filter structures and fast RLS algorithm." *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 994-1014, July 1987.
- [17] J. D. Markel and A. H. Gray, Jr., "Roundoff noise characteristics of a class of orthogonal polynomial structures." *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 473-486, 1975.
- [18] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, N.J. 1993.
- [19] A. Gersho, and R. M. Gray, *Vector Quantization and Signal Compression*, Massachusetts, Kluwer Academic Publishers, 1991.
- [20] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, New York, 3rd Edition, 1989.
- [21] T. Kohone, "New developments of learning vector quantization and self-organizing map." *Symposium on Neural Networks, Alliances and Perspectives in Senri 1992*, Osaka, Japan.
- [22] F. K. Soong, "A vector quantization approach to speaker recognition." *ICASSP 1988*, pp. 387-390, 1988.
- [23] J. P. Campbell, Jr., "Speaker recognition: a tutorial." *Proceeding of IEEE*, vol. 85, No. 9, pp. 1437-1462, Sept. 1997.

Appendix

Computational Complexity of ARMA Lattice Algorithm

An ARMA lattice model with minimal arrangement ($p=q$) is shown below:



The computation required for the building blocks is as follows:

I_AR: (5M. 3S)

AR_MA: (10M. 6S)

MA_AR: (10M. 6S)

M---multiplication S---summation

So for an ARMA lattice model of order length $N(p=q)$, the total computation required is:

$$(5M. 3S) + (N/2)*(10M. 6S) + (N/2)*(10M. 6S) = ((5+10N)M. (3+6N)S)$$

VITA AUCTORIS

NAME: Tracy, Xiaoping, LI

PLACE OF BIRTH: CHINA

DATE OF BIRTH: April, 1971.

EDUCATION: B.Eng.
Department of Electrical Engineering
Xian Jiaotong University
Xian, China.
1988-1992.

M.A.Sc.
Department of Electrical & Computer Engineering
University of Windsor
Windsor, Ontario, Canada.
1997-1999