

L_p MODELS IN SPEECH CODING AND
MARKOV CHAINS IN SPEECH
RECOGNITION

By

JAMES LOWELL LANSFORD
//

Bachelor of Science
Auburn University
Auburn, Alabama
1980

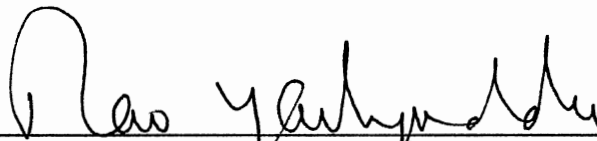
Master of Science
Georgia Institute of Technology
Atlanta, Georgia
1982

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
May, 1988

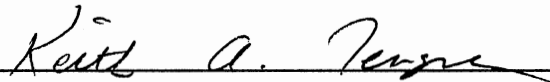
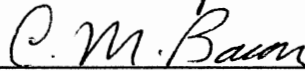
Thesis
1988D
L295A
cop.2

L_p MODELS IN SPEECH CODING AND
MARKOV CHAINS IN SPEECH
RECOGNITION

Thesis Approved:



Thesis Advisor



Dean of the Graduate College

PREFACE

This thesis presents three primary ideas: 1) an extension of the traditional Burg (forward-backward harmonic mean) estimation method from its least squares (L_2 norm) roots to the more general L_p normed case; 2) an adaptive speech coding scheme which attempts to find the optimal p -normed estimator for a given segment of speech; and 3) some experimental results in what might be called explicit Markov models for speech recognition, which is based on work that has been done in so-called “hidden” Markov models.

This work has been supported by the National Security Agency under contract number MDA-904-85-15-0002; I would like to thank Tom Tremain, Rich Dean, and Dave Kemp for their sponsorship and helpful suggestions during the course of this study.

I would like to thank my major advisor, Dr. Rao Yarlagadda, for his encouragement, support, and patience; it has been a pleasure to work with and learn from him. I would also like to thank the members of my committee for their suggestions and guidance: Dr. Keith Teague, Dr. Virgil Marco, Dr. Charles Bacon, and Dr. Randy Reininger.

Like everyone who has reached this point in his education, I could not have gotten here without the love and encouragement of a lot of people. I would like to begin by thanking some of the most special ones: Fay Sanders, for giving me the love of mathematics; Jud and Vera Milburn, for giving me invaluable editorial, financial and moral support; and Larry Holland, for giving me time and motivation to finish. Others who have been a big help along the

way are Joan Armstrong, Joyce Kennamer, Rayford Green and Jewell Brewton.

The most important contributors to my success, however, did not live to see me complete the degree – my parents. This thesis is dedicated to the memory of my mother, who died shortly before its completion; she always stressed the value of education and hard work – this thesis could not have been done without that example.

Finally, I wish to thank the love of my life, Lynn Milburn Lansford, for being a constant source of love, support, patience, encouragement, motivation, and stability. When all the world seemed topsy-turvy and the deadlines seemed impossible, she helped me sort it all out and get the job done. Dissertations are hard on wives and families and I deeply appreciate their understanding and acceptance of the long hours and missed activities.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
Overview of Speech Coding Issues	1
Overview of Speech Recognition Issues	5
Chapter Outline	7
II. LINEAR PREDICTIVE MODELS FOR SPEECH CODING . .	9
Introduction	9
Classical Least Squares Solution	11
Maximum Likelihood Models	16
III. SOLUTION METHODS FOR L_p -NORMED MODELS	20
Introduction	20
Properties of the L_p Solution	21
L_1 Specific Procedures	30
Residual Steepest Descent Algorithm	35
IV. GENERALIZED BURG ALGORITHM FOR L_p MODELING. . .	44
Use of Weighted Median for L_1 Burg	45
Derivation of L_p Burg Algorithm	47

Chapter	Page
V. OPTIMAL ADAPTIVE L_p MODELS FOR SPEECH	53
Statistical Properties of Speech Residuals	54
Kurtosis as a Measure of Dispersion	60
Kurtosis of Speech Residuals	61
Adaptive Speech Coding Using Kurtosis in LPC-10	65
VI. SPECTRAL ESTIMATION OF SPEECH USING L_p MODELS..	72
Review of Spectral Estimation	72
Experimental Results	73
VII. MARKOV CHAINS IN SPEECH RECOGNITION.	87
Pattern Recognition Concepts	87
Speech Recognition	96
Overview of Cochlear Modeling	100
Properties of Markov Chains	103
Historical Notes	114
The Viterbi Algorithm (VA)	116
Use of VA on Functions of Markov Chains	123
Summary of Computer Tools	129
Results	130
VIII. CONCLUSIONS AND AREAS FOR FURTHER RESEARCH..	140
CITED REFERENCES	144

LIST OF FIGURES

Figure	Page
1. Lattice Architecture	14
2. PDF of Speech With Gamma and Laplace Densities	18
3. P-Gaussian PDF's for Several Values of P	19
4. a) L_2 Error Surface	23
b) L_2 Contour Plot	23
5. a) $L_{1.5}$ Error Surface	24
b) $L_{1.5}$ Contour Plot	24
6. a) L_1 Error Surface	25
b) L_1 Contour Plot	25
7. a) $L_{0.5}$ Error Surface	26
b) $L_{0.5}$ Contour Plot	26
8. Example of Solution Space for L_1 Problem	29
9. a), b), c), and d), Simplex Tableaus for Example	32,33
10. Solution Locus for Linear Programming	34
11. a) Solution Locus Using L_2 at Start	39
b) Solution Locus Using (0,0) at Start	40
12. a) Solution Locus for L_2	41
b) Solution Locus for $L_{1.5}$	41
c) Solution Locus for $L_{0.5}$	42
13. Speech Waveform: /a/ from "add"	55
14. Residual for /a/	56
15. Histogram of Residual for /a/	56
16. Speech Waveform: /ts/ from "cats"	57
17. Residual for /ts/	57
18. Histogram of Residual for /ts/	58
19. Speech Waveform: /z/ from "thieves"	58
20. Residual for /z/	59

Figure		Page
21.	Histogram of Residual for /z/	59
22.	Kurtosis as Related to PDF Shape	62
23.	Kurtosis of Residual for /a/	63
24.	Kurtosis of Residual for /ts/	64
25.	Kurtosis of Residual for /z/	65
26.	Block Diagram of LPC-10 Transmitter	67
27.	Block Diagram of Optimal p LPC-10 Transmitter	68
28.	Kurtosis of Frames: "Thieves who rob friends deserve jail"	70
29.	Optimal p-value used for "Thieves" Sentence	71
30.	Speech Waveform: "Cats"	76
31.	Spectral Surface, Covariance Model, "Cats", $p=-0.1$	77
32.	Spectral Surface, Covariance Model, "Cats", $p=0.5$	77
33.	Spectral Surface, Covariance Model, "Cats", $p=1.0$	78
34.	Spectral Surface, Covariance Model, "Cats", $p=1.5$	78
35.	Spectral Surface, Covariance Model, "Cats", $p=2.0$	79
36.	Spectral Surface, Covariance Model, "Cats", $p=3.0$	79
37.	Spectral Surface, Burg Model, "Cats", $p=-0.1$	80
38.	Spectral Surface, Burg Model, "Cats", $p=0.5$	80
39.	Spectral Surface, Burg Model, "Cats", $p=1.0$	81
40.	Spectral Surface, Burg Model, "Cats", $p=1.5$	81
41.	Spectral Surface, Burg Model, "Cats", $p=2.0$	82
42.	Spectral Surface, Burg Model, "Cats", $p=3.0$	82
43.	Spectral Surface, Weighted Median (L_1) Model, "Cats"	83
44.	Spectral Surface, 20 dB SNR, Covariance Model, "Cats", $p=2.0$	84
45.	Spectral Surface, 20 dB SNR, Burg Model, "Cats", $p=2.0$	84
46.	Spectral Surface, 20 dB SNR, Covariance Model, "Cats", $p=1.0$	85
47.	Spectral Surface, 20 dB SNR, Burg Model, "Cats", $p=1.0$	85
48.	Comparison of L_2 Covariance and L_1 Covariance Spectra	86
49.	Comparison of L_2 Burg and L_1 Burg Spectra	86
50.	Minimum Probability of Error Decision Function	89
51.	Decision Surfaces for Gaussian Distributions	91
52.	Variation of a Feature Cluster Over Time	93
53.	Syntactical Pattern Recognition	94

Figure	Page
54. Dynamic Programming	96
55. Block Diagram of Harpy System	98
56. Cochlear Modeling	102
57. Signal Flow Graph for Example	105
58. Example of an Ergodic Chain	106
59. Example of Transient States	107
60. Frequency Response of Diagonal Terms of $\pi(z)$	112
61. Trellis for Example	118
62. Solution Path and Distances for Example	121
63. Recognition System Block Diagram	122
64. Waveform Encoding Effects	124
65. Reflection Coefficient Feature Extraction	127
66. Spectral Model Feature Extraction	129
67. Time and Zero Crossing Signals – “FIVE”	132
68. Markov Transition Matrix for “FIVE”	132
69. Signal Flow Graph – “FIVE” (Zero Crossing)	133
70. Frequency Response – “FIVE” (Zero Crossing)	133
71. Recognition Sequence for Example	134
72. Signal Flow Graph – “ONE” (Reflection Coefficient)	136
73. Frequency Response – “ONE” (Reflection Coefficient)	136
74. Confusion Matrix – “ONE” (Reflection Coefficient)	137
75. Signal Flow Graph – “ONE” (Spectral Model)	138
76. Frequency Response – “ONE” (Spectral Model)	138
77. Confusion Matrix – “ONE” (Spectral Model)	139

CHAPTER I

INTRODUCTION

Overview of Speech Coding

Transmission of speech over a digital communications channel poses a number of problems not found in the analog world; among the most significant of these is information bandwidth. If one assumes that analog voice requires 4kHz of bandwidth, the most popular coding technique (μ -law, $\mu=255$) used by the telecommunications industry would require 64 kbits/sec. or about 64 kHz of bandwidth, assuming a modest one Hz/bit transmission scheme. Clearly, this 16-fold bandwidth increase is undesirable; to reduce the information rate, various forms of data compression are employed, adding to the complexity of the system. These compression systems fall into a range of general categories (Jayant and Noll [1984], p. 9): 1) Waveform compression, such as ADPCM (Rabiner and Schafer [1978], p. 226); 2) Parametric models, such as short-time Fourier transform or linear predictive models (Rabiner and Schafer [1978], p. 396); and 3) literal recognition of speech with text to speech reconstruction at the receiver (Flanagan, 1972).

The tradeoffs involved in each can be summarized as follows:

Coding type	Complexity	Comments
Uncoded	Very Low	<ul style="list-style-type: none"> • At least 64 kbits/sec required for transmission • Clarity excellent • Voice characteristics (inflection, tonality) preserved
Waveform (ADPCM, etc.)	Low	<ul style="list-style-type: none"> • 8-32 kbits/sec • Clarity greatly diminished at lower rates • Voice characteristics preserved
Parametric (LPC, short term Fourier)	Moderate	<ul style="list-style-type: none"> • 2.4-9.6 kbits/sec • Clarity better than with equivalent waveform coder rate • Voice characteristics not preserved as well as in waveform coding
Recognition	High	<ul style="list-style-type: none"> • <100 bits/sec • Synthesis from text to speech; inflection and tonality lost

In many applications, especially telephony, retention of the speakers' speech characteristics is important for normal, natural conversation. This, in conjunction with the fact that 2400-4800 bits/sec can be transmitted relatively easily over commercial phone lines, implies that a good parametric model would seem to be of great importance.

Given that a parametric model is desired, two problems remain:

- 1) Choice of an approach used to develop a model, and
- 2) Determination of the "best" model for a given approach.

Obviously, a number of books could be and have been written on these topics; it is the intent here to address some new issues in the determination of a suitable model and computation of the model parameters, where a statistical approach to the modelling problem is used as a basis.

Use of Parametric models for Speech Coding

A widely used model in statistics is the autoregressive (AR) model, which is also called the "linear predictive" model.

The name "linear prediction" comes from the formulation of the model; the basic assumption is that the next sample in a sequence can be estimated from a linearly weighted combination of previous samples. In more mathematical terms, if we assume a sequence $x(n)$, $n=1,2,3,\dots,N$ then the following relation would apply (Rabiner and Schafer, 1978, p. 399):

$$\tilde{x}(n) = \sum_{i=1}^m a_i x(n-i+1) \quad (1.1)$$

where m is the order of the predictor and the a_i are predictor coefficients, which must be calculated from the data. The process of extracting the a_i from the data $x(n)$ is also called "deconvolution".

Parameters of a linear predictive model are usually calculated by minimizing the sum of the squared residuals (or errors), where the residual is

the difference between the actual data and the model. This is also referred to as an L_2 model, where the “2” indicates the residual terms are raised to the second power before summing. In general, one could raise the residual terms to some arbitrary p -th power and minimize this quantity; this is called the L_p normed model. Values of p other than two may offer some advantages in a number of ways; several studies have been done using the L_p model in other areas, particularly seismic data processing, with particular interest in L_1 models (Taylor, *et al*, 1979 and Yarlagadda, *et al*, 1985). Qualitatively, the L_1 (absolute value) solution tends to ignore outliers while the L_2 model tries to satisfy all points as best it can; values of p between one and two blend these characteristics somewhat. Thus, the L_1 model is “best” for a “long-tailed” probability density function, such as Laplacian.

In fact, it can be shown (Pham and DeFigueiredo, 1987) that for a given value of p , the L_p model will be the maximum likelihood estimator for residuals whose distribution is of the form $e^{-|r|^p}$. If the nature of the residuals can be characterized to show they are of this form for speech, then the L_p models should be more optimal in a statistical sense.

Also of importance in parametric modeling is stability of the model. If the AR parameters a_i are considered coefficients of a recursive filter structure, the synthesis filter would have a transfer function

$$P(z) = \frac{G}{1 - \sum_{i=1}^m a_i z^{-i}} \quad (1.2)$$

where G is the overall gain and z is the z -transform operator.

For a synthesis system to properly reproduce the speech, this filter must be stable – the poles of $P(z)$ must lie inside the unit circle. The basic linear

predictive model only assures stability under special circumstances; synthesis structures called lattice filters allow a stable model to be computed under all circumstances.

This work will review techniques for calculation of the L_2 then examine the methods that can be used to calculate the L_p model. An extension of the L_2 lattice techniques to the more general L_p case will next be considered. Optimal selection of p for speech coding applications will also be discussed, where optimality will be with reference to a measure of the distribution of the residuals. Finally, spectral estimation of speech using L_p models will be discussed and experimental results will be described.

Overview of speech recognition issues

The final chapter of this thesis is a brief introduction to the field of sequential decision processes. A sequential decision process (or sequential classifier) has the attractive property of being "real-time"; that is, a signal can be classified as it is received, without having to be captured and processed. Sequential classifiers also have the interesting property that they follow a "most likely" path as they attempt to arrive at a decision, which should allow "soft" decisions to be made and the "lines of reasoning" to be traced, both of which are important artificial intelligence concepts.

The primary purpose of the chapter is to explore some current ideas in sequential classification as applied to speech recognition. Why speech recognition? Well, tracking the temporal patterns of the spectrum of the speech signal would seem to be a reasonable way to extract the information

in a speech signal (Levinson *et al*, 1983). Other signals have a structure that is amenable to sequential processing, of course; convolutional codes used in digital communication are especially suited to sequential processing techniques.

Some words of caution are in order before proceeding; speech recognition is an extremely difficult problem. As Flanagan (1976) stated: "The speech typewriter...in its full fledged form...may not happen this century, if ever." Pierce(1969), in a refreshingly self-critical letter to the *Journal of the Acoustical Society of America*, was equally candid:

It is hard to gauge the the success of an attempt at speech recognition even when the statistics are given. In general, it appears that recognition around 95% correct can be achieved for clearly pronounced isolated words from a chosen small vocabulary (the digits, for example) spoken by a few chosen talkers. Better results have been attained for one talker. Performance has gone down dramatically as the vocabulary was expanded, and appreciably as the number and variety of talkers were increased. It is not easy to see a practical, economically sound application for speech recognition with this capability.

While speech recognition systems have improved somewhat since 1969, the point of Pierce's letter is well taken; the difficulty of the problem will likely prevent any quantum leaps in performance. Some researchers (Flanagan *et al*, 1980) believe that grandiose projects should be minimized and that research should concentrate on incremental improvements in existing techniques. Many research programs purport to be the one that will finally solve the problem, and while good results have been obtained, natural man-machine communication is still far in the future. This investigation makes

no claim that any technique is THE answer; rather, it is an attempt to gain a better understanding of sequential detection and its application to signal recognition - an incremental increase in existing techniques.

The recognition techniques described in this thesis are based on work performed at Bell Laboratories (Rabiner *et al*, 1983) and at IBM (Baker, 1975; Kaneko and Dixon, 1983) with some significant differences.

Chapter Outline

Chapter II reviews existing methods for selecting a linear predictive model for speech; it describes both forward prediction as well as lattice (forward-backward) techniques. Maximum likelihood models are also briefly discussed.

Chapter III describes the properties of and solution strategies for what are called p-normed forward prediction models; these are a generalization of traditional least squares models (L_2) for the L_p case.

Chapter IV gives a derivation of a new algorithm for calculating the Burg estimators based on a p-normed error criterion. Some previous results for the L_1 case are described and the L_p approach is developed by extending previous work in L_2 and L_1 modeling.

Chapter V contains some experimental results using an adaptive coding scheme for a digital speech communications system. The kurtosis of the residual is used to arrive at a "near" optimal p-normed model for a given segment of speech.

Explicit spectral models are the subject of Chapter VI; the linear

prediction inverse filter $P(z)$ is treated as a spectral estimate of the time signal $x(n)$. Speech signals are examined by a variety of spectral estimators, where performance both with and without additive noise is demonstrated.

A different subject, speech recognition, is the topic of Chapter VII. Some experimental results in Markov modeling and sequential classification are described.

Finally, conclusions and future work comprise Chapter VIII, the final chapter, which is followed by references.

CHAPTER II

LINEAR PREDICTIVE MODELS FOR SPEECH CODING

Preliminaries

The usual method of solving for the predictor coefficients is to minimize the squared error between the predicted values (\tilde{x}) and the actual values (x). There are a number of ways to formulate this problem, however, and there are some implicit assumptions in the least squared error approach. To begin with, the linear predictor given previously can be written as a matrix equation as follows (Yarlagadda, *et al*, 1985):

$$\begin{bmatrix} x_1 & 0 & \cdots & 0 \\ x_2 & x_1 & \cdots & 0 \\ x_3 & x_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ x_m & x_{m-1} & \cdots & x_1 \\ \cdots & \cdots & \cdots & \cdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-m} \\ x_N & x_{N-1} & \cdots & x_{N-m+1} \\ 0 & x_N & \cdots & x_{N-m+2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & x_N \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ \cdots \\ x_{m+1} \\ \cdots \\ x_N \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \quad (2.1)$$

or

$$\underline{X}\underline{a} = \underline{y}$$

\underline{X} (qxm)

\underline{a} (mx1)

\underline{y} (qx1)

N is the window width

q = N+m-1

This is called the autocorrelation formulation of the one-step ahead linear prediction problem for the least squares (L_2) solution; the sequence is arbitrarily set to zero outside of the analysis interval.

Another way to set up the prediction equations is called the covariance formulation in L_2 (shown below); here no assumption is made about the length of the sequence.

$$\begin{bmatrix}
 x_m & x_{m-1} & \cdots & x_1 \\
 x_{m+1} & x_m & \cdots & x_2 \\
 x_{m+2} & x_{m+1} & \cdots & x_3 \\
 \cdots & \cdots & \cdots & \cdots \\
 x_{2m} & x_{2m-1} & \cdots & x_{m-1} \\
 \cdots & \cdots & \cdots & \cdots \\
 x_{N-m} & x_{N-m-1} & \cdots & x_{N-2m} \\
 \cdots & \cdots & \cdots & \cdots \\
 x_{N-2} & x_{N-3} & \cdots & x_{q-1} \\
 x_{N-1} & x_{N-2} & \cdots & x_q
 \end{bmatrix}
 \begin{bmatrix}
 a_1 \\
 a_2 \\
 \cdots \\
 a_m
 \end{bmatrix}
 =
 \begin{bmatrix}
 x_{m+1} \\
 x_{m+2} \\
 x_{m+3} \\
 \cdots \\
 x_{2m+1} \\
 \cdots \\
 x_{N-m+1} \\
 \cdots \\
 x_{N-1} \\
 x_N
 \end{bmatrix}
 \quad (2.2)$$

or $\mathbf{X}\mathbf{a} = \mathbf{y}$

N is the window width

\mathbf{X} (qxm)

\mathbf{a} ($mx1$)

\mathbf{y} ($qx1$)

$q = N-m-1$

Solution of either of these formulations for the predictor coefficients a_i involves solution of an overdetermined system of equations, which cannot be solved in a conventional sense without some further assumptions. One approach, which yields the least squares solution, is to multiply both sides by \mathbf{X}^t , then recognize that $\mathbf{X}^t\mathbf{X}$ is square and (usually) invertible. In this thesis, it is assumed that \mathbf{X} is of maximum rank. In certain applications \mathbf{X} may not be of full rank; in such cases, methods such as diagonal loading are used where $[\epsilon\mathbf{I} + \mathbf{X}^t\mathbf{X}]$ is used instead of $\mathbf{X}^t\mathbf{X}$, where \mathbf{I} is the identity matrix, and ϵ is some small

positive number. In the usual case however, the least squared error solution to the set of linear prediction equations can be written as:

$$\underline{a} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \underline{y} \quad (2.3)$$

The squared error can now be derived by defining the error or residual sequence \underline{r} :

$$r_i = y_i - (\mathbf{X}\underline{a})_i \quad (2.4)$$

$$E_2 = \sum_i r_i^2 = \sum_i (y_i - (\mathbf{X}\underline{a})_i)^2 \quad (2.5)$$

While this solution may seem compact and well formulated, it may not yield a good model for speech. Why? Predictor coefficients derived from the least squares method are the maximum likelihood estimators only if the sequence \underline{r} is Gaussian [Mendel, 1987, p. 93]. This assumption may not be true for speech, especially voiced speech, which does not sound like Gaussian noise at all. As Geary [1947] says, “all texts should state: Normality is a myth, there has never been, and never will be, a normal distribution”. These issues will be addressed in more detail later in this thesis. For now, we shall review techniques for solving the L_2 problem and use these as an introduction to L_p solution methods.

Classical Least Squares Solution Techniques

As noted in equation (2.3), the form of the least squares solution is $\underline{a} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \underline{y}$, regardless of whether the problem was formulated as an autocorrelation or as a covariance problem (although obviously the \mathbf{X} and \underline{y}

are different in each case). Calculation of the \underline{a} vector could be done by using direct techniques, but this is almost never done because inversion of the $\mathbf{X}^t\mathbf{X}$ matrix directly is too time consuming except for the case where m , the size of the square $\mathbf{X}^t\mathbf{X}$ matrix, is small. Instead, the $\mathbf{X}^t\mathbf{X}$ matrix is inverted by exploiting special properties it possesses. In particular, if the autocorrelation problem has been posed, the $\mathbf{X}^t\mathbf{X}$ matrix will be Toeplitz (symmetric with all values along a given diagonal equal); and the \underline{a} vector can be solved for quite efficiently using the Levinson-Durbin procedure (Rabiner and Schafer, 1978, p. 411). In this case, the $\mathbf{X}^t\mathbf{X}$ matrix turns out to be a matrix of autocorrelation values ; thus the system of equations to be solved is of the following form:

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(m-1) \\ R_n(1) & R_n(0) & \cdots & R_n(m-2) \\ \cdots & \cdots & \cdots & \cdots \\ R_n(m-1) & R_n(m-2) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \cdots \\ R_n(m) \end{bmatrix} \quad (2.6)$$

where $R_n(k)$ is the autocorrelation of the signal $x(n)$ at the k -th lag which is given by:

$$R_n(k) = \sum_{i=1}^{N-k} x(n+i) x(n+i+k) \quad 0 \leq k \leq m \quad (2.7)$$

In series form, the system of equations can be written:

$$\sum_{i=1}^m a_i R_n(|j-i|) = R_n(j) \quad 1 \leq j \leq m \quad (2.8)$$

Rabiner and Schafer (1978, p. 411) offer a concise and clear summary of Durbin's method; the details of the algorithm will not be described here.

If the problem is to solve the covariance formulation, the $\mathbf{X}^t\mathbf{X}$ matrix is symmetric but not Toeplitz. An efficient means exists to find \underline{a} , although it is not as fast as the Levinson-Durbin algorithm. This technique is called the

square root method or Cholesky decomposition. For this case the $\mathbf{X}^t\mathbf{X}$ matrix is a matrix of values that has the properties of a covariance matrix, and the system of equations takes the following form:

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \cdots & \phi_n(1,m) \\ \phi_n(2,1) & \phi_n(2,2) & \cdots & \phi_n(2,m) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_n(m,1) & \phi_n(m,2) & \cdots & \phi_n(m,m) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \cdots \\ \phi_n(m,0) \end{bmatrix} \quad (2.9)$$

which can be written in matrix notation as $\Phi\mathbf{a}=\mathbf{\psi}$. The entries in Φ are expressed by (Rabiner and Schafer, 1978, p. 403)

$$\phi_n(i,k) = \sum_{j=1}^N x(n+j-i) x(n+j-k) \quad \begin{matrix} 1 \leq i \leq m \\ 0 \leq k \leq m \end{matrix} \quad (2.10)$$

Solution of the equations can efficiently be accomplished using the Cholesky decomposition as mentioned above, which factors the Φ matrix into the form \mathbf{VDV}^t , where \mathbf{V} is lower triangular and \mathbf{D} is diagonal (Graybill, 1976, p. 231).

Another issue mentioned earlier is filter stability, which is important for analysis/synthesis systems. The L_2 autocorrelation method is always guaranteed to yield a stable $P(z)$ – all poles are inside the unit circle; the L_2 covariance method may not always yield a stable polynomial (Makhoul, 1977). A relatively recent solution to this problem can be found in lattice structure models; in these cases, filter stability can be assured using a wide variety of constraint functions. Figure 1 shows the lattice architecture, where the k_j are called reflection coefficients (they are also the same as partial autocorrelation coefficients in L_2 – often the k_j are used with a minus sign) (Makhoul, 1977). For an m -th order predictor, if all $|k_j| < 1$, then this is a necessary and sufficient condition for stability of $P(z)$ (Markel and Gray, 1976,

p. 98). Thus, any solution method that can be shown to always have all $|k_j| < 1$ will always yield stable predictors.

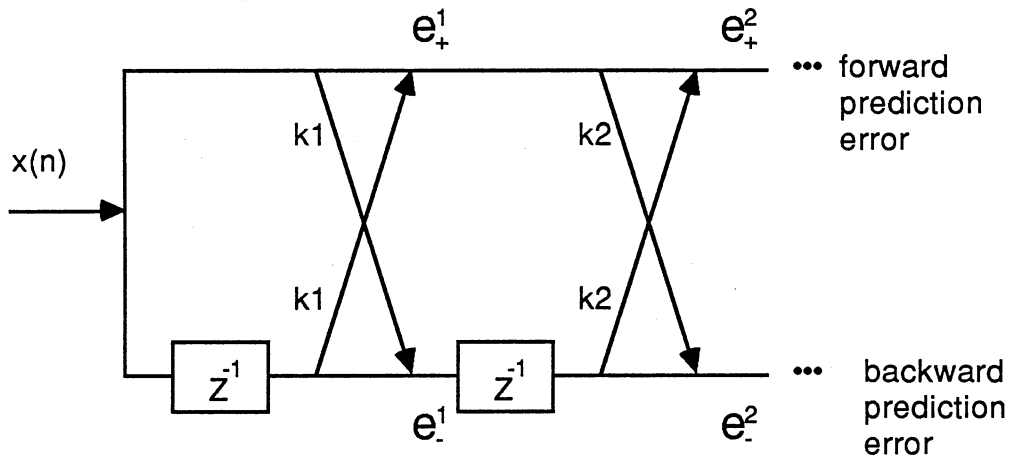


Figure 1: Lattice Architecture

The L_2 solution that assures stability is due to Burg; the derivation will lay the groundwork for the L_p solution. At the j -th stage of the lattice, the Burg algorithm requires minimization of the quantity (Burg, 1968):

$$E_B^j = \sum_{n=L}^N |e_+^j(n)|^2 + \sum_{n=L}^N |e_-^j(n)|^2 \quad (2.11)$$

where N is the number of data samples, $e_+^j(n)$ is the forward error at sample n for the j^{th} stage, $e_-^j(n)$ is the backward error at sample n for the j^{th} stage, and L is the lower limit of the analysis interval. Viswanathan and Makhoul (1979) use two different values for L : $L=j+1$ or $L=m+1$, with the former being used more commonly in speech.

We want to find a reflection coefficient k_{j+1} that allows the errors to be a first order recursion from one lattice stage to the next.

$$e_+^{j+1}(n) = e_+^j(n) + k_{j+1} e_-^j(n-1) \quad (\text{forward error}) \quad (2.12)$$

$$e_-^{j+1}(n) = e_-^j(n-1) + k_{j+1} e_+^j(n) \quad (\text{backward error}) \quad (2.13)$$

and

$$e_+^0(n) = e_-^0(n) = x(n) \quad (\text{initial error is data sequence}) \quad (2.14)$$

Solution for the desired k_{j+1} is accomplished by setting the derivative of the error to zero and rearranging terms. At the $j+1$ st stage, the error is:

$$E_B^{j+1} = \sum_{n=L}^N [e_+^{j+1}(n)]^2 + \sum_{n=L}^N [e_-^{j+1}(n)]^2 \quad (2.15)$$

Substituting in the recursion expressions for the error terms yields:

$$\begin{aligned} E_B^{j+1} &= \sum_{n=L}^N [e_+^j(n) + k_{j+1} e_-^j(n-1)]^2 \\ &\quad + \sum_{n=L}^N [e_-^j(n-1) + k_{j+1} e_+^j(n)]^2 \end{aligned} \quad (2.16)$$

Taking the partial derivative with respect to k_{j+1} and setting it to zero gives the expression:

$$\begin{aligned} 0 &= \sum_{n=L}^N e_+^j(n) e_-^j(n-1) + k_{j+1} \sum_{n=L}^N e_-^j(n-1)^2 \\ &\quad + \sum_{n=L}^N e_-^j(n-1) e_+^j(n) + k_{j+1} \sum_{n=L}^N e_+^j(n)^2 \end{aligned} \quad (2.17)$$

The reflection coefficient can now be solved for directly.

$$k_{j+1} = \frac{-2 \sum_{n=L}^N e_+^j(n) e_-^j(n-1)}{\sum_{n=L}^N [e_-^j(n-1)^2 + e_+^j(n)^2]} \quad (2.18a)$$

which is, in vector form,

$$k_{j+1} = - \left[\begin{pmatrix} \underline{e}_-^j \\ \underline{e}_+^j \end{pmatrix} \begin{pmatrix} \underline{e}_-^j \\ \underline{e}_+^j \end{pmatrix}^t + \begin{pmatrix} \underline{e}_+^j \\ \underline{e}_-^j \end{pmatrix} \begin{pmatrix} \underline{e}_+^j \\ \underline{e}_-^j \end{pmatrix}^t \right]^{-1} \left[\begin{pmatrix} \underline{e}_+^j \\ \underline{e}_-^j \end{pmatrix} \begin{pmatrix} \underline{e}_-^j \\ \underline{e}_+^j \end{pmatrix}^t + \begin{pmatrix} \underline{e}_-^j \\ \underline{e}_+^j \end{pmatrix} \begin{pmatrix} \underline{e}_+^j \\ \underline{e}_-^j \end{pmatrix}^t \right] \quad (2.18b)$$

From the expression in equation (2.18a), it follows that any $|k_{j+1}| < 1$ since the ratio is of the form $-2ab/(a^2 + b^2)$, and since $(a+b)^2 > 0$ this implies that $a^2 + b^2 > |2ab|$.

In general, the Burg solution will be different from either the autocorrelation or the covariance solution. What we have, then, are three formulations of the same problem that yield three different answers. We shall soon see that each formulation can be solved for a p-normed solution, further complicating the choice of the best model to use.

A topic that should also be addressed when discussing optimal L_p models is that of efficiency of the model. In particular, the optimal estimator in a statistical sense is the one which is the maximum likelihood estimator for a given probability density function (pdf) (Mendel, 1987, p. 91). In the case of a linear predictive (autoregressive) process, the distribution of the residual sequence should be used to establish the maximum likelihood predictor (Rice and White, 1964).

While there is an infinite number of distributions that the residual could be drawn from, the selection process can be greatly simplified if some logical assumptions can be made. A good starting point would be examination of the

distribution of the speech signal and the residual from linear predictive analysis. Paez and Glisson (1972) report that the pdf of speech is closer to a gamma or Laplace density, as shown in Figure 2; while it is difficult to infer the distribution of the residuals from this, we shall see later that experimental results bear this out. It can be shown that the maximum likelihood estimator for a Laplacian distribution is the L_1 estimator (Hogg, 1977); this gives a strong motivation for investigating models other than L_2 for speech.

Pham and deFigueiredo (1987) show that both the Gaussian and Laplacian distributions are special cases of a more general family of distributions called p-Gaussian. These pdf's are of the form $e^{-|r|^p}$, where we define r as the residual sequence and p is generally in the range $1 \leq p \leq \infty$. Thus, $p=2$ gives a Gaussian distribution and the Laplacian distribution results when $p=1$. An infinite set of distributions can be generated by varying p ; some of these are shown in Figure 3, which also indicates that as p goes to infinity, the uniform distribution results. The p-Gaussian family of pdf's, then, represent a wide range of likelihood functions. Pham and deFigueiredo (1987) point out that the L_p predictor is maximum likelihood for the corresponding p-Gaussian pdf; any computational procedure which would allow calculation of an L_p model for a variety of values of p would give great flexibility in choosing an efficient model for speech, especially if a criterion can be found that will choose an optimal value for p .

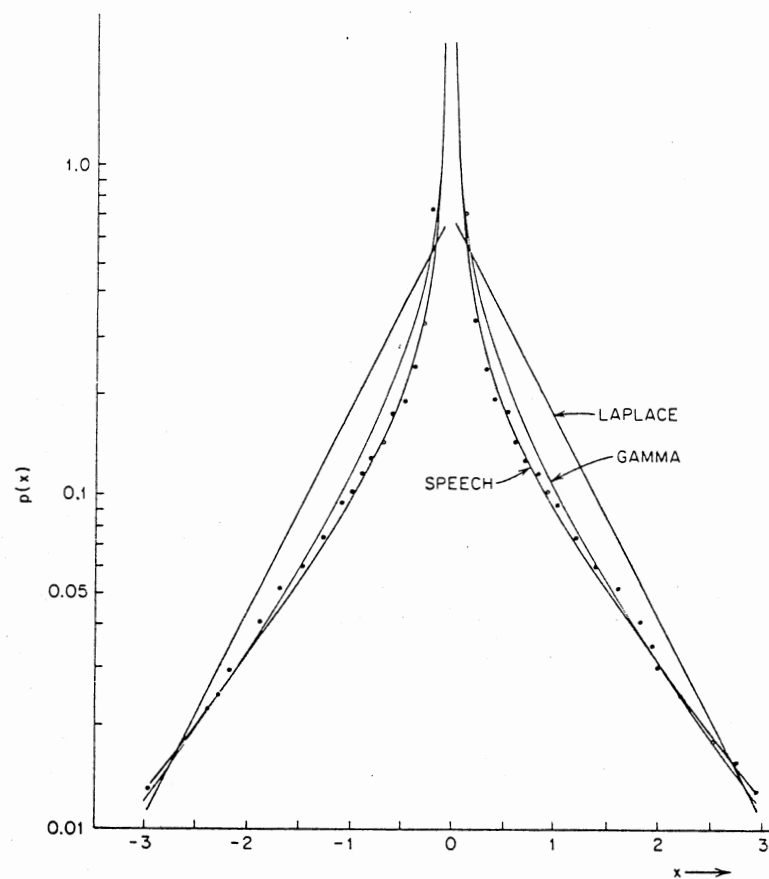


Figure 2: PDF of Speech With Gamma and Laplace Densities

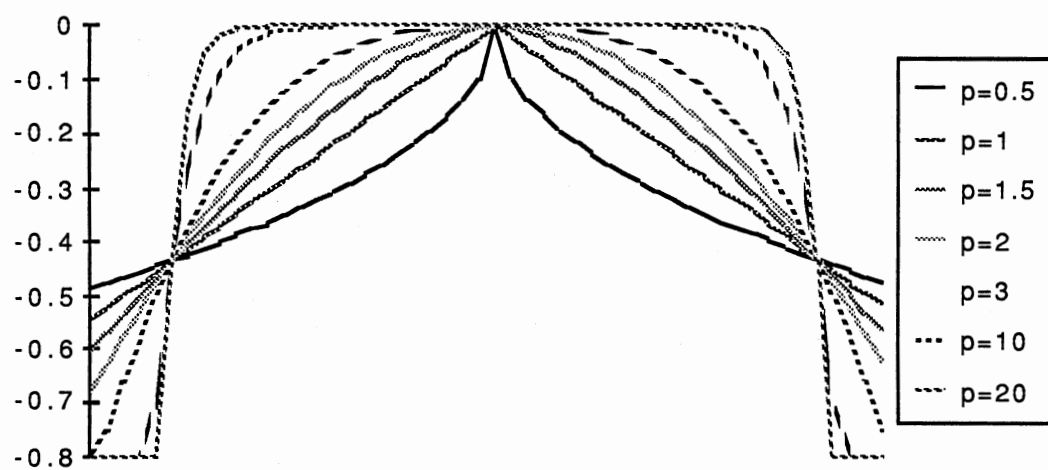


Figure 3: p-Gaussian pdf's for several values of p

CHAPTER III

SOLUTION AND PROPERTIES OF THE L_p LINEAR PREDICTOR

Introduction

As mentioned previously, the L_2 solution is only one of an infinite number of solutions to the overdetermined system of linear prediction equations. In particular, one can define a p-normed error criterion as follows:

$$E_p = \sum_i |y_i - (X\hat{a})_i|^p = \sum_i |r_i|^p \quad (3.1)$$

where $r_i = y_i - (X\hat{a})_i$ and E_p is to be minimized for a given value of p.

For L_2 , the solution exists and can be found using Durbin's method (for the autocorrelation formulation) or Cholesky's decomposition (for the covariance formulation) (Rabiner and Schafer, 1978, p.407). For other values of p, some specific procedures have been developed. For example, the L_1 solution can be calculated using linear programming (as will be shown shortly), as can the L_∞ solution (Pham and deFigueiredo, 1987). In general, the p-normed solution to the linear prediction problem can be efficiently solved by using the residual steepest descent (RSD) algorithm, among others. Some of these techniques will be described later in this chapter.

Properties of the L_p Solution

To demonstrate the properties of the L_p solution, consider taking the impulse response of some known linear system; we would expect the recovered model coefficients to correspond to the parameters of the underlying linear system if the model order is sufficiently high and no noise is present. As an example, consider the difference equation:

$$x(n)=s(n) + 1.6 x(n-1) - 0.8 x(n-2) \quad (3.2)$$

which has a unit sample response of

1, 1.6, 1.76, 1.536, 1.05, 0.4506, -0.119, ...

This is a linear system with conjugate poles at $0.8 \pm j0.4$ in the Z-plane; it corresponds to two vectors whose magnitude is approximately 0.9. Thus, while this is a stable system, the poles are fairly near the unit circle. The linear prediction equations for the autocorrelation formulation for the case of $m=2$ and $q=6$ are then:

$$\begin{bmatrix} 1 & 0 \\ 1.6 & 1 \\ 1.76 & 1.6 \\ 1.536 & 1.76 \\ 0 & 1.536 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 1.76 \\ 1.536 \\ 0 \\ 0 \end{bmatrix} \quad (3.3)$$

which is the practical problem of a truncated sequence. The ideal solution should correspond to the system parameters (1.6, -0.8)^t.

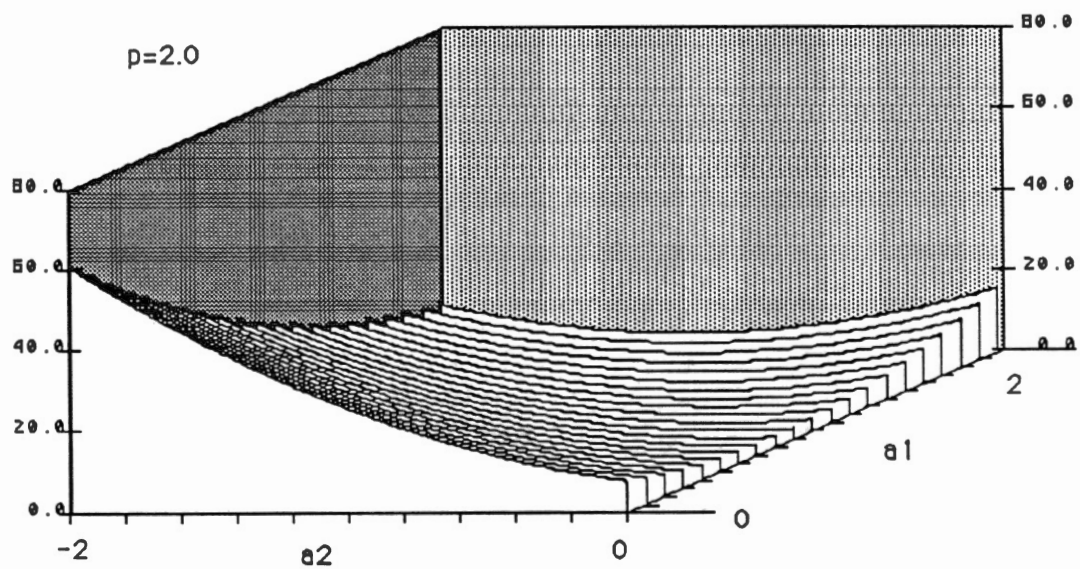
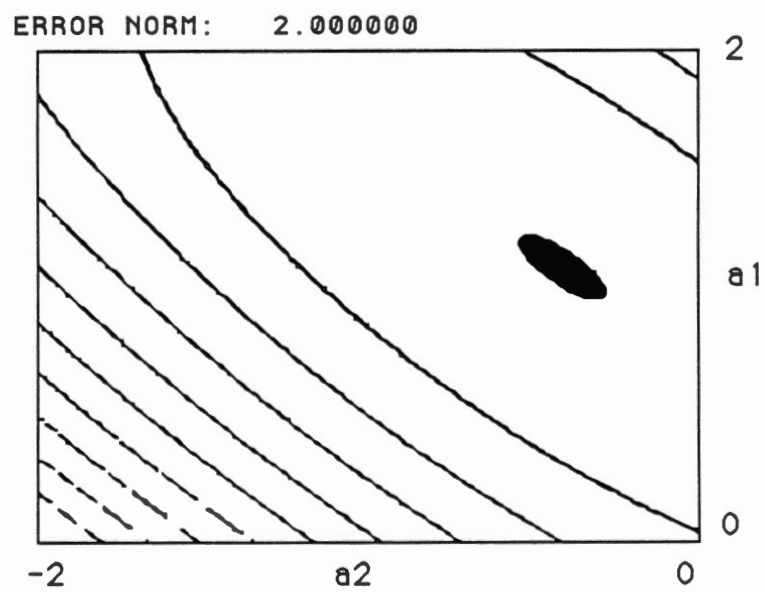
This system of equations translates into the following minimization problem when solving for the p -normed error:

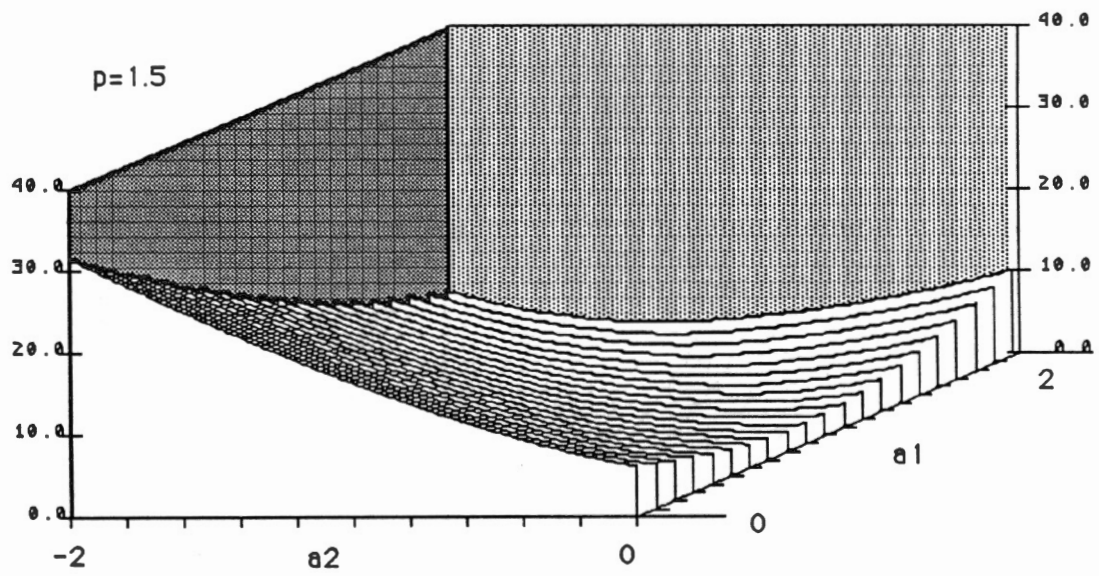
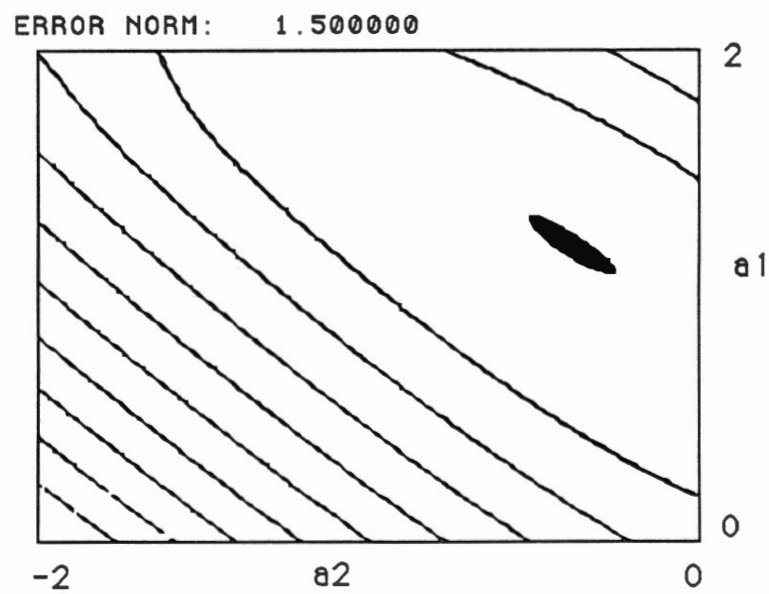
Minimize:

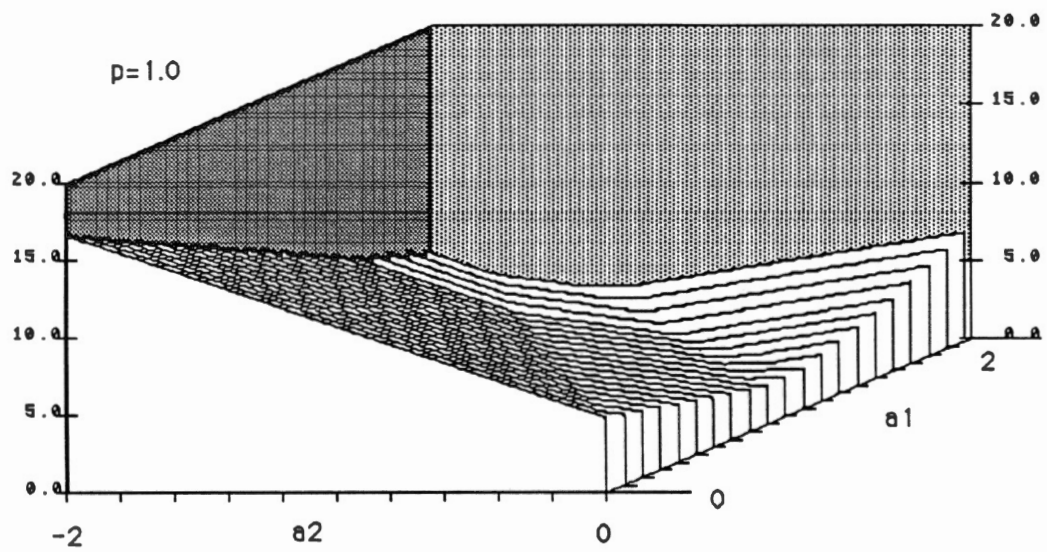
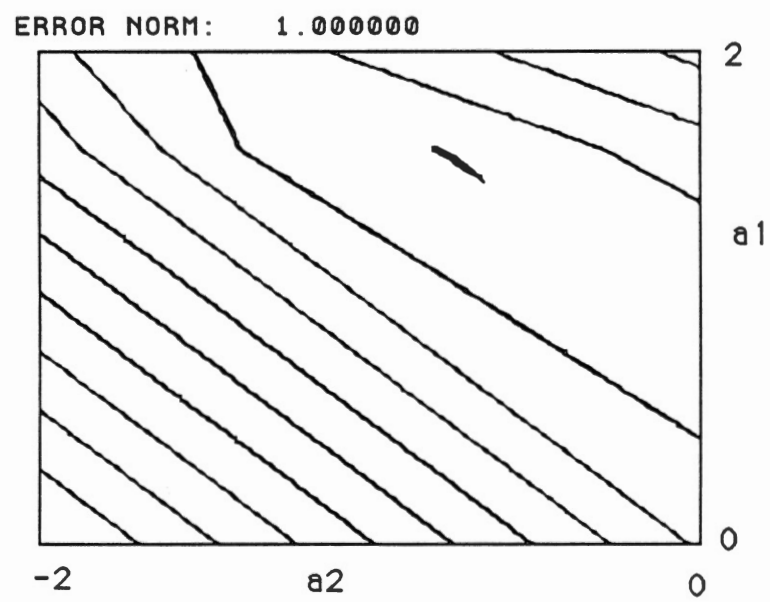
$$\begin{aligned} \phi(a_1, a_2) = & (|1.6 - a_1|)^p + (|1.76 - 1.6a_1 - a_2|)^p + (|1.536 - 1.76a_1 - 1.6a_2|)^p \\ & + (|-1.536a_1 - 1.76a_2|)^p + (|-1.536a_2|)^p \end{aligned} \quad (3.4)$$

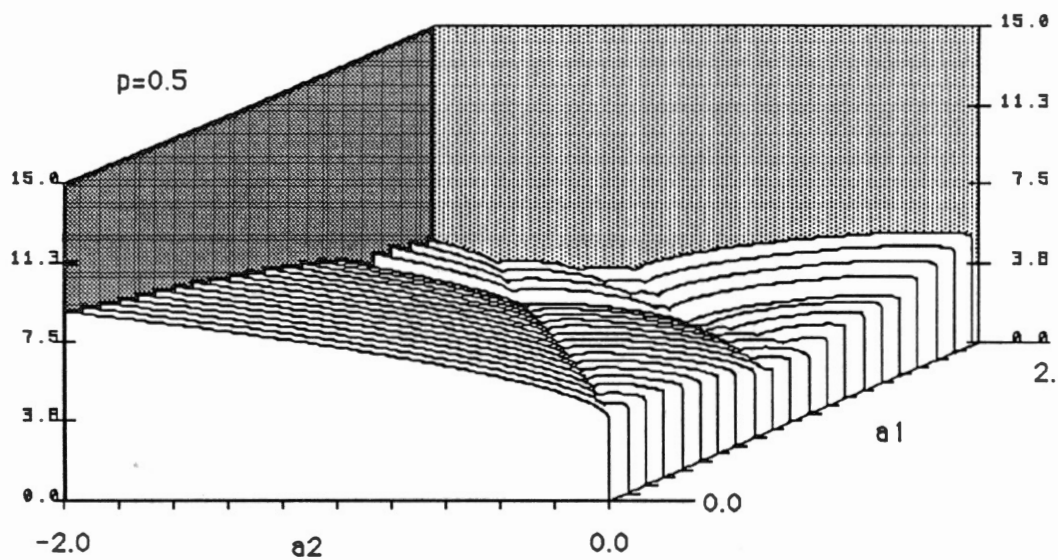
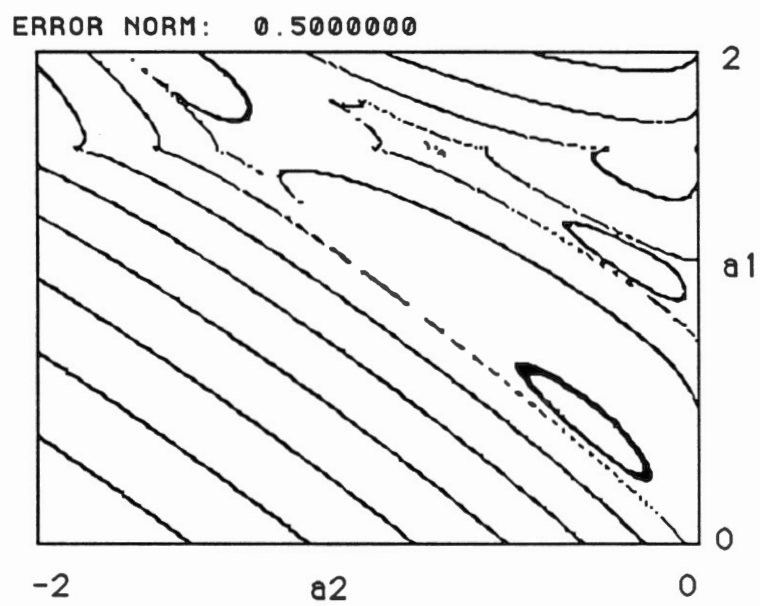
where the values of a_1 and a_2 are unconstrained.

Before exploring techniques for finding the minimum of the error function $\phi(a_1, a_2) = E_p$, a discussion of some graphical methods may help clarify the problem. Two types of plots will be employed; the first is the error surface generated by ϕ while a_1 and a_2 are varied over a range of p values; this surface should have a "valley" for a minimum to exist. While these surfaces show important properties such as convexity, the second type of graph, which is a contour plot of the error function ϕ , will be useful when looking at solution trajectories. The contours are drawn at ten equally spaced values of $\phi(a_1, a_2)$ over the rectangular region bounded by $0 \leq a_1 \leq 2$ and $-2 \leq a_2 \leq 0$. The residual at any point in these plots will point perpendicular to the contour lines shown; thus, the system will have a solution whenever there is an open circle or a "spot" that indicates a minimum. Figure 4a) is the L_2 error surface, while b) is its corresponding contour plot. Likewise, Figures 5a) and b) are the $L_{1.5}$ error surface and contour plot, and Figures 6a) and b) are the plots for the L_1 case. Finally, Figures 7a) and b) give the error surface and contour plots for the $L_{0.5}$ case. The first three cases clearly have solutions in the vicinity of $\underline{a} = (1.6, -0.8)$, but when $p=0.5$, something goes awry; while the function has a global minimum, it also appears to have numerous other local minima. The case for $p=0.5$ does not, in fact, correspond to a linear normed space (not globally convex - not all points in the space can be represented as a linear combination of the basis vectors), but it may still have some usefulness. In this case, the residual does not always point in the direction of the global solution.

Figure 4a: L_2 Error SurfaceFigure 4b: L_2 Contour Plot

Figure 5a: $L_{1.5}$ Error SurfaceFigure 5b: $L_{1.5}$ Contour Plot

Figure 6a: L_1 Error SurfaceFigure 6b: L_1 Contour Plot

Figure 7a: $L_{0.5}$ Error SurfaceFigure 7b: $L_{0.5}$ Contour Plot

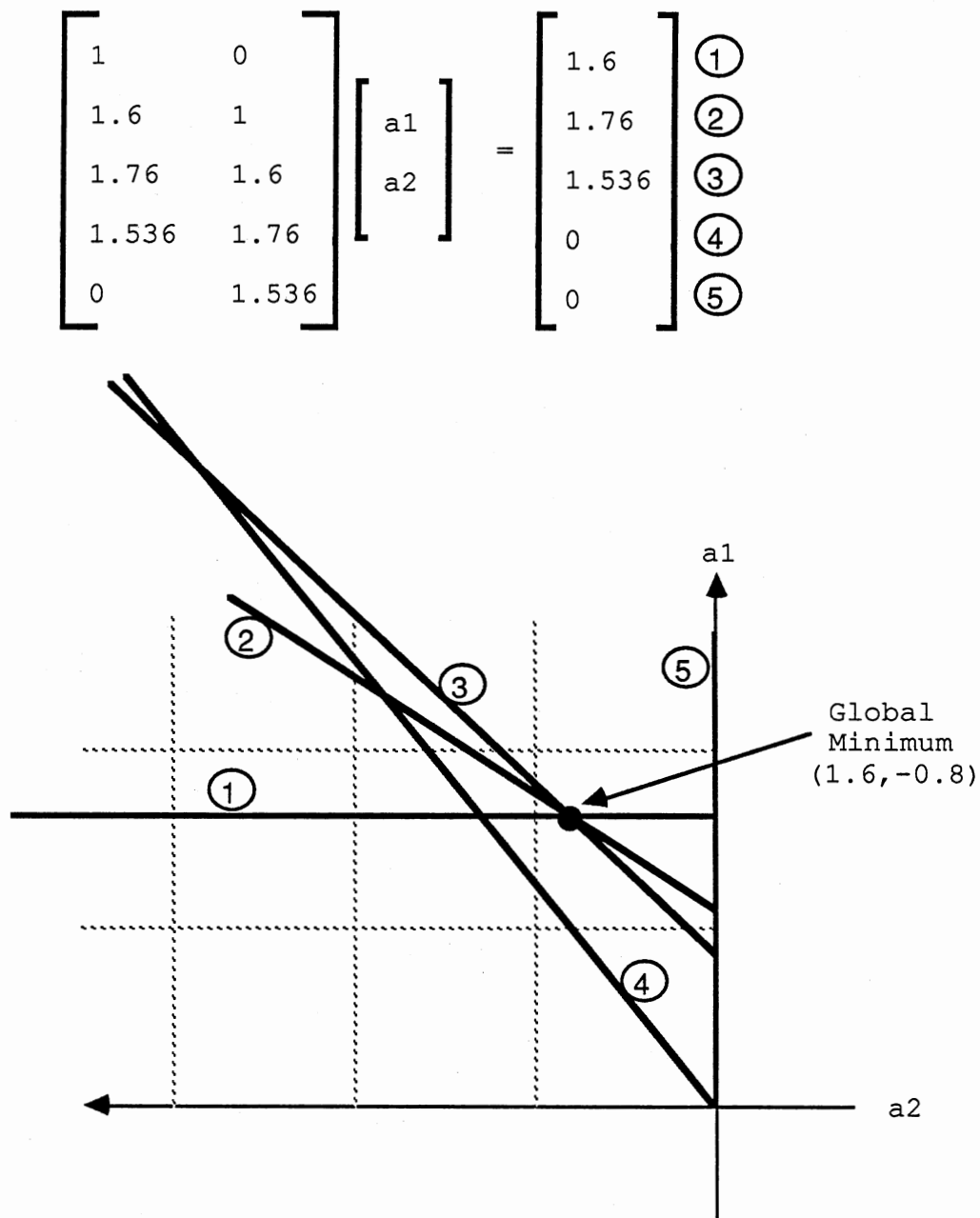
The L_2 autocorrelation method is always guaranteed to yield a stable $P(z)$ – all poles inside the unit circle; the L_2 covariance method is not (Makhoul, 1977). In the general L_p problem, no formulation mentioned so far can assure stability except for the autocorrelation form with $p=2$; other values of p may yield unstable models, no matter which method is used. In particular, the autocorrelation model is always stable for $2 \leq p < 3$, but there may exist some p_0 in the interval $1 < p_0 < 2$ for which the prediction filter may not be stable. In that case, stability is not assured for any model generated in the range $1 < p < p_0$ (Bednar, *et al*, 1986). Filter stability can be assured by using a different formulation of the linear prediction problem – the lattice or Burg algorithm, which will be addressed in Chapter IV. As a simple example of the instability problem, consider the following scalar case:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} a = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 0 \end{bmatrix} \quad (3.5)$$

The L_2 solution to this problem is $50/55 = 10/11$, which is less than one, indicating a stable solution. The L_1 solution, on the other hand, is $5/4$, which yields an unstable predictor.

Before proceeding, some words about the L_1 solution are in order: $\phi(a_1, a_2)$ becomes linear when $p=1$, indicating that the minimum occurs at a vertex of the polytope bounded by ϕ (Jeter, 1986, p. 68). For this, the L_1 case, we want to find a set of two equations that are satisfied exactly and yield the minimum sum of absolute errors. A “brute force” approach would require examining ${}_5C_2 = 10$ sets of equations (or ${}_qC_2$ in general). If the five equations are sketched in the $a_1 - a_2$ plane, the solution space in Figure 8 results, which

shows the possible solution locations at the intersections of each pair of lines. By calculating the value of $\phi(a_1, a_2)$ at each intersection, the minimum absolute error solution can be found. Solution for this global minimum in the general L_1 case can be much more efficiently found using linear programming techniques (Barrodale and Roberts, 1973).

Figure 8: Example of solution space for L_1 problem

Use of Linear Programming Techniques for L_1 normed models

Linear programming was the only technique available for L_1 deconvolution until relatively recently. Linear programming is based on the idea that if all variables are allowed to take on only positive values in an underdetermined system of linear constraint equations with positive solutions, then a systematic procedure can be used to arrive at the minimum of a linear objective function (assuming a finite minimum exists). This is true because positivity of the variables and constants forces the region bounded by the objective function and the constraints to be a convex polyhedron (Thie, 1979, p. 98). Given some starting point on a vertex of the polyhedron, the "simplex algorithm" can be used to move from vertex to vertex until an optimum solution is reached.

Solution of the deconvolution problem with an L_1 objective function can be put into a linear programming form so that the simplex method can be used to solve for the \underline{a} vector (Barrodale and Roberts, 1974). The first problem is that the L_p problem is an overdetermined system of equations while linear programming requires an underdetermined system. The process of casting the L_p problem in standard form will rectify this.

As noted previously, the system of equations to be analyzed is $\mathbf{X}\underline{a}=\underline{y}$. The solution \underline{a} is found by a constrained minimization procedure that minimizes E_p , where p equals one for this case. To cast this in a linear programming form, we shall change the formulation slightly:

$$\begin{aligned} &\text{Minimize } \sum_i |r_i| \\ &\text{Subject to } [X\underline{a} - \underline{y}] = \underline{r} = \text{Col}(r_1, r_2, \dots, r_q) \end{aligned} \quad (3.6)$$

which merely states that the p-norm residual is to be minimized. Here, the unknowns \underline{a} and \underline{r} are "unconstrained" - they can be either positive or negative. Since convexity (and hence ability to find a solution) requires positive unknowns only, the vectors \underline{r} and \underline{a} must be split into the difference between two positive vectors: $\underline{r}^+ - \underline{r}^-$ and $\underline{a}^+ - \underline{a}^-$. The problem now is (Taylor, *et al*, 1979):

$$\begin{aligned} &\text{Minimize } \sum_i |r_i^+| + \sum_i |r_i^-| \\ &\text{Subject to: } X(\underline{a}^+ - \underline{a}^-) + (\underline{r}^- - \underline{r}^+) = \underline{y} \end{aligned} \quad (3.7)$$

$$\underline{a}^+, \underline{a}^- \geq 0, \quad \underline{r}^-, \underline{r}^+ \geq 0$$

In matrix form, this is:

$$\begin{aligned} &\text{Minimize: } \begin{pmatrix} \mathbf{0}^t & \mathbf{0}^t & \mathbf{1}^t & \mathbf{1}^t \end{pmatrix} \begin{bmatrix} \underline{a}^+ \\ \underline{a}^- \\ \underline{r}^- \\ \underline{r}^+ \end{bmatrix} \\ &\text{Subject to: } (\mathbf{X} - \mathbf{X} \quad \mathbf{I} \quad -\mathbf{I}) \begin{bmatrix} \underline{a}^+ \\ \underline{a}^- \\ \underline{r}^- \\ \underline{r}^+ \end{bmatrix} = \underline{y} \end{aligned} \quad (3.8)$$

$$\underline{a}^+, \underline{a}^- \geq 0, \quad \underline{r}^-, \underline{r}^+ \geq 0$$

where $\mathbf{1}$ is a column vector of all ones and $\mathbf{0}$ is a vector of all zeroes.

Figure 9 a) shows this matrix written in tableau form for the simplex algorithm, where the objective function ($\sum_i |r_i^+| + \sum_i |r_i^-|$) is written along the bottom row of the tableau. The 1-normed error will then be the negative of the value in the lower right corner when the algorithm stops.

Using the simplex algorithm, three pivots will be performed to bring the tableau into the halting position (see Thie [1979] for details of the simplex and dual simplex methods) as shown in Figures 9 b), c) and d). At this point, the solution $a_1^+ = 1.6$, $a_2^- = 0.8$ can be read directly from the tableau. This implies that the solution is $\underline{a} = (1.6, -0.8)^t$.

a_1^+	a_2^+	a_1^-	a_2^-	r_1^-	r_2^-	r_3^-	r_4^-	r_5^-	r_1^+	r_2^+	r_3^+	r_4^+	r_5^+	y
10.00	0.00	-10.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00	0.00	0.00	0.00	15.00
15.00	10.00	-15.00	-10.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00	0.00	0.00	17.50
17.50	15.00	-17.50	-15.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00	0.00	15.36
15.36	17.50	-15.36	-17.50	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00	0.00
0.00	15.36	0.00	-15.36	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00
-58.96	-58.96	58.96	58.96	0.00	0.00	0.00	0.00	0.00	2.00	2.00	2.00	2.00	2.00	-48.96

$\sum |r_i|$

Figure 9a: Initial Simplex Tableau

0.00	-11.46	0.00	11.46	1.00	0.00	0.00	-0.65	0.00	-1.00	0.00	0.00	0.65	0.00	15.00
0.00	-8.33	0.00	8.33	0.00	1.00	0.00	-1.04	0.00	0.00	-1.00	0.00	1.04	0.00	17.50
0.00	-4.17	0.00	4.17	0.00	0.00	1.00	-1.15	0.00	0.00	0.00	-1.00	1.15	0.00	15.36
1.00	1.15	-1.00	-1.15	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	-0.07	0.00	0.00
0.00	15.36	0.00	-15.36	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	-1.00	0.00
0.00	8.50	0.00	-8.50	0.00	0.00	0.00	3.84	0.00	2.00	2.00	2.00	-1.84	2.00	-48.96

Figure 9b: Simplex Tableau After First Pivot

0.00	-1.00	0.00	1.00	0.09	0.00	0.00	-0.06	0.00	-0.09	0.00	0.00	0.06	0.00	1.40
0.00	0.00	0.00	0.00	-0.73	1.00	0.00	-0.57	0.00	0.73	-1.00	0.00	0.57	0.00	5.96
0.00	0.00	0.00	0.00	-0.36	0.00	1.00	-0.91	0.00	0.36	0.00	-1.00	0.91	0.00	9.54
1.00	0.00	-1.00	0.00	0.10	0.00	0.00	0.00	0.00	-0.10	0.00	0.00	0.00	0.00	1.50
0.00	0.00	0.00	0.00	1.34	0.00	0.00	-0.87	1.00	-1.34	0.00	0.00	0.87	-1.00	21.45
0.00	0.00	0.00	0.00	0.75	0.00	0.00	3.35	0.00	1.25	2.00	2.00	-1.35	2.00	-36.95

Figure 9c: Simplex Tableau After Second Pivot

0.00	-1.00	0.00	1.00	0.11	0.00	-0.06	0.00	0.00	-0.11	0.00	0.06	0.00	0.00	0.80
0.00	0.00	0.00	0.00	-0.50	1.00	-0.52	0.00	0.00	0.50	-1.00	0.52	0.00	0.00	0.00
0.00	0.00	0.00	0.00	-0.40	0.00	1.10	-1.00	0.00	0.40	0.00	-1.10	1.00	0.00	10.50
1.00	0.00	-1.00	0.00	0.10	0.00	0.00	0.00	0.00	-0.10	0.00	0.00	0.00	0.00	1.50
0.00	0.00	0.00	0.00	1.69	0.00	-0.96	0.00	1.00	-1.69	0.00	0.96	0.00	-1.00	12.29
0.00	0.00	0.00	0.00	0.21	0.00	1.49	2.00	0.00	1.79	2.00	0.51	0.00	2.00	-22.78

-L₁ Error

Figure 9d: Final Simplex Tableau

Referring to the contour maps shown previously, the simplex algorithm moves from the origin to the final solution (1.6, -0.8) along the path given in

Figure 10. Since the system of equations is explicitly evaluated at each step, note that the solution is exact in the sense that it satisfies two equations exactly, although much more computation is required to achieve this.

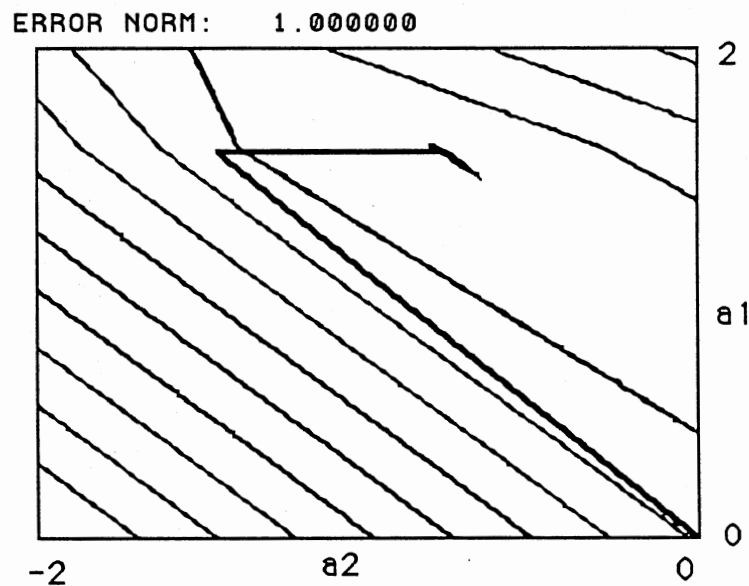


Figure 10: Solution Locus for Linear Programming

Each point of the simplex tableau could require as many as $6 \times 15 = 90$ multiply/add operations, although faster algorithms exist, as will be discussed shortly. Thus, this example required about 270 operations. Other methods can be employed to achieve approximately the same result with far fewer calculations; as m and q become large, the simplex method requires too many computations and too much memory to be useful (Yarlagadda, *et al*, 1985).

Improvements to the simplex algorithm have, in some cases, dramatically improved the speed of the algorithm; the two techniques mentioned most often in the literature are Khachyan's algorithm (see Aspvall and Stone, 1979) and Karmarkar's algorithm (Karmakar, 1984). Both techniques achieve performance improvements by going through the convex solution polytope rather than along its vertices. For more information on Karmakar's algorithm, the reader is referred to Rockett and Stevenson (1987) as well as Strang (1986), both of which give lucid descriptions of the algorithm. It differs from the simplex method primarily in that the locus of intermediate solutions does not remain on the polytope but rather on a hypersphere drawn from within the solution space. Use of this method allows solution of the linear programming problem in $O(m^k)$ (polynomial time) rather than $O(k^m)$ (exponential time) (Rockett and Stephenson, 1987)

Even with these improvements, the fact still remains that linear programming can only solve the L_1 or L_∞ problem; the general L_p problem requires use of other computation techniques, which will be considered shortly.

Use of Residual Steepest Descent Algorithm for L_p Linear Prediction

To find a general L_p solution to the linear prediction equations, we shall consider the RSD algorithm, as described in Yarlagadda, *et al* (1985). This algorithm is a steepest descent method with an adaptive stepsize. For the situation at hand, the deconvolution problem $\mathbf{X} \underline{\mathbf{a}} = \underline{\mathbf{y}}$ can be solved in the p -

normed sense for the vector \underline{a} by the following algorithm:

Residual Steepest Descent Algorithm

0. Calculate initial \underline{a}

a) Use previous \underline{a}

b) Use L_2 solution (McCormick and Sposito ,1976)

ITERATE OVER K

1. Calculate $\underline{r}(k) = (X\underline{a}(k)) - \underline{y}$ (residual vector)

2. Let $\gamma_i(k) = \text{SGN}(r_i(k)) |r_i(k)|^{p-1} \quad i=1, \dots, q$

3. Minimize $E(k)$ w.r.t. Δ_k where

$$E(k) = \| \underline{r}(k) - \Delta_k X (X^t X)^{-1} X^t \gamma(k) \|_p$$

This can be solved directly in a p-normed sense or using iteratively reweighted least squares (IRLS).

4. $\underline{a}(k+1) = \underline{a}(k) - \Delta_k (X^t X)^{-1} X^t \gamma(k)$

5. If Δ_k is "sufficiently small", solution has converged; otherwise, go to step one.

This algorithm converges for $1 \leq p \leq 3$ but does not converge to a true L_1 solution in all cases. For most practical applications, however, the algorithm converges to a correct solution within acceptable limits.

Use of the RSD algorithm on the example problem for the L_1 solution will now be illustrated.

The initial L_2 solution $\underline{a}(0) = (1.116, -0.4133)^t$ yields residual and gamma vectors of :

$$\underline{r} = \begin{bmatrix} -0.4840 \\ -0.3879 \\ -0.2333 \\ 0.9866 \\ -0.6349 \end{bmatrix} \quad \underline{\gamma} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

(3.9)

The next step is the minimization given in step three; we shall use IRLS, which can be summarized as follows:

Let $\underline{x}' = X (X^t X)^{-1} X^t \underline{y}(k)$ (vector derived from quantities in RSD)

$a' = \Delta_j$ (at the j^{th} iteration)

$\underline{y}' = \underline{r}$ (from equation 3.9)

$\underline{r}'(j) = \underline{y}' - \underline{x}' a'$

We wish to minimize

$$E_p = \sum_{n=1}^q |y'_n - x'_n a'|^p \quad (3.10)$$

Now let \mathbf{W} be a diagonal matrix whose entries are defined as

$$W_{ii}(j) = \begin{cases} |r'_{ii}(j)|^{p-2} & |r'_{ii}(j)| > \epsilon \\ \epsilon^{p-2} & |r'_{ii}(j)| < \epsilon \end{cases} \quad (3.11)$$

where ϵ is a small positive number, typically 0.001, and the $W_{ii}(j)$ are normalized such that the largest value is one. Then,

$$a'(j+1) = (\underline{x}'^t \mathbf{W}(j) \underline{x}')^{-1} \underline{x}'^t \mathbf{W}(j) \underline{y}' \quad (3.12)$$

Note that a' is a scalar and that the quantity on the right is the ratio of two scalars.

In the IRLS part of the algorithm, the a' (Δ_k back in the original problem) may not go to zero but is merely required to reach a minimum. When a' stabilizes, the equivalent Δ value is returned to the RSD algorithm. For the example,

$$\mathbf{r}' = \mathbf{y}' - \mathbf{a}'\mathbf{x}' = \begin{bmatrix} -0.4840 \\ -0.3880 \\ -0.2330 \\ 0.9866 \\ -0.6399 \end{bmatrix} - \mathbf{a}'(0) \begin{bmatrix} -0.279 \\ -0.489 \\ -0.560 \\ -0.505 \\ -0.066 \end{bmatrix} \quad (3.13)$$

where $\mathbf{a}'(0)$ can be picked arbitrarily. If we let $\mathbf{a}'(0) = 0$, the IRLS algorithm will calculate $\mathbf{a}'(1) = 3.727$; continuing in this fashion, \mathbf{a}' stabilizes to the second decimal place after eight iterations at a value of 0.417. It has been observed that the L_2 solution is a good starting point for the IRLS algorithm.

Going back to the RSD algorithm, this value of $\Delta_1 = \mathbf{a}' = 0.417$ yielded a new $\mathbf{a}(1) = (1.232, -0.394)^t$ which is closer to the “true” value of $(1.6, -0.8)^t$.

On the second pass through the loop, $\Delta_2 = 0.006$ was calculated and yielded $\mathbf{a}(2) = (1.1222, -0.3954)^t$. At this point, the algorithm was terminated since Δ had converged to less than 0.1 (a typical value). For this solution, the residual vector is:

$$(\mathbf{y} - \mathbf{X}\mathbf{a}) = \mathbf{r} = \begin{bmatrix} -0.368 \\ -0.184 \\ 0.000 \\ 1.200 \\ -0.607 \end{bmatrix}_{L_1} \quad (3.14)$$

where the last two values are attempts to predict outside the rectangular window and can be disregarded. The three values inside the window were predicted quite closely; compare this to the L_2 residual:

$$\mathbf{r} = \begin{bmatrix} -0.484 \\ -0.388 \\ -0.233 \\ 0.987 \\ -0.635 \end{bmatrix}_{L_2} \quad (3.15)$$

which shows the effect of “outliers” on the coefficients and residual.

An insightful way to analyze the performance of RSD is to use the contour map and plot the locus of intermediate solutions as the algorithm iterates. Figure 11 a) shows the solution trajectory for L_1 when the algorithm is started at the L_2 solution $(1.116, -0.4133)^t$, while 11 b) shows an initial solution of $(0, 0)^t$. Clearly, the choice of initial solution is not critical, although one or two extra iterations may be required.

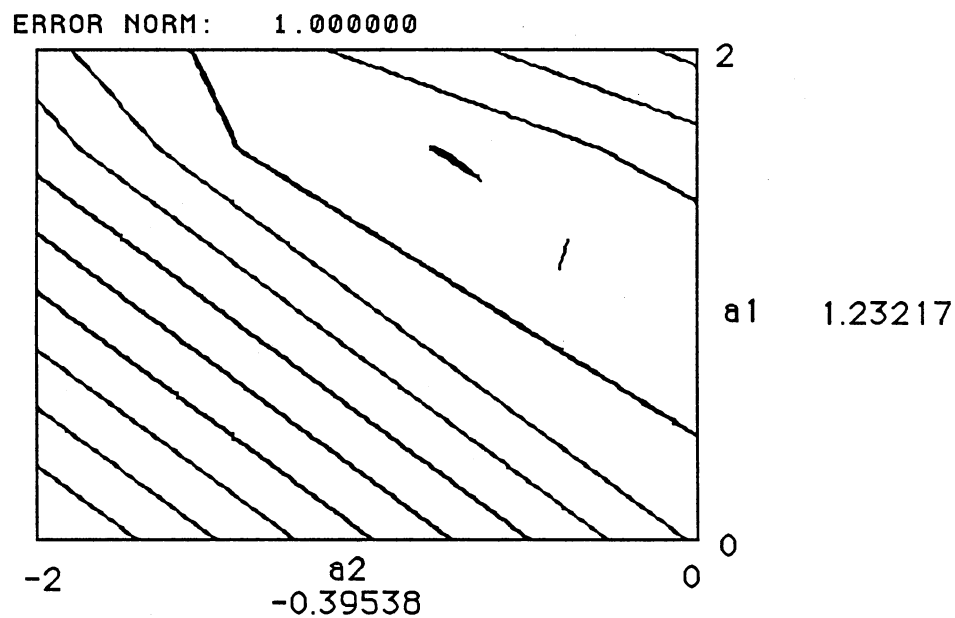


Figure 11a: Solution Locus Using L_2 at Start

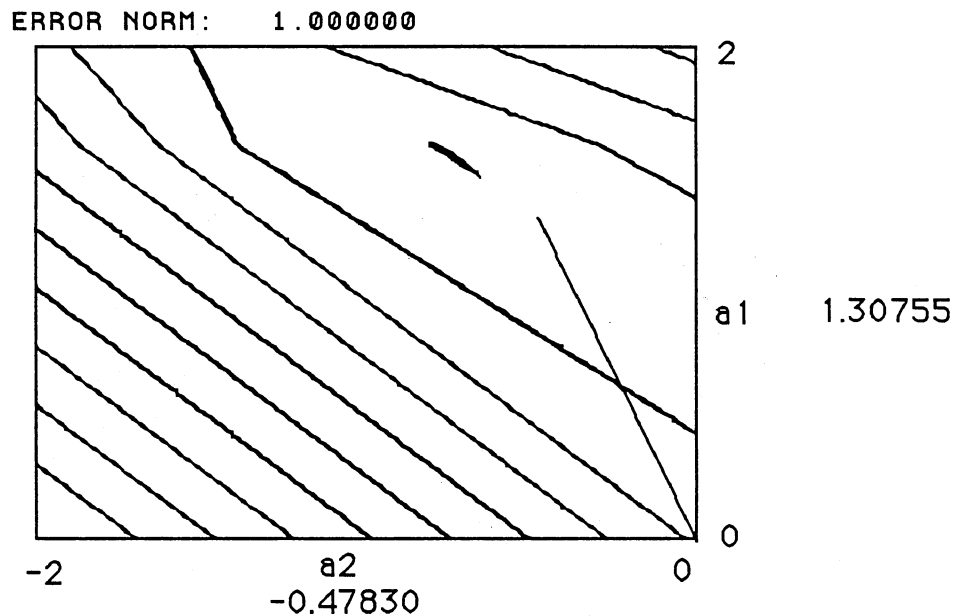
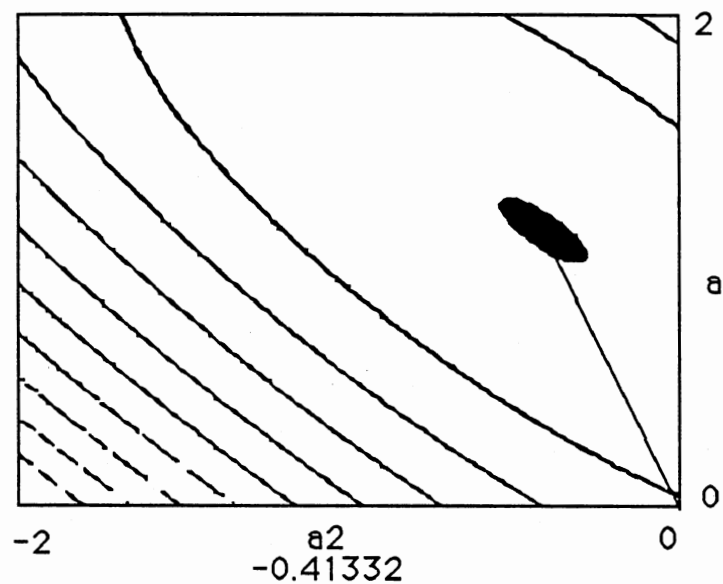


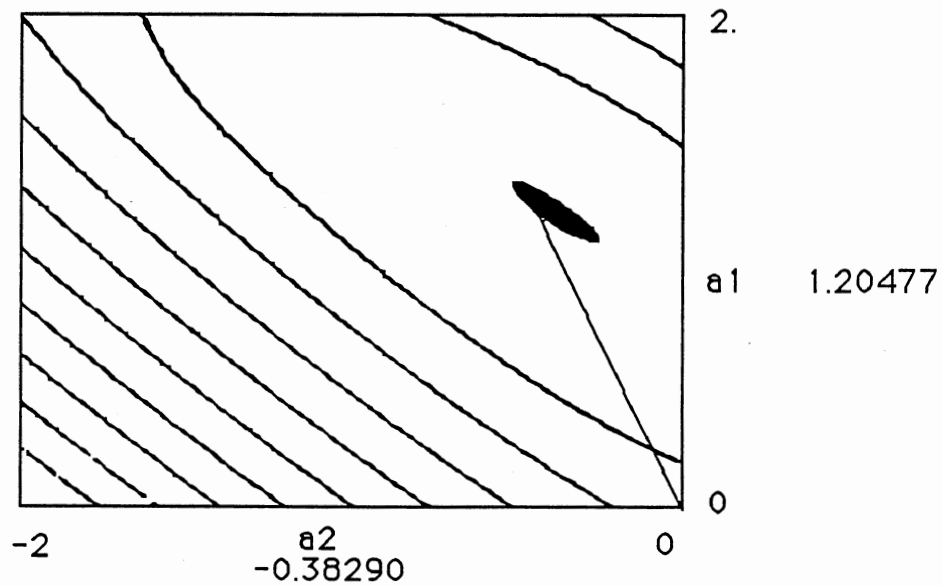
Figure 11b: Solution Locus Using (0,0) at Start

Finally, the RSD algorithm was run using other values of p ; Figures 12 a), b), and c) show the locus of intermediate solutions for $p=2$, 1.5, and 0.5 respectively. For $p=2$ and 1.5, the algorithm was started at $(0, 0)^t$, while $p=0.5$ was started from the L_2 solution. Notice that in all cases, the algorithm stopped before reaching the “true” minimum; this is typical of steepest descent algorithms since convergence is only tested to within some ε (tolerance value). For the L_1 and L_2 cases, the difference between the actual p -normed error and the error when RSD stopped is only a few percent, at most.

ERROR NORM: 2.000000

Figure 12a: Solution Locus for L_2

ERROR NORM: 1.500000

Figure 12b: Solution Locus for $L_{1.5}$

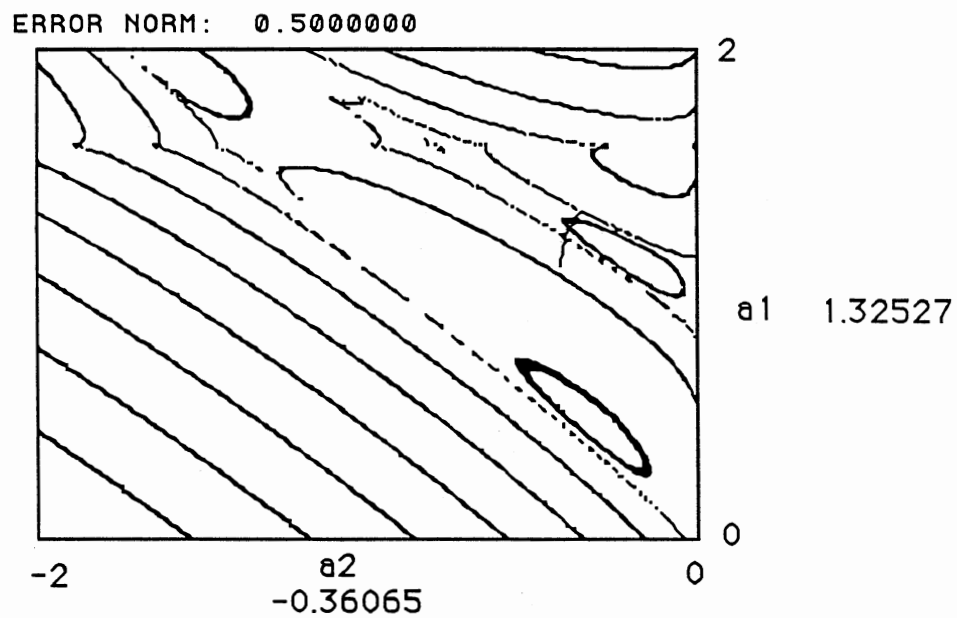


Figure 12c: Solution Locus for $L_{0.5}$

The number of operations required to compute the L_1 solution using RSD can be derived as follows:

Step	Estimated Number of Operations (Multiply/add)	Comments
0	0	(if previous solution used)
1	mq	$\underline{y} - X\underline{a}$
2	0	(sgn time is negligible)
3	$m^2(q+1)$	$(X^tX)^{-1}$
	mq	$(X^tX)^{-1}X^tg(k)$
	mq	$(X^tX)^{-1}X^tg(k)$
	$2(6q)$	IRLS algorithm loops twice
4	m	$a(k+1)$

$$\text{Total: } 3mq + 12q + m^2(q + 1) + m$$

For the example: 116

Thus, for this simple example, the RSD algorithm required only 116 operations while the simplex method required 270, a considerable savings. According to Makhoul (1977), the Cholesky decomposition requires operations on the order of m^3 , although it is not an iterative procedure because the L_2 solution can be expressed in closed form.

CHAPTER IV

GENERALIZED BURG ALGORITHM

As mentioned in Chapter II, lattice techniques have been widely used in speech synthesis; as formulated by Burg, these models are always stable and yield a solution in terms of reflection coefficients, which can be coded quite efficiently in a transmission system (Itakura and Saito, 1972). In addition, the reflection coefficients are calculated sequentially; another term can be added to the model quite easily. Extension of the traditional L_2 procedures to L_p would be quite useful because 1) L_p models are not always stable, and 2) The statistics of the speech residual indicate that the L_p model is more efficient. Alternative norms for the Burg algorithm have been investigated only briefly in the literature – most recently by Denoël and Solvay (1985), who solved the Burg problem in L_1 using what they refer to as a weighted median. As an example, consider calculation of the first reflection coefficient of the sequence

$$x(n) = 1, 2, 3, 4, 5$$

For the first stage, the derivation given in Chapter II, equation (2.14) requires that

$$e_+^0(n) = e_-^0(n) = x(n).$$

The L_2 algorithm would then give, from equation (2.18),

$$k_1 = \frac{-2 \sum_{n=L}^N e_+^0(n) e_-^0(n-1)}{\sum_{n=L}^N [e_-^0(n-1)^2 + e_+^0(n)^2]} \quad (4.1)$$

in this example L equals two, which results in $k_1 = -0.9524$.

Denoël and Solvay (1985) begin with the harmonic mean objective function as defined by Burg:

$$E_1^j = \sum_i \left[\left| e_+^j \right| + \left| e_-^j \right| \right] \quad (4.2)$$

then substitute the error recursion

$$E_1^j = \sum_i \left[\left| e_+^{j-1}(i) + k_j e_-^{j-1}(i-1) \right| + \left| e_-^{j-1}(i-1) + k_j e_+^{j-1}(i) \right| \right] \quad (4.3)$$

By isolating the reflection coefficient term and pulling a term out of the absolute value, the expression can be written as:

$$E_1^j = \sum_i \left[\left| e_-^{j-1}(i-1) \right| \left| \frac{e_+^{j-1}(i)}{e_-^{j-1}(i-1)} + k_j \right| + \left| e_+^{j-1}(i) \right| \left| \frac{e_-^{j-1}(i-1)}{e_+^{j-1}(i)} + k_j \right| \right] \quad (4.4)$$

which is a combination of sums of the general form:

$$E_1^j = \sum_i \left[\left| w^{j-1}(i) \right| \left| q^{j-1}(i) - k_j \right| \right] \quad (4.5)$$

where the $w(i)$ correspond to the first terms in the absolute value signs and the $q(i)$ correspond to the ratios.

A solution to this scalar L_1 problem is what Denoël and Solvay call the weighted median of the sequence $q(i)$. In the example given previously, the

first reflection coefficient can be calculated as follows:

$$e_+^0(n) = e_-^0(n) = x(n) = (1, 2, 3, 4, 5)$$

Using equation (4.5), this implies that

$$w(i) = (1, 2, 3, 4, 2, 3, 4, 5)$$

$$\text{and } q(i) = (2, 3/2, 4/3, 5/4, 1/2, 2/3, 3/4, 4/5)$$

The next step in the algorithm is sorting of the $q(i)$ into ascending order; this yields the sequence

$$q(i)_{\text{sorted}} = (1/2, 2/3, 3/4, 4/5, 5/4, 4/3, 3/2, 2)$$

$$\text{and } w(i)_{\text{reordered}} = (2, 3, 4, 5, 4, 3, 2, 1)$$

where each of the $w(i)$ are merely reordered to match its corresponding $q(i)$ value.

The next step in the procedure is to accumulate a partial sum of the sorted $w(i)$ until this partial sum becomes greater than or equal to one half of the complete sum, which is 24 in this example. Finally, the solution can be found as the $q(i)$ corresponding to the $w(i)$ which satisfied the criterion above. If there are an even number of points, Denoël and Solvay prescribe taking the solution corresponding to the first partial sum equalling or exceeding the one-half sum limit. The table below gives the solution.

$q(i)_{\text{sorted}}$	$w(i)_{\text{reordered}}$	$\sum w(i)$	$\text{sum}/2 = 12$
1/2	2	2	
2/3	3	5	
3/4	4	9	
4/5	5	14	$\Leftarrow \text{solution} = 4/5$
5/4	4	18	
4/3	3	21	
3/2	2	23	
2	1	24	

The weighted median procedure for finding the L_1 solution to the Burg algorithm is not efficient because of the sorting operations required, as Denoël and Solvay admit; they claim that by restricting the problem to finite precision arithmetic, fast sorting algorithms can be used to speed up the process significantly. A more fundamental problem is that the solution can only be found for $p=1$ (by Denoël and Solvay) or $p=2$ (by Burg); in this thesis, the Burg algorithm will be generalized to the L_p case, where p can take on any value in the range $1 \leq p \leq 3$ and convergence to a solution is assured (Byrd and Pyne, 1979).

Use of the approach to solution of the L_2 case for the L_p case yields the following:

$$E_p^j = \sum_{i=L}^N \left[\left| e_+^{j-1}(i) + k_p e_-^{j-1}(i-1) \right|^p + \left| e_-^{j-1}(i-1) + k_p e_+^{j-1}(i) \right|^p \right] \quad (4.6)$$

(forward error) (backward error)

where, like the L_2 case, L equals two.

An attempt to solve these equations in the same manner as the L_2 case leads to the following messy expressions because of the absolute values inside the brackets:

$$\begin{aligned}
\frac{\partial E_p^{j+1}}{\partial k_{j+1}} = 0 = & \sum_{n=L}^N [|a_n + k_{j+1}b_n|^{p-1} |b_n| \text{SGN}(a_n + k_{j+1}b_n)] \quad (\text{forward part}) \\
& + [|b_n + k_{j+1}a_n|^{p-1} |a_n| \text{SGN}(b_n + k_{j+1}a_n)] \quad (\text{backward part})
\end{aligned} \tag{4.7}$$

where the a_n and the b_n are $e_+^j(n)$ and $e_-^j(n-1)$, respectively.

No clear solution technique has emerged for this form; the original E_p^{j+1} can be cast as an IRLS problem, however, making possible an iterative solution. The IRLS technique (Yarlagadda, *et al*, 1985) allows efficient calculation of a p-normed scalar predictor for a vector equation $\underline{X}\underline{a}=\underline{y}$. If the backward error term is rewritten after a change of variables, the p-normed error at the $j+1^{\text{th}}$ stage becomes:

$$E_p^{j+1} = \sum_{n=L}^N \left(\left| e_+^j(n) + k_{j+1}e_-^j(n-1) \right|^p + \left| e_-^j(n-1) + k_{j+1}e_+^j(n) \right|^p \right) \tag{4.8}$$

If we write out the terms, a structure becomes evident; consider the following expansion:

$$\begin{aligned}
E_p^{j+1} = & \left| e_+^j(L) + k_{j+1}e_-^j(L-1) \right|^p + \left| e_-^j(L-1) + k_{j+1}e_+^j(L) \right|^p \\
& + \left| e_+^j(L+1) + k_{j+1}e_-^j(L) \right|^p + \left| e_-^j(L) + k_{j+1}e_+^j(L+1) \right|^p \\
& \vdots \\
& + \left| e_+^j(N) + k_{j+1}e_-^j(N-1) \right|^p + \left| e_-^j(N-1) + k_{j+1}e_+^j(N) \right|^p
\end{aligned} \tag{4.9}$$

Minimization of this quantity is equivalent to finding the L_p solution to the system

$$\begin{bmatrix} e_{-}^j(L-1) \\ \vdots \\ e_{-}^j(N-1) \\ e_{+}^j(L) \\ \vdots \\ e_{+}^j(N) \end{bmatrix} k_{j+1} = - \begin{bmatrix} e_{+}^j(L) \\ \vdots \\ e_{+}^j(N) \\ e_{-}^j(L-1) \\ \vdots \\ e_{-}^j(N-1) \end{bmatrix} \quad (4.10)$$

or

$$\begin{bmatrix} \underline{b} \\ \underline{f} \end{bmatrix} k_{j+1} = - \begin{bmatrix} \underline{f} \\ \underline{b} \end{bmatrix} \quad (4.11)$$

in the p -normed sense, where \underline{b} is the backward error vector and \underline{f} is the forward error vector. Naturally, this overdetermined system can be solved in any number of ways, including minimum p -norm. As a check, the L_2 solution would be $k_1 = -(\underline{b}^t \underline{b} + \underline{f}^t \underline{f})^{-1} (\underline{b}^t \underline{f} + \underline{f}^t \underline{b})$ which, since the quantities inside the parenthesis are scalars, is clearly the same form as equation (4.1).

Consider the following example; if the sequence given previously (1, 2, 3, 4, 5) is cast as a covariance problem, the solution to the first order covariance L_2 problem is $a=4/3$ which yields an unstable filter. The L_2 Burg algorithm yields a reflection coefficient of $k_1=-0.9524$, as given earlier, which is a stable filter. The L_p Burg algorithm for the first stage would be

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} k_1 = - \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad (4.12)$$

which for $p=1$ yields $k_1=-0.8$. The recursion can be used to calculate the next error vector from the previous error and the reflection coefficient.

$$\underline{b} = \begin{bmatrix} 1.0 \\ -0.6 \\ -0.4 \\ -0.2 \\ 0.0 \end{bmatrix} \quad \underline{f} = \begin{bmatrix} 1.0 \\ 1.2 \\ 1.4 \\ 1.6 \\ 1.8 \end{bmatrix} \quad (4.13)$$

It has been found during this investigation that the L_1 solution may be considerably different from the L_2 solution and that this affects the spectral properties of the model, as will be shown in Chapter VI.

Stability of the L_p solution can be described in terms of the error vectors \underline{f} and \underline{b} ; if we consider the partition in equation (4.11), the p -normed solution at each step of the IRLS algorithm will be of the form:

$$k_{j+1} = - \left[\underline{b}^t \mathbf{W}_1 \underline{b} + \underline{f}^t \mathbf{W}_2 \underline{f} \right]^{-1} \left[\underline{b}^t \mathbf{W}_1 \quad \underline{f}^t \mathbf{W}_2 \right] \begin{bmatrix} \underline{f} \\ \underline{b} \end{bmatrix} \quad (4.14)$$

where \mathbf{W}_1 and \mathbf{W}_2 are diagonal weighting matrices as defined in equation (3.11). Since k_{j+1} is a scalar, this expression can be rewritten as:

$$k_{j+1} = \frac{- \left[\underline{b}^t \mathbf{W}_1 \underline{f} + \underline{f}^t \mathbf{W}_2 \underline{b} \right]}{\left[\underline{b}^t \mathbf{W}_1 \underline{b} + \underline{f}^t \mathbf{W}_2 \underline{f} \right]} \quad (4.15)$$

If $\mathbf{W}_1 = \mathbf{W}_2 = \mathbf{I}$, the identity matrix, this expression is the same form as equation (2.18); stability is assured by the same argument that holds for the L_2 case. For the more general case, consider equation (4.11); for the first iteration of the IRLS algorithm, it follows from the L_2 stability argument that the next value of k_{j+1} must be less than one in absolute value, if the initial starting point is +1 or -1. Thus, the absolute value of k_{j+1} must decrease if it is chosen initially to be the marginally stable solution – k_{j+1} must then be bounded by $[-1, +1]$.

The generalized Burg algorithm is a powerful extension to the L_2 method; it may even be possible to use other weighting functions to optimize with respect to other objective functions.

During development and testing of the L_p Burg algorithm, an important computational issue arose. As is well known, the L_1 solution always exists but may not be unique (Yarlagadda, *et al*, 1985). In fact, for an even number of points, there will usually be an infinity of solutions (Denoël and Solvay, 1985), which will always be the case for the Burg algorithm. Since this family of solutions may span a considerable range in the solution space, care must be taken to assure that a given L_p model will yield a synthesis filter which will give acceptable speech quality. An initial solution of $k_j^0 = 0$ was used in preliminary experiments to save computation; this resulted in solutions which were too small to adequately capture the formant structure. Similarly, setting the initial $k_j^0 = 1$, which should pick solutions closer to the unit circle in the Z -plane, made the synthesis filter overly resonant and, again, sound quality suffered. It was found, as suggested by Yarlagadda, *et al* (1985), that the L_2 solution is the “best” starting point for the IRLS algorithm. The L_p Burg

algorithm converges rapidly for a wide range of values of p and model sizes. It offers a powerful extension to existing lattice techniques.

CHAPTER V

OPTIMAL ADAPTIVE L_p MODELS FOR SPEECH

As discussed in Chapter II, a p -normed model may offer greater statistical efficiency for speech since the L_2 (least squares) model is the maximum likelihood estimator only if the residual is Gaussian, which is generally not the case. The family of p -Gaussian distributions encompasses a wide variety of common distributions, including Laplacian ($p=1$), Gaussian ($p=2$), and uniform ($p \rightarrow \infty$). A p -normed model is the maximum likelihood estimator for the corresponding p -Gaussian distribution. The L_p model can be calculated efficiently using the residual steepest descent algorithm. Since the distribution of the residuals determines the maximum likelihood estimator for a given signal, the most obvious starting point for an analysis of speech is an investigation of the residual sequence.

The process of deconvolution is an attempt to remove information that is linearly predictable from a weighted sum of previous samples; the residuals from this process should then be nearly uncorrelated. The inverse filter used

$$H(z) = 1 - \sum_{i=1}^m a_i z^{-i} \quad (5.1)$$

to generate the residuals is often called a “whitening filter” for this reason, since white noise is by definition uncorrelated, resulting in a nearly flat spectrum (Rabiner and Schafer, 1978, p. 422). The fact that deconvolution may give uncorrelated residuals, however, tells nothing about the distribution of them – the pdf cannot be inferred from the correlation coefficient.

Experimental results will be used to indicate possibilities for a pdf and to provide a rationale for a p-Gaussian basis for a speech coding model.

Speech is traditionally divided into two types, voiced and unvoiced, with the understanding that some sounds in speech have both voiced and unvoiced properties. Consider the segment of speech given in Figure 13; this is the sound /a/ from the word “add” as spoken by a female speaker. If this segment is analyzed by a least squares algorithm to yield a sixth order model and the residual sequence is generated, the sequence given in Figure 14 results. Note the prominent glottal pulses and what is apparently noise between the pulses; if we calculate the histogram of this sequence, the plot in Figure 15 results. The assumption that was made in Chapter II was that speech, especially voiced speech, is more Laplacian than Gaussian. For an L_1 model to be the maximum likelihood estimator for it, the residual must also be Laplacian, which is a long-tailed distribution containing a significant number of outliers. Techniques to test the distribution will be discussed shortly.

An issue mentioned by Denoël and Solvay (1985) is the use of the L_1 residual for pitch estimation of voiced speech. Since the L_1 estimator is “robust,” i.e., it rejects impulsive behavior, the glottal pulses are more strongly evident in the residuals.

To see how voiced speech compares with unvoiced speech, consider the unvoiced segment of speech given in Figure 16, which is the /ts/ sound from the word “cats”. In the same fashion as before, the residual from a sixth order model was generated and is shown in Figure 17, which shows that the noiselike characteristics of unvoiced speech yield a noiselike residual. The

histogram of this segment is given in Figure 18, which shows a fairly compact distribution with a few outliers, some of which may be significant.

Finally, an example of a voiced fricative segment of speech is given in Figure 19, which shows the /z/ sound from the word “thieves” as spoken by a male speaker. In this case, the residual given by Figure 20 shows the periodicity one would expect of a voiced sound but without the glottal excitation as in the case of voiced speech. The distribution, which is given in Figure 21, exhibits properties of both voiced and unvoiced speech, as would be expected.

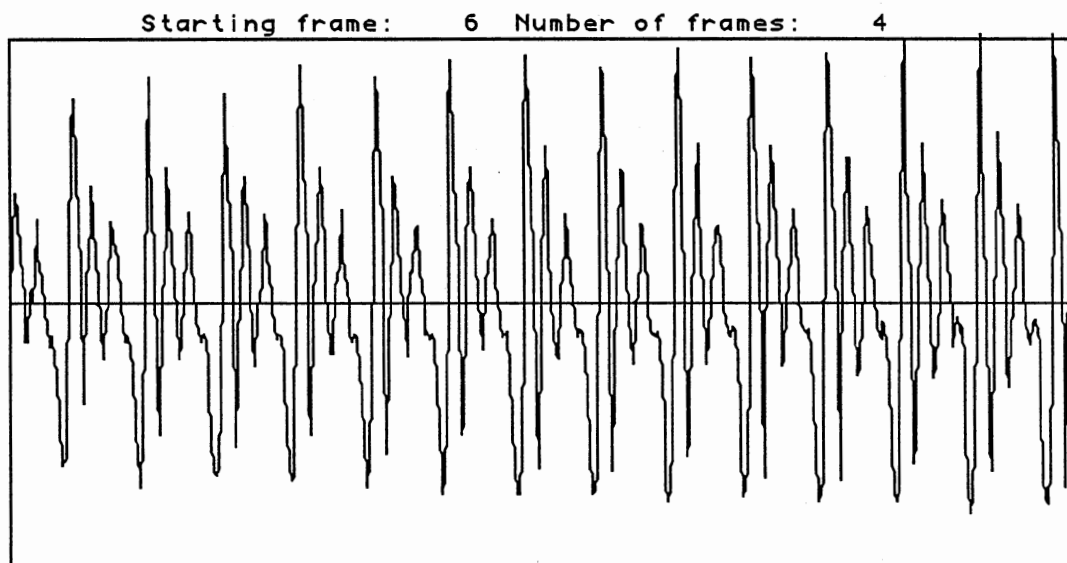


Figure 13: Speech Waveform: /a/ from “add”

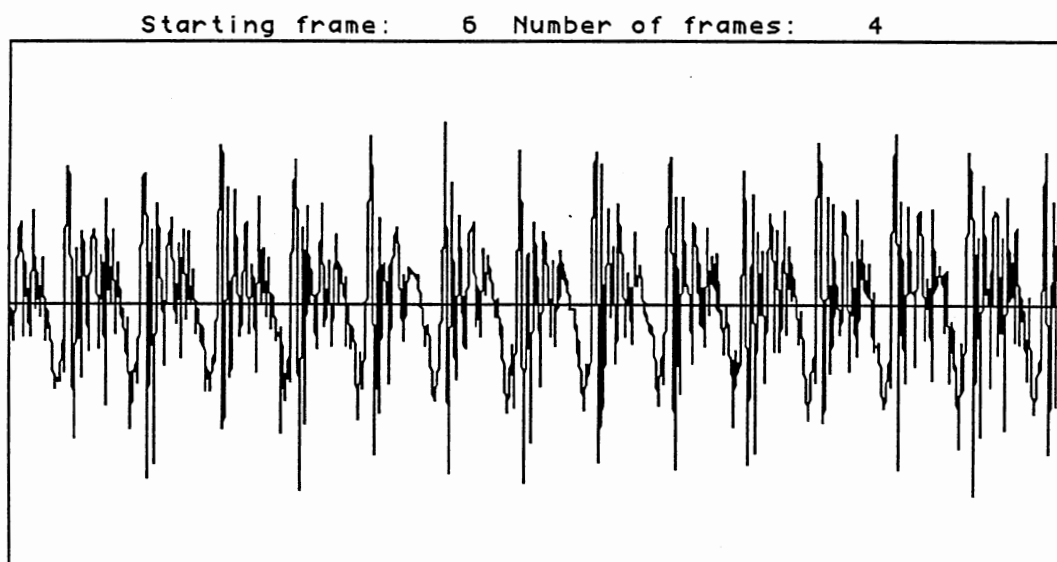


Figure 14: Residual for /a/

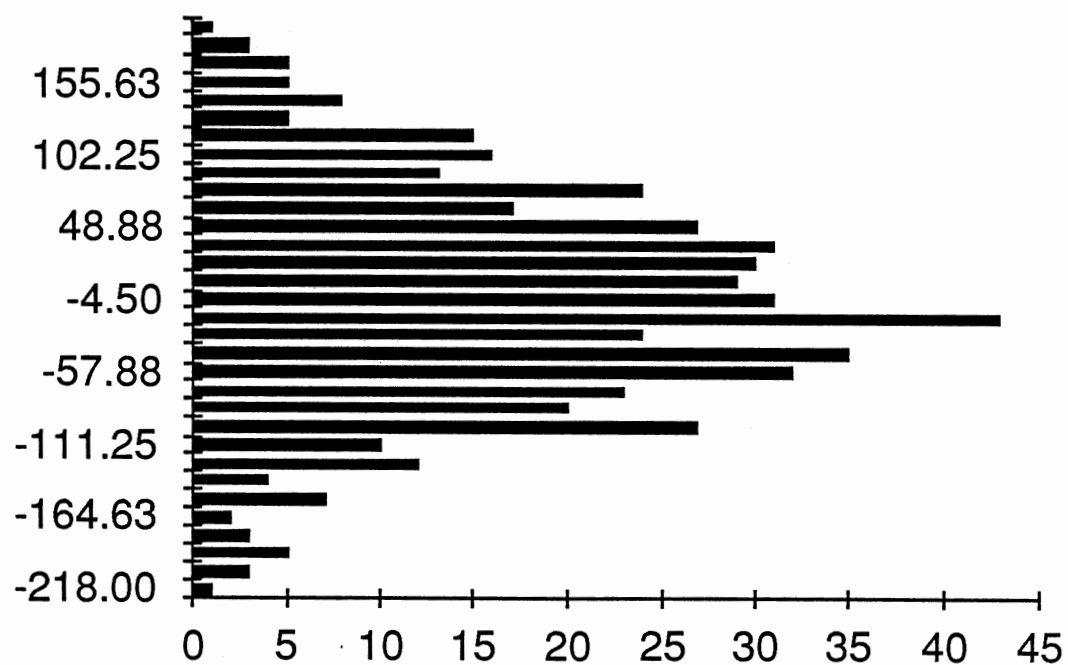


Figure 15: Histogram of Residual for /a/

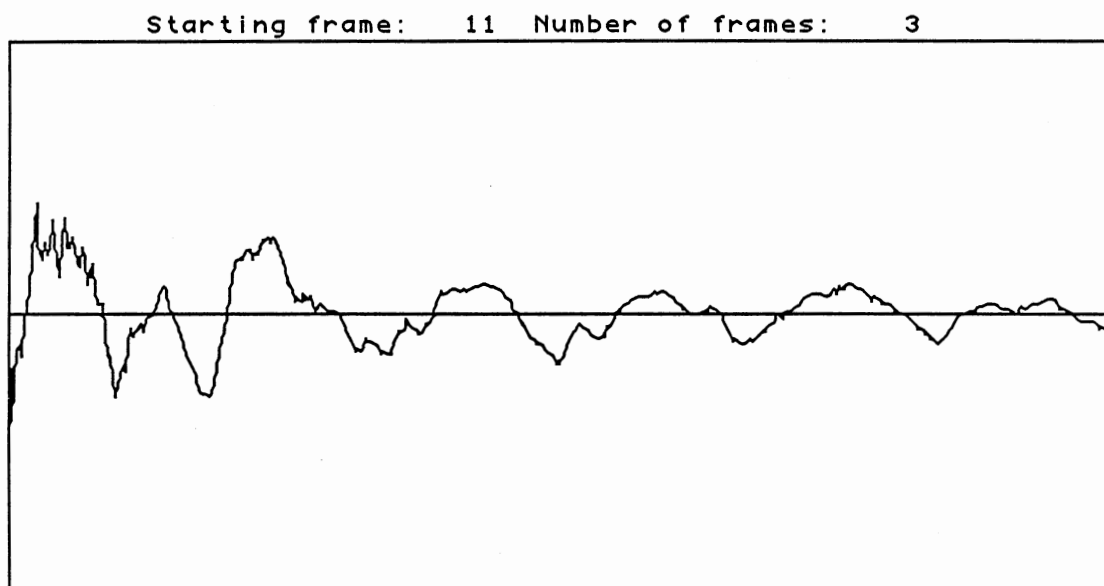


Figure 19: Speech Waveform: /ts/ from "cats"

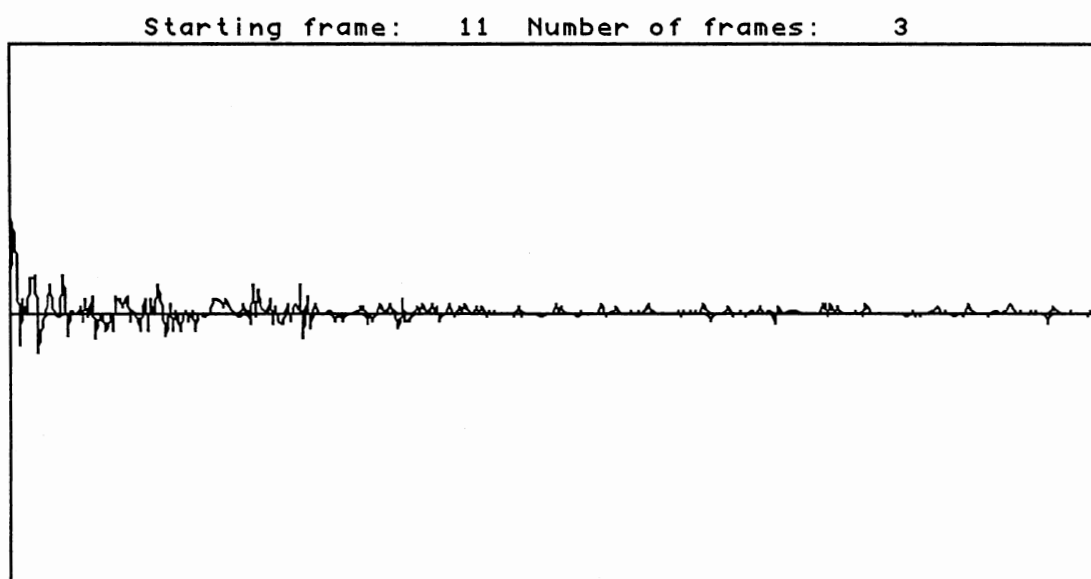


Figure 17: Residual for /ts/

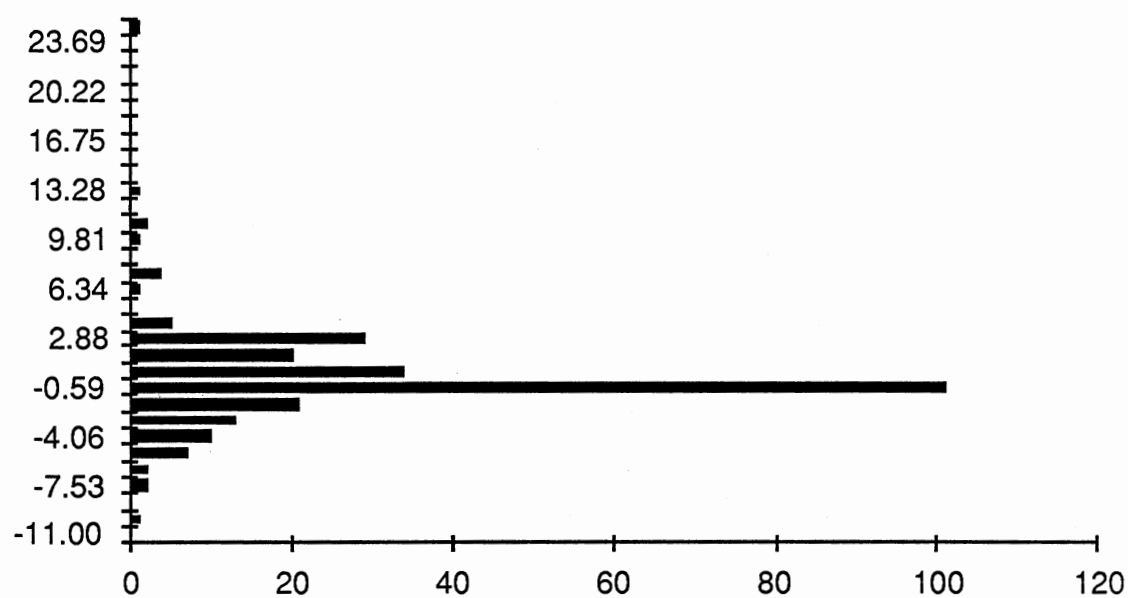


Figure 18: Histogram of Residual for /ts/

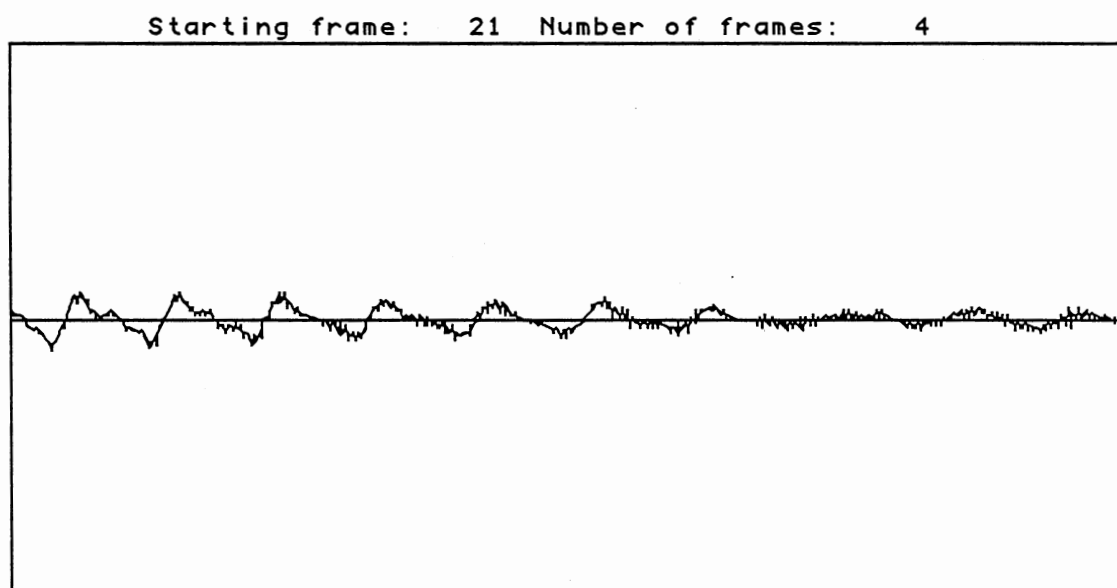


Figure 19: Speech Waveform: /z/ from "thieves"

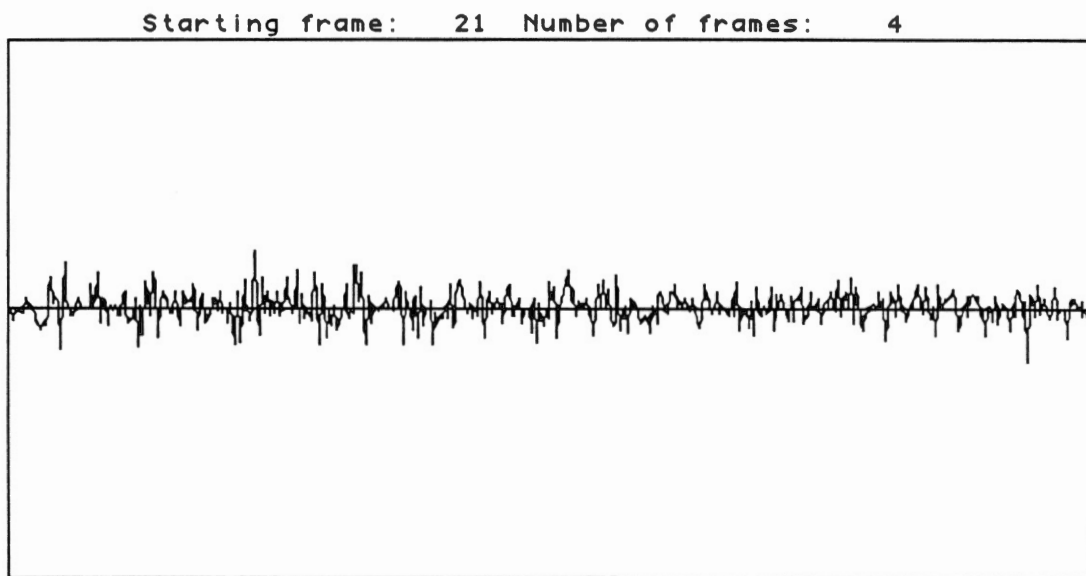


Figure 20: Residual for /z/

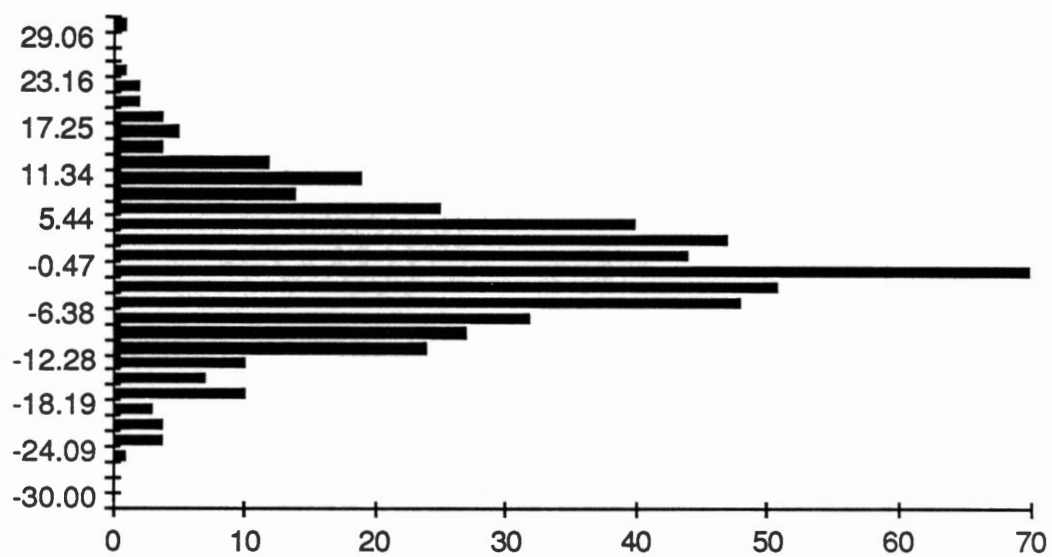


Figure 21: Histogram of Residual for /z/

These experimental results, while not quantitative in nature, do indicate that an assumption of Gaussian residuals for speech is not valid; speech residuals appear to deviate significantly from Gaussian. This will be further verified shortly.

The p-Gaussian family of distributions encompasses a wide variety of useful distributions, as has been mentioned. An optimum value of p is desired that is derived from some measure of how non-Gaussian the residual sequence is. While a traditional statistical procedure such as the chi-squared test could be used, it is not particularly powerful as a test for normality, nor does it give any indication of which p-Gaussian distribution would be more suitable if the normality test fails. Other tests such as Kolmogorov-Smirnoff could also be used but offer no obvious mapping to an optimal p .

What is needed is a distribution independent measure of the dispersion of the residuals. Such a measure is the kurtosis, which is defined as (Olkin, *et al*, 1980, p. 152):

$$k_4 = \frac{\frac{1}{N} \sum (x - \mu_x)^4}{\left(\frac{1}{N} \sum (x - \mu_x)^2 \right)^2} = \frac{M_4}{(M_2)^2} \quad (5.2)$$

which is, in words, the fourth moment divided by the second moment squared.

Kurtosis has been used to measure the deviation from Gaussian in applications ranging from general detection theory (Miller and Thomas, 1972) to noise generated by ice in sonar signals (Dwyer, 1982, pp. 79-90), to general undersea noise (Machell and Penrod, 1982; Wilson and Powell, 1982). Kurtosis is a measure of the "peakedness" of a distribution and is

particularly useful when dealing with a symmetric, unimodal distribution (Croxtan, 1953, p. 101). The Gaussian distribution, $N(0,1)$, is often used as a reference and can easily be shown to have a kurtosis of three, since its variance is one and its fourth moment is three. Many texts in statistics refer to a related term, the coefficient of excess, which is simply the kurtosis minus three (Patel, *et al*, 1976, p. 5). The Gaussian distribution is called mesokurtic; distributions which have a narrower modal portion and higher tails are called leptokurtic and are characterized by a kurtosis greater than three, while broader, lower-tailed distributions are called platykurtic and have a kurtosis less than three (Croxtan, 1953, p. 101). Figure 22 gives an example of leptokurtic, mesokurtic and platykurtic distributions. The kurtosis of most common distributions have been calculated (see Patel, *et al*, 1976); some examples of the kurtosis values for distributions of interest are

Uniform: $k_4 = 1.8$ (Patel, *et al*, 1976, p. 36)

Laplace $k_4 = 6$ (Patel, *et al*, 1976, p. 34)

This agrees with the qualitative assessment made previously that the Laplace distribution has higher (or longer) tails than does the Gaussian distribution. It should be noted that statistical testing of a Gaussian vs. non-Gaussian hypothesis is possible and that confidence intervals for such a hypothesis test have been tabulated for various values of N , the number of samples (Croxtan, 1953, p. 342).

Verification of the assumption that kurtosis is a suitable measure of the distribution of speech must come experimentally, since speech is non-stationary. The three segments of speech described earlier were analyzed

using a conventional least squares algorithm, and the residual was generated. Following this, the kurtosis of the 128 samples prior to and including a given point was calculated. This sequence was written to disk and plotted. Figures 23, 24, and 25 show the kurtosis of the residual for the speech segments given in Figures 13, 16, and 19, respectively. As these figures show, the kurtosis varies widely and is generally nowhere near three, failing the 99% confidence interval test ($2.24 <-> 4.24$) for 125 samples from a Gaussian distribution in many places. The nominal Gaussian value of three is given by the horizontal reference line in the figures. Thus, speech residuals are not generally Gaussian and a p-normed model would offer the flexibility of an adaptive model that is closer to the maximum likelihood estimator of speech, perhaps improving the formant structure of synthesized speech.

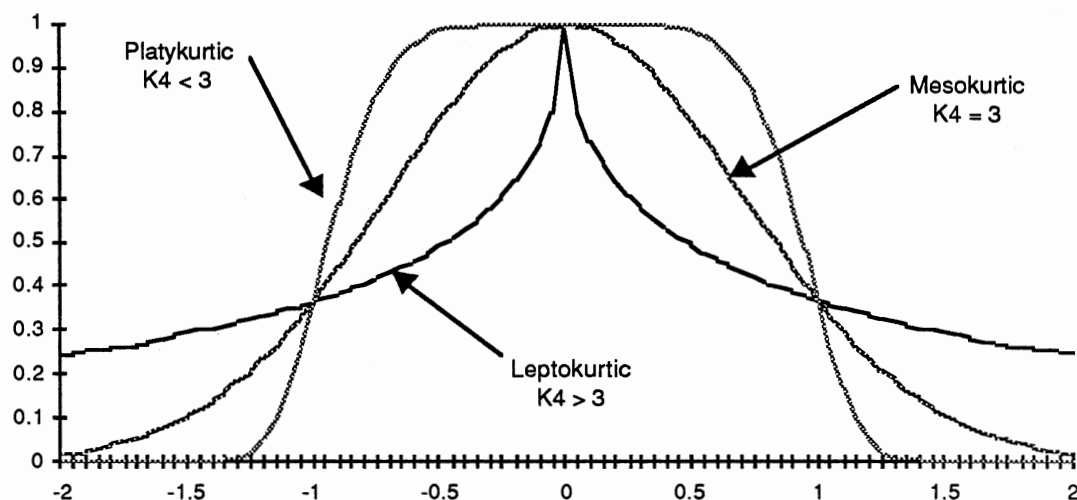


Figure 22: Kurtosis as Related to PDF Shape

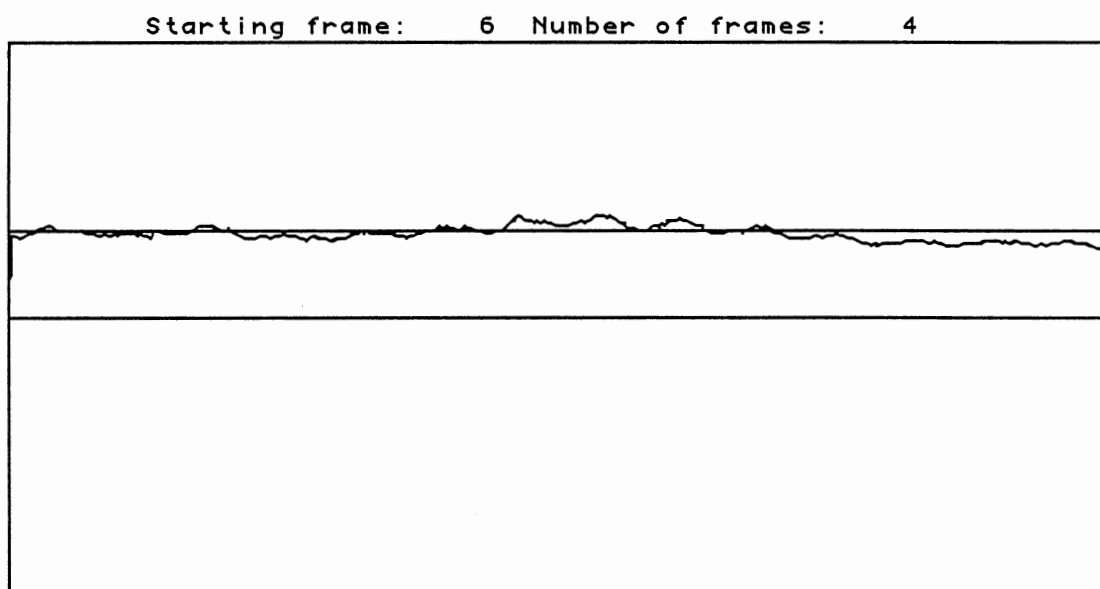


Figure 23: Kurtosis of Residual for /a/

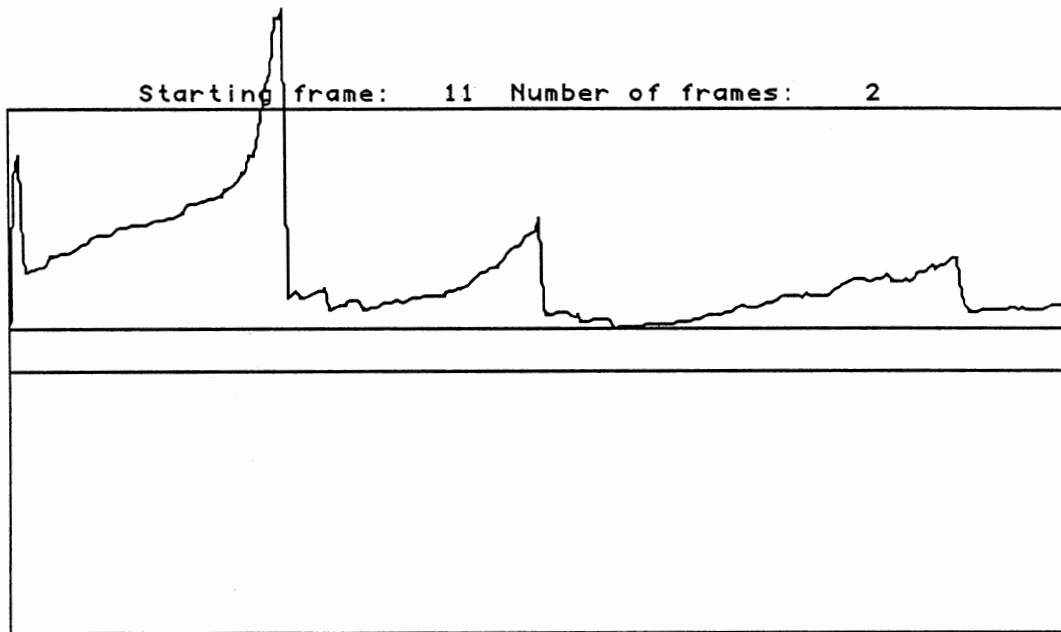


Figure 24: Kurtosis of Residual for /ts/

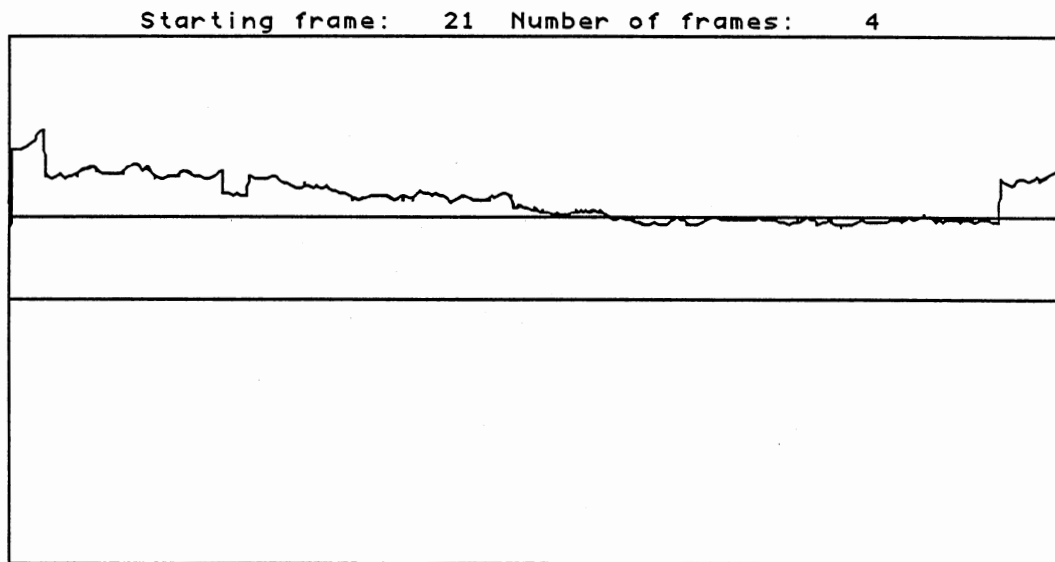


Figure 25: Kurtosis of Residual for /z/

To test the intelligibility of L_p models in speech coding, the government standard LPC-10 algorithm has been used. LPC-10 is a linear predictive coding analysis/synthesis system that uses a tenth order model, computed using the covariance method, to achieve a 2400 bits/second output rate. Figure 26 shows a block diagram of the transmitter portion of LPC-10 (after Tremain, 1982), which is the portion of the system of most interest here. Conversion of LPC-10 to use an L_p solution merely involves modification of two blocks in the system: matrix load (loads the \mathbf{X} matrix and \mathbf{y} vector) and matrix solution (uses the Cholesky decomposition to solve for the \mathbf{a} vector, then converts these to reflection coefficients for coding). A special subroutine, called SOLVOPTP, replaces the load and invert modules to compute the L_p model and convert the predictor coefficients to reflections coefficients using

the "stepup" procedure (Markel and Gray, 1976, p. 151). The block diagram of the modified system is given in Figure 27, where the highlighted blocks are the additions. Instability of the model, if it occurs, is treated by use of the previous frame's reflection coefficients. No changes were made in the receiver portion of the algorithm.

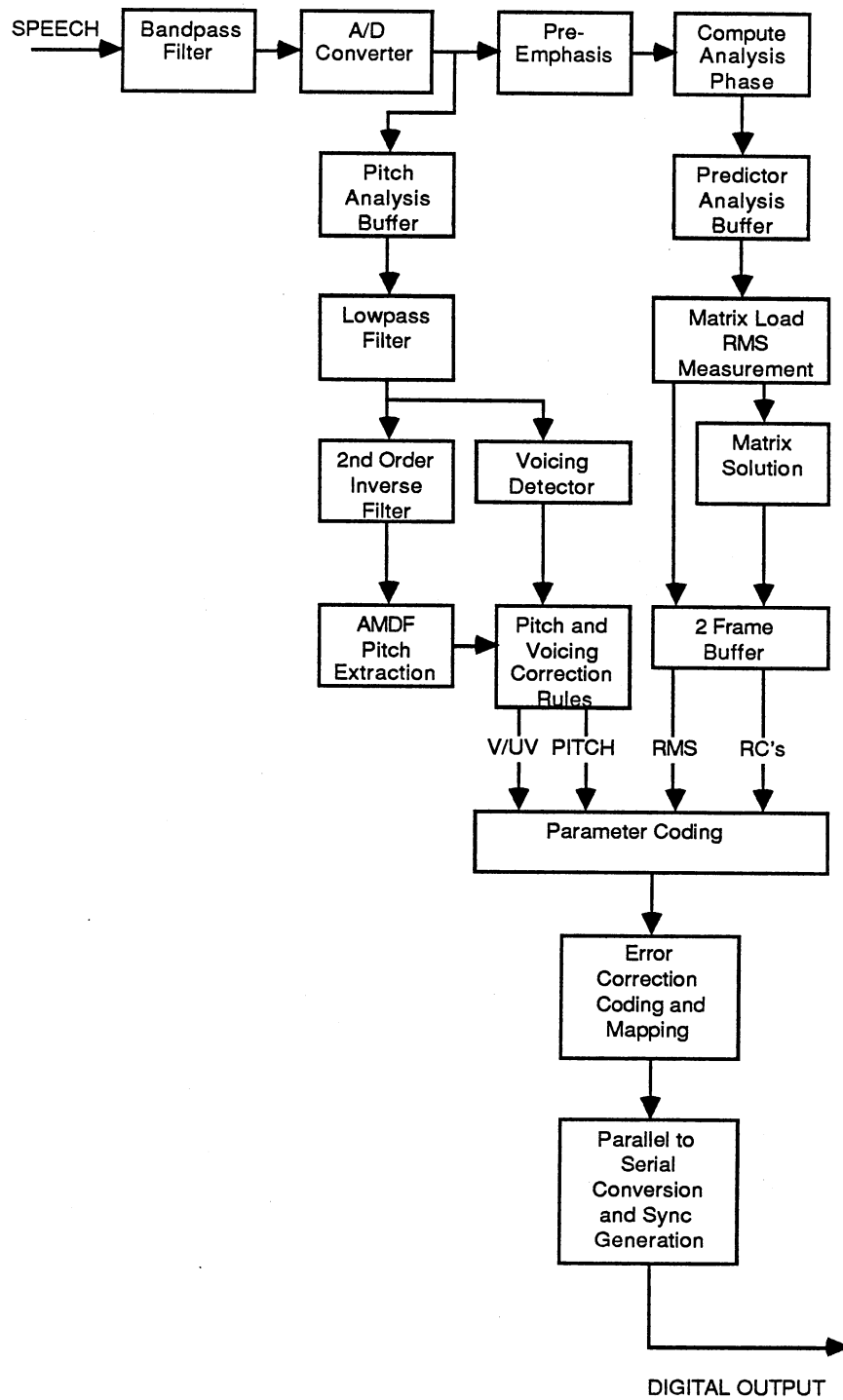


Figure 26: Block Diagram of LPC-10 Transmitter

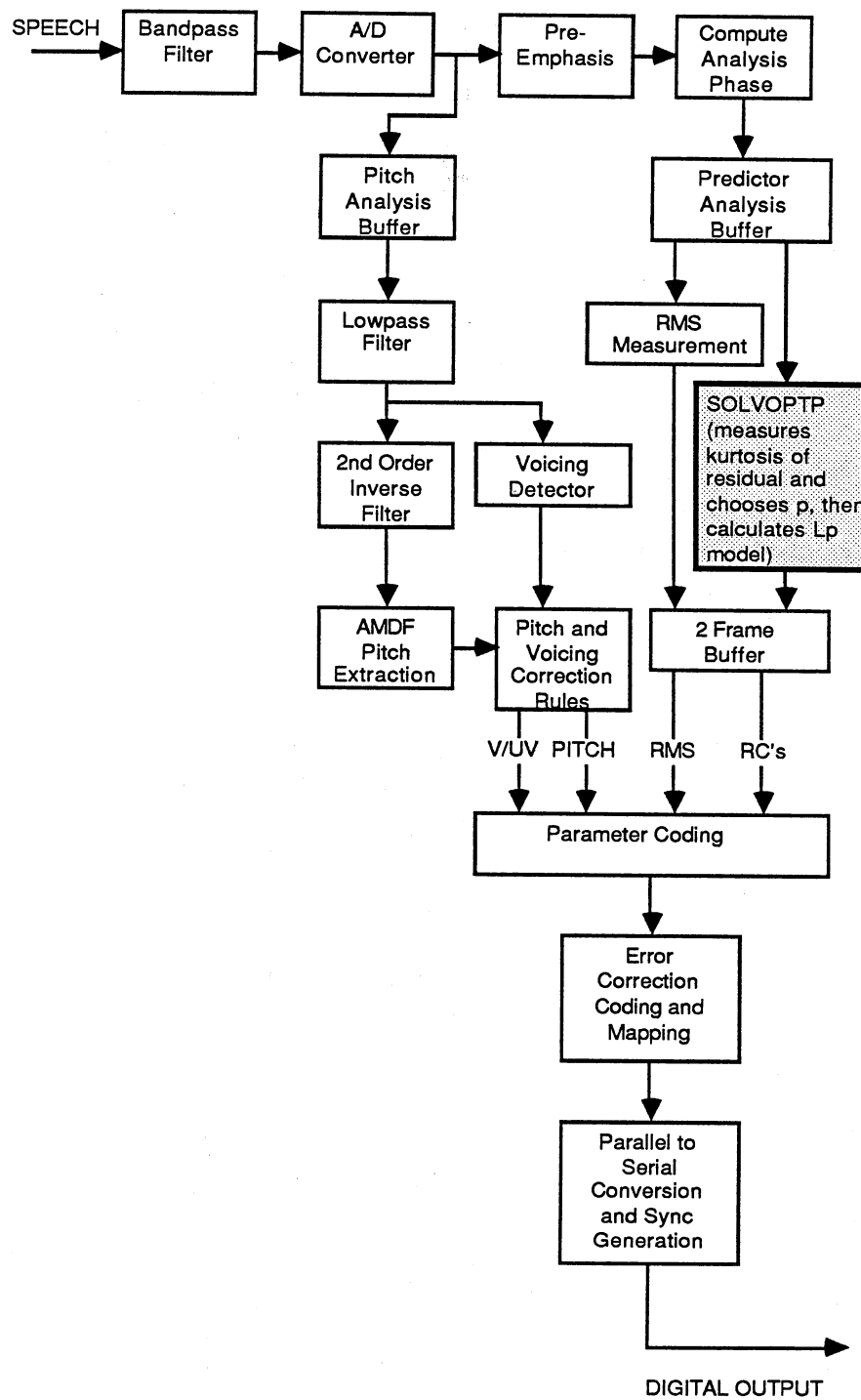


Figure 27: Block Diagram of Optimal p LPC-10 Transmitter

The value of p used is derived from a linear mapping of the kurtosis of the residual, where the mapping is performed as follows:

Kurtosis p value used

$$0 \leq k_4 \leq 6 \quad p = 3 - [k_4/3]$$

$$k_4 > 6 \quad p = 1$$

K_4 cannot be less than 0

Using SOLVOPTP, the LPC-10 program was run for a variety of speech segments. While the reader obviously cannot hear the resulting synthesized speech, it is instructive to plot the optimal values of p used for each frame of speech in LPC-10. Figure 28 shows the kurtosis, while Figure 29 shows the p -value used as a function of frame number for the sentence, "Thieves who rob friends deserve jail." Note that the kurtosis is always greater than three (3.187, actually) for the entire sentence, indicating that, in this case, the residual distribution is always more long-tailed than Gaussian.

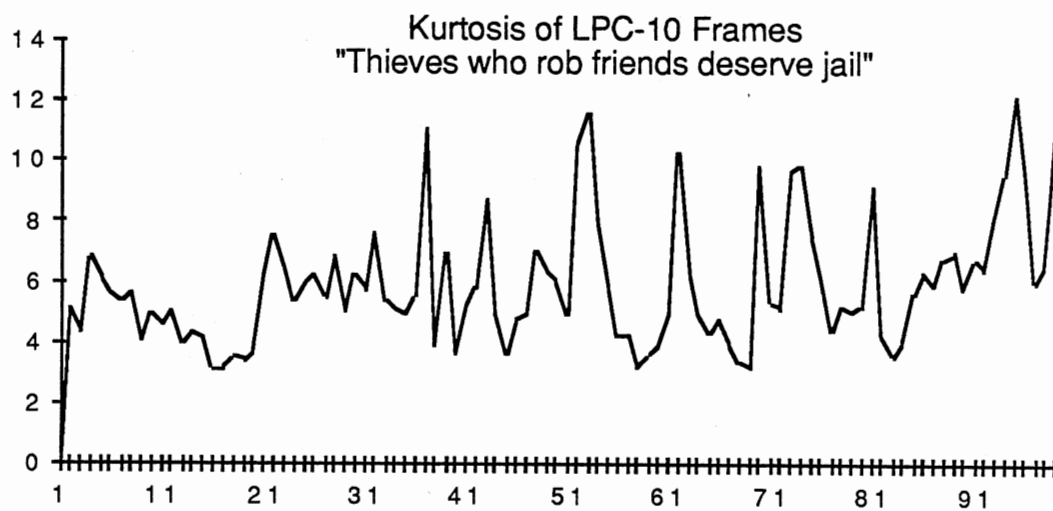


Figure 28: Kurtosis of Frames: "Thieves who rob friends deserve jail"

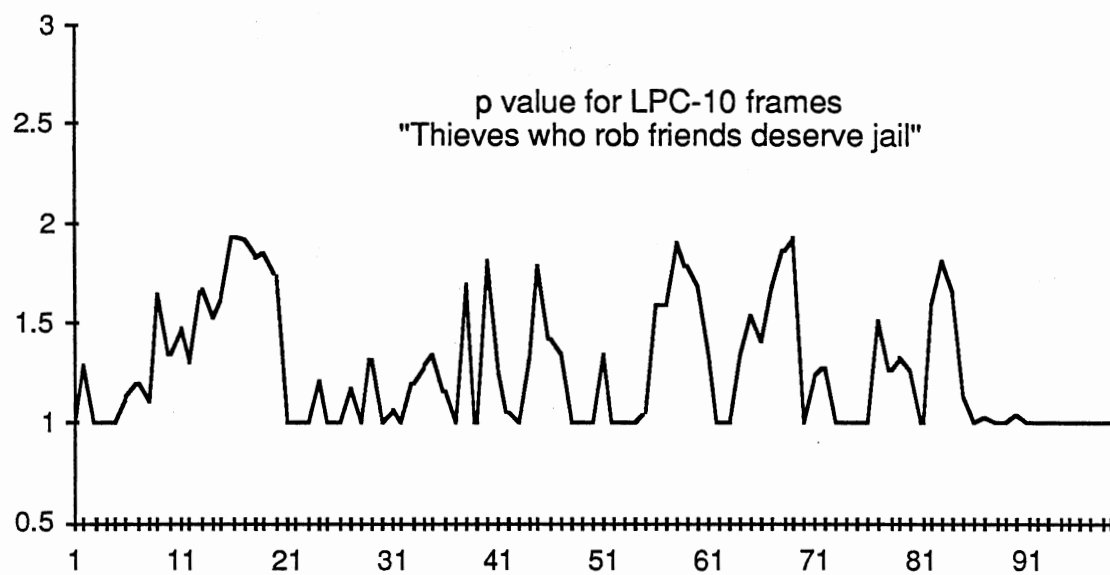


Figure 29: Optimal p-value used for "thieves" sentence

CHAPTER VI

SPECTRAL ESTIMATION OF SPEECH USING L_p MODELS

As mentioned in Chapter I, the linear predictive coefficients can be viewed as a filter with a transfer function of:

$$P(z) = \frac{G}{1 - \sum_{i=1}^m a_i z^{-i}} \quad (6.1)$$

If the magnitude of $P(z)$ is plotted as a function of frequency by setting $z=e^{j\omega}$ (where a sampling rate of one Hertz can be assumed without loss of generality), then an estimate of the “true” spectrum of $x(n)$ results (Makhoul, 1975). Spectral estimation of speech is important for formant and pitch estimation and gives great insight into the performance of a speech model. Much work has been done in spectral estimation of speech (Markel and Gray, 1976; Itakura and Saito, 1970; Atal and Hanauer, 1971) and even in L_p spectral models (Schroeder, 1985; Schroeder and Yarlagadda). It is the intent here to analyze and compare both the covariance derived L_p spectra and the Burg derived L_p spectra in “clean” and noisy environments.

The first series of spectral plots was generated using the covariance formulation of the L_p linear prediction problem; the figures given are 3-D plots of $P(e^{j\omega})$ as a function of frequency and time of 128 sample frames from the word “cats”, as shown in Figure 30.

Figure	p value	
31	-0.1	
32	0.5	
33	1.0	(Least absolute value solution)
34	1.5	
35	2.0	(Least squares solution)
36	3.0	

Next, using the L_p Burg algorithm, the following figures were generated from the same word.

Figure	p value	
37	-0.1	
38	0.5	
39	1.0	
40	1.5	
41	2.0	('normal' Burg solution)
42	3.0	

Using the weighted median algorithm from Denoël and Solvay (1985), the same spectrum versus time plot was generated in Figure 43; it is interesting to note that the L_p algorithm, when started at the L_2 solution, appears to preserve the formants more effectively.

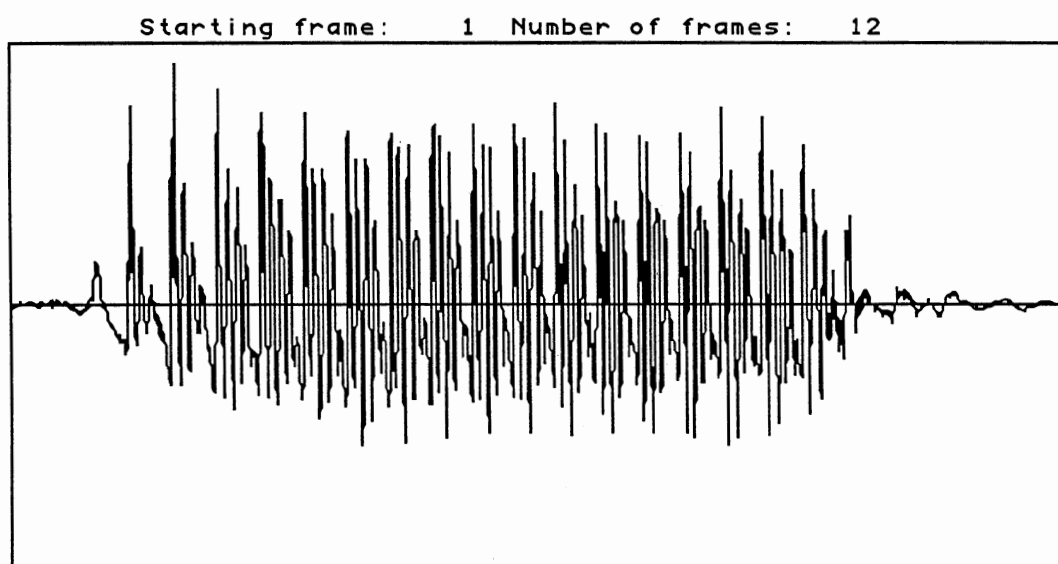
Finally, performance in noise was evaluated. The Burg and covariance techniques were both used with noisy speech to test the noise immunity of the estimator. A uniformly distributed noise source was added to the speech data, which were 12-bit quantized $([-2048, 2047])$; the approximate

signal-to-noise ratio was 20 dB, which was calculated by taking ten times the logarithm of the ratio of maximum speech power (assuming a sinusoid of maximum amplitude) to the noise power. In addition, an impulse train with a period of 16 ms and amplitude of 64 was added to the signal. Using this corrupted speech, the following figures were generated.

Figure	p value	Comments
44	2.0	Covariance
45	2.0	Burg
46	1.0	Covariance
47	1.0	Burg

A rather surprising result is that the L_1 Burg algorithm appears to perform poorly in noise; it is speculated that the latitude afforded each of the reflection coefficients at each stage of the algorithm allows the poles of the model $P(z)$ to slip too near the unit circle. Further research is needed to observe the pole locations as the noise level is varied. It may be necessary to further constrain the solution as the L_p Burg algorithm iterates to prevent the poles from “slipping”. As Markel (1987) noted, the fact that the Burg algorithm is only constrained locally (at each stage) rather than being a global optimization, some artifacts or problems may creep in. Other than this, the L_p models form the covariance technique performed well in the presence of noise. In fact, for the uniform noise used here, one would expect the L_2 estimator to perform at least as well as the L_1 model but the formants seem to be slightly more pronounced in the figures, indicating that the robustness of

the L_1 estimator is quite powerful in rejecting noise. As a final basis of comparison, the spectra from two different methods were plotted on the same figure. As shown in Figure 48, the L_2 covariance model is shown on the same axis as the L_1 covariance model. Figure 49 shows the L_2 Burg and L_1 Burg model spectra. As these figures show, the difference between these models is not dramatic, but in the case of speech coding in a realistic noisy environment, the differences may become more apparent.



Word: "Cats" (no noise)

Figure 30: Speech Waveform : "Cats"

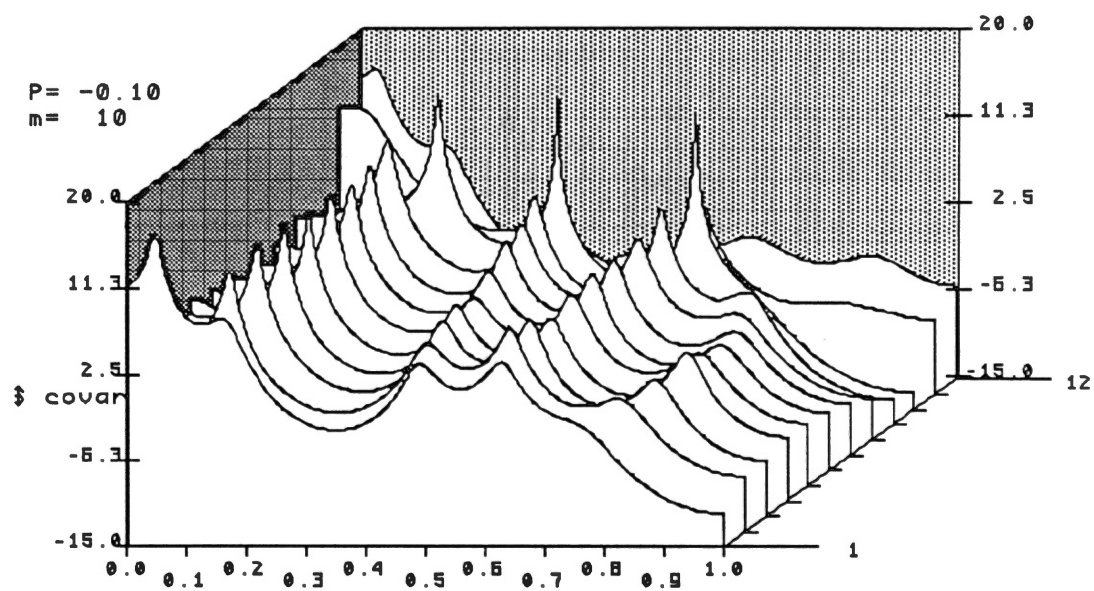


Figure 31: Spectral Surface, Covariance Model, "Cats", $\rho = -0.1$

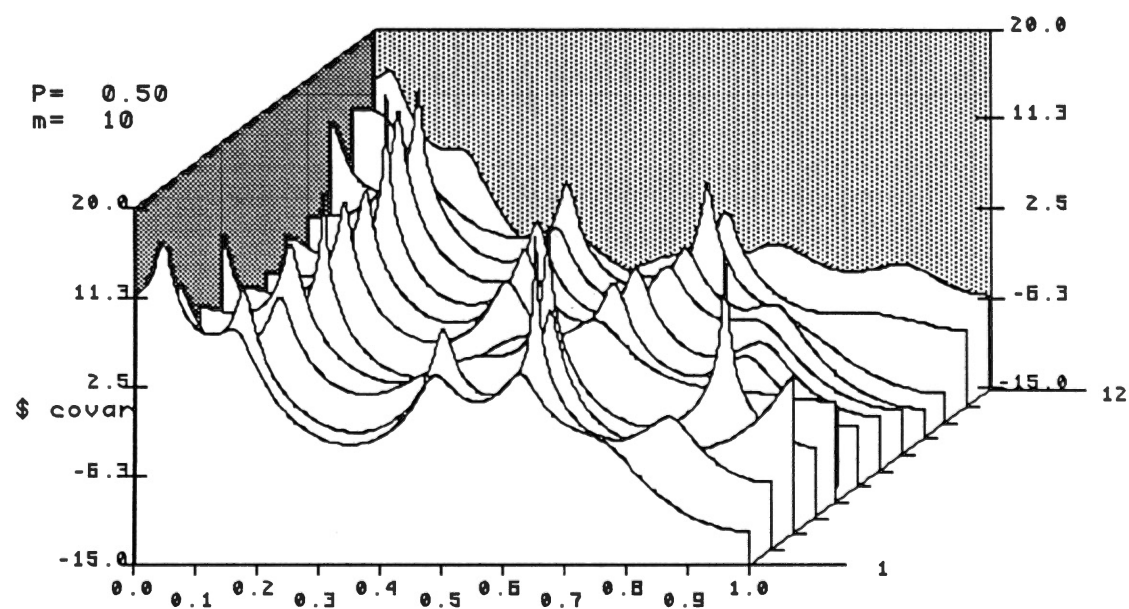
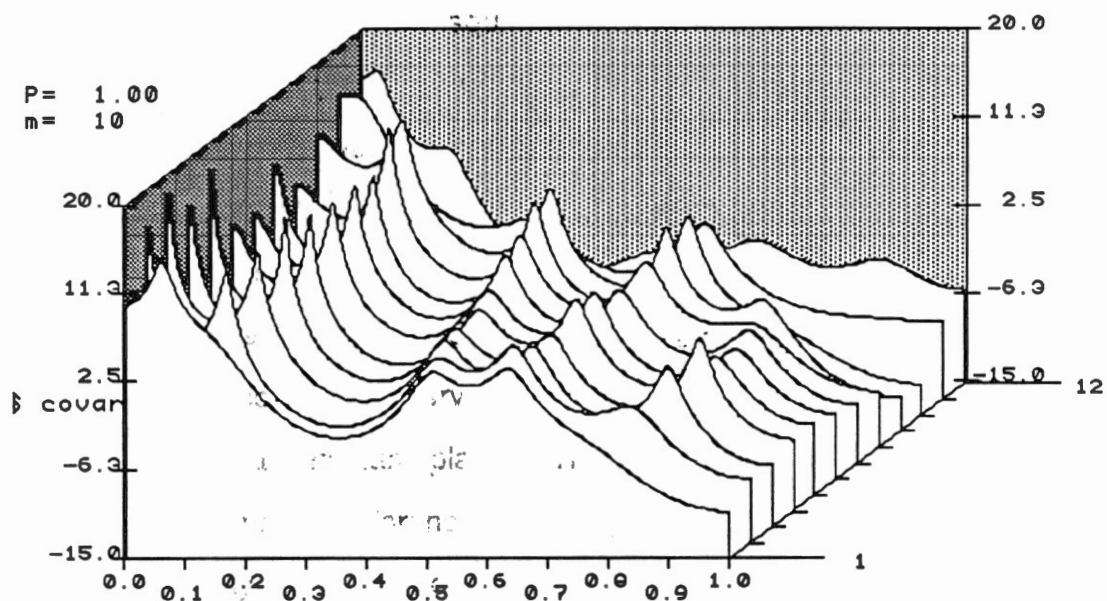
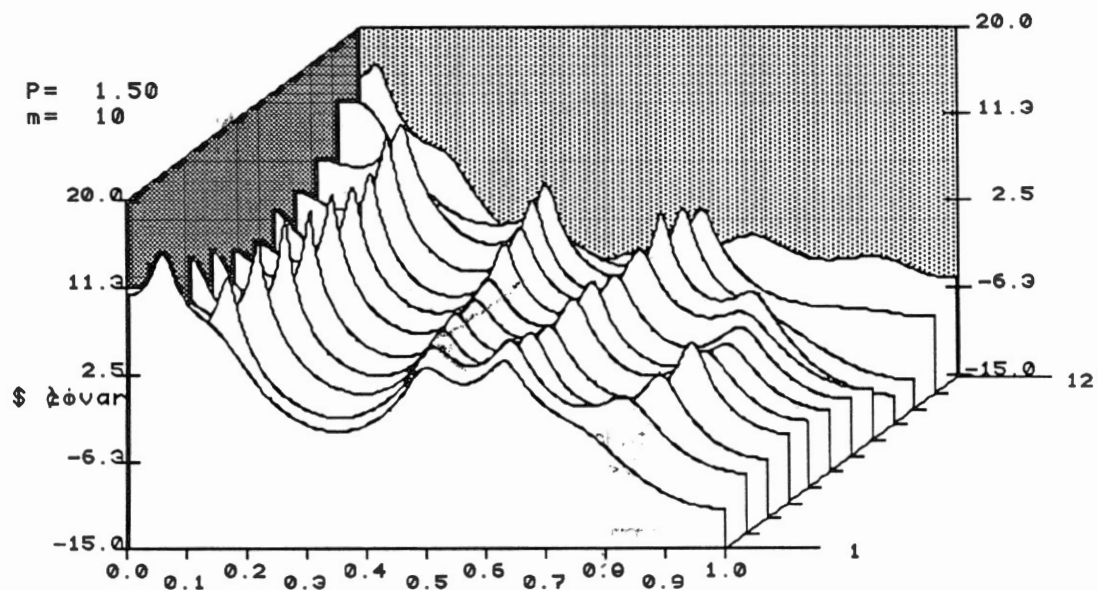
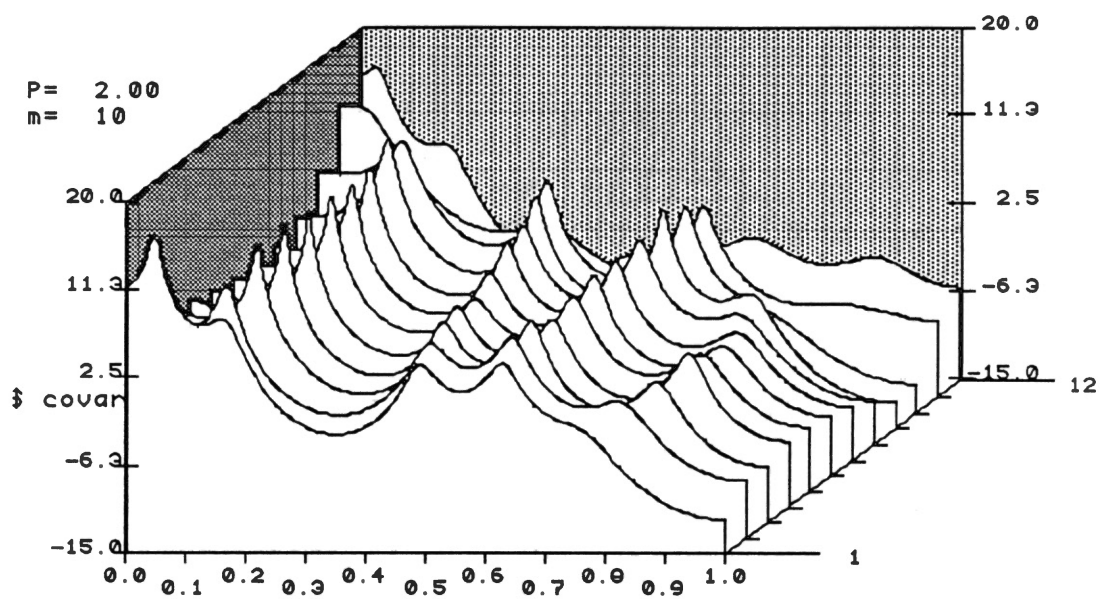
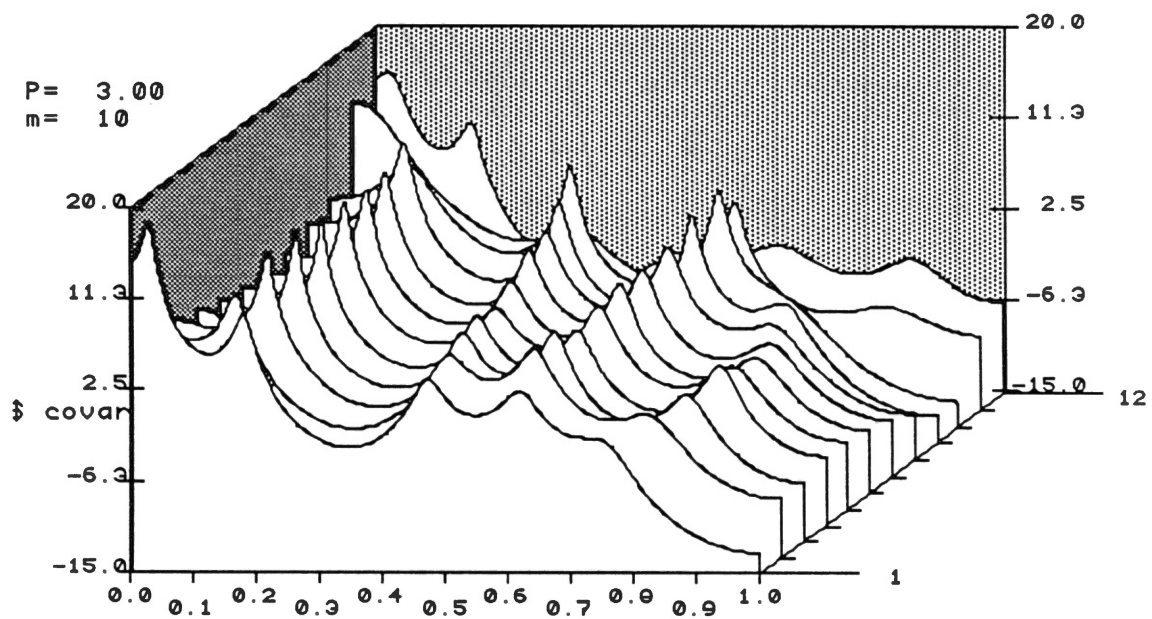
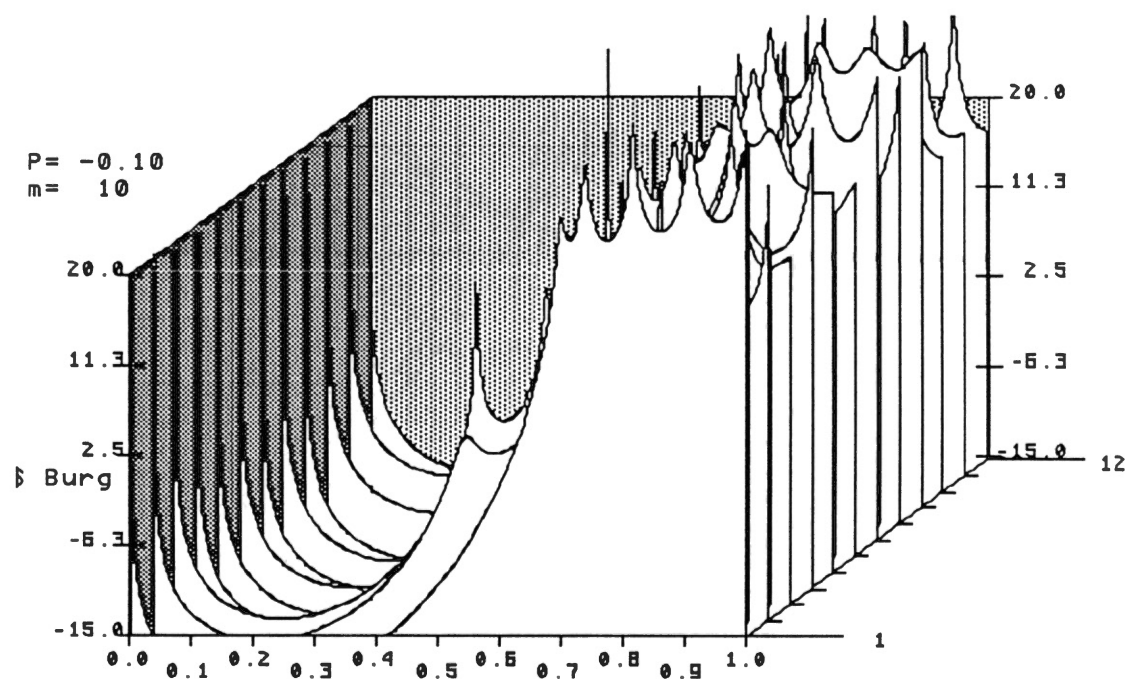
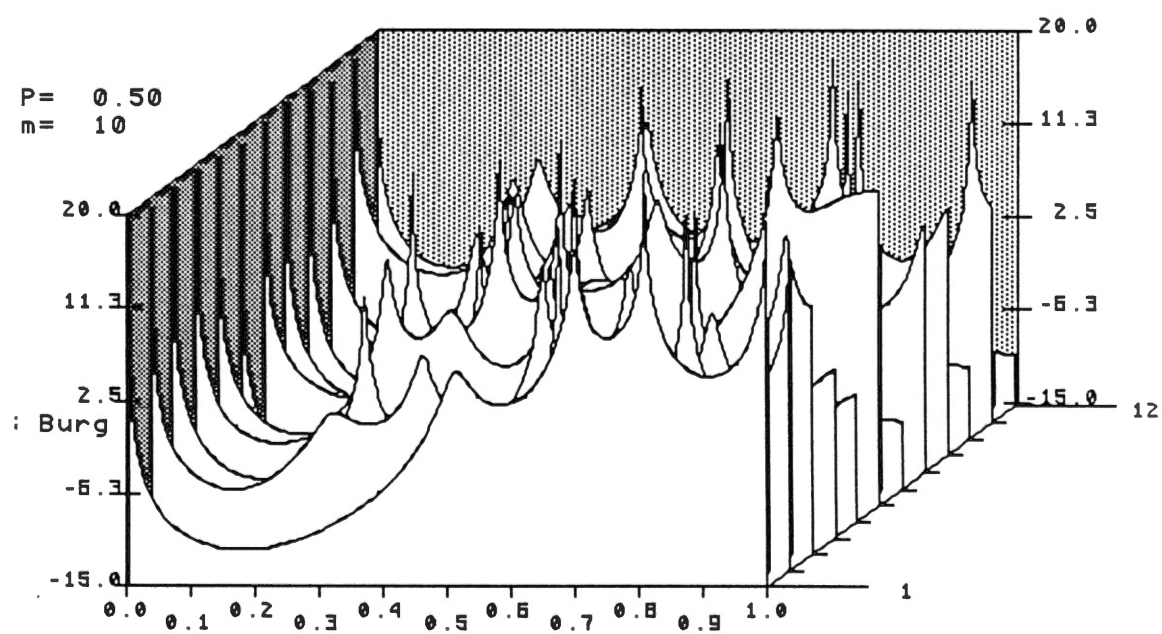
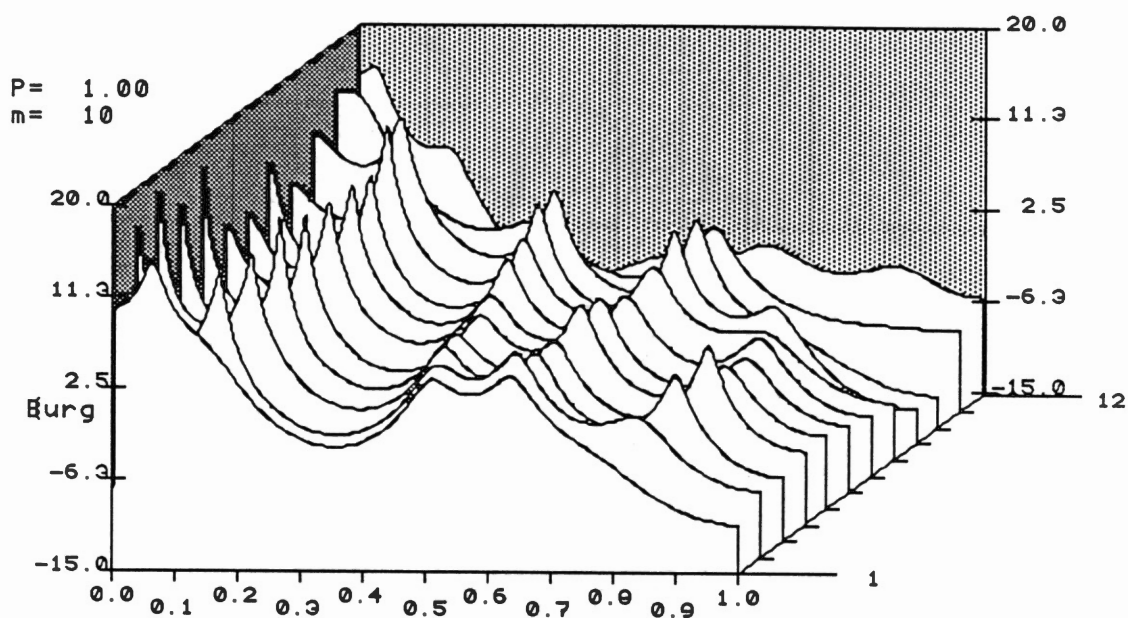
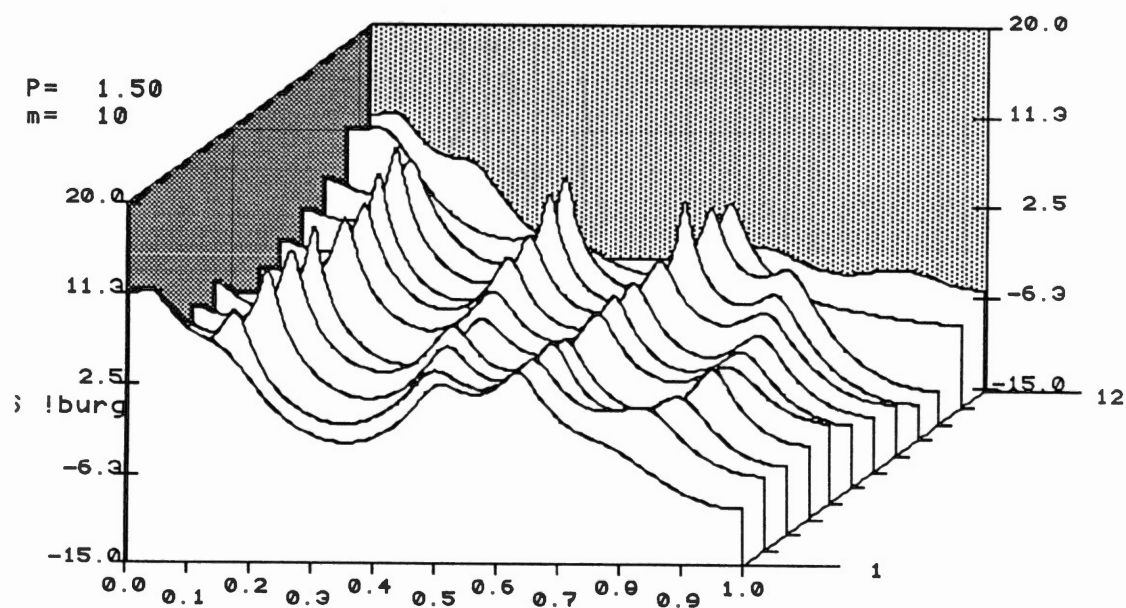


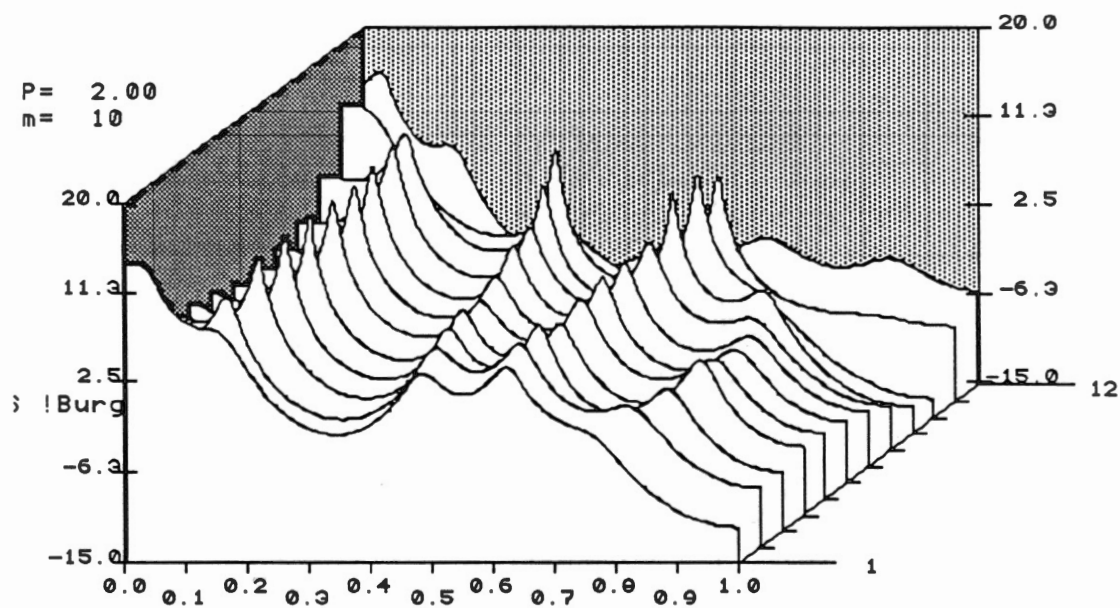
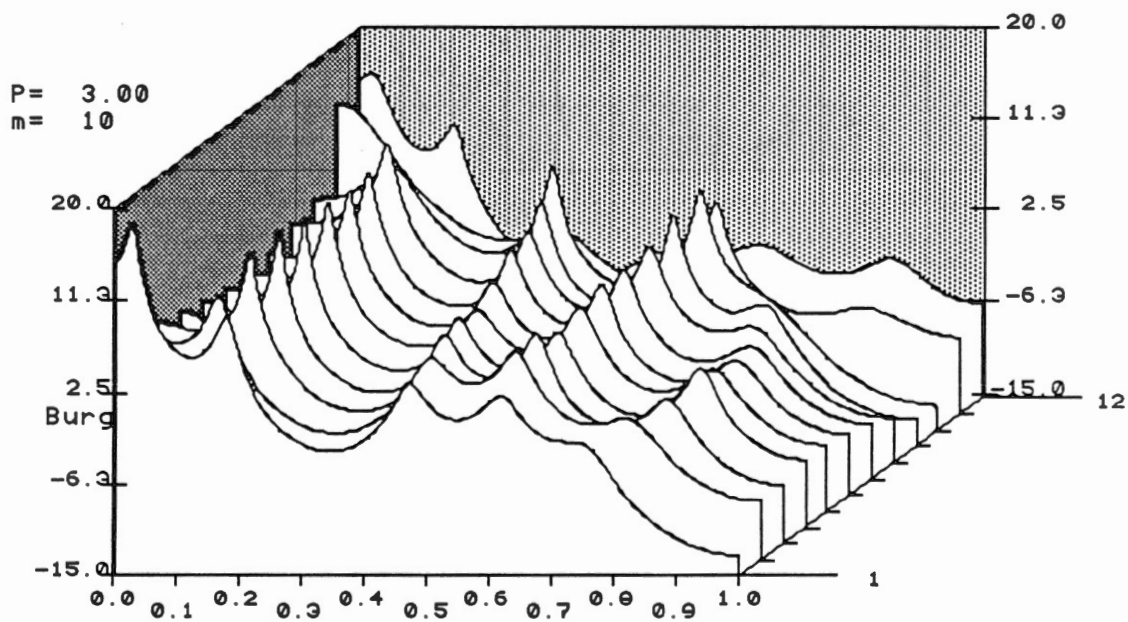
Figure 32: Spectral Surface, Covariance Model, "Cats", $\rho = 0.5$

Figure 33: Spectral Surface, Covariance Model, "Cats", $p=1.0$ Figure 34: Spectral Surface, Covariance Model, "Cats", $p=1.5$

Figure 35: Spectral Surface, Covariance Model, "Cats", $p=2.0$ Figure 36: Spectral Surface, Covariance Model, "Cats", $p=3.0$

Figure 37: Spectral Surface, Burg Model, "Cats", $p=-0.1$ Figure 38: Spectral Surface, Burg Model, "Cats", $p=0.5$

Figure 39: Spectral Surface, Burg Model, "Cats", $p=1.0$ Figure 40: Spectral Surface, Burg Model, "Cats", $p=1.5$

Figure 41: Spectral Surface, Burg Model, "Cats", $p=2.0$ Figure 42: Spectral Surface, Burg Model, "Cats", $p=3.0$

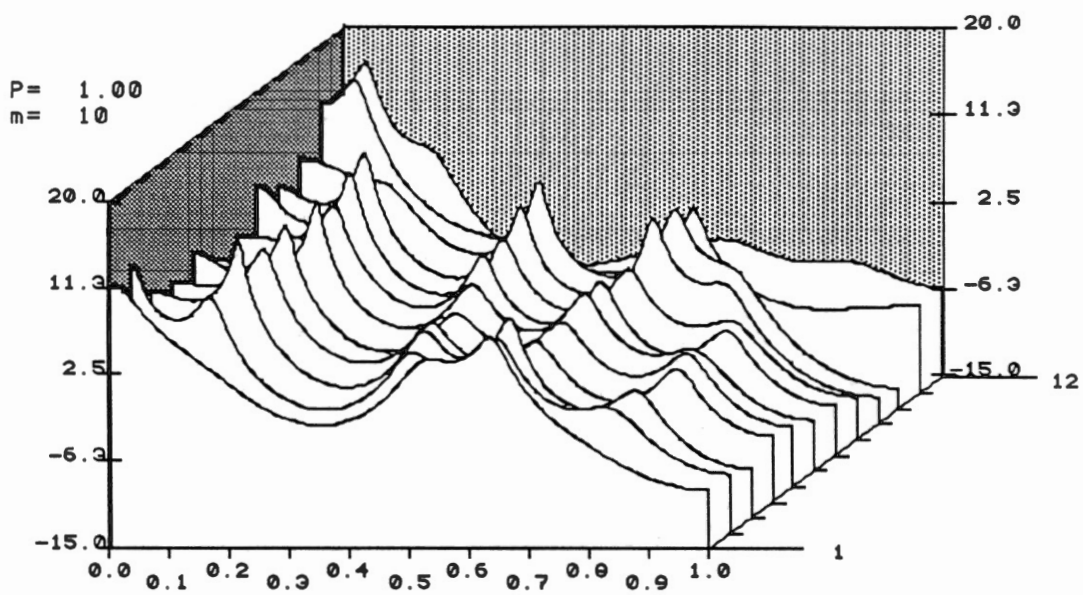


Figure 43: Spectral Surface, Weighted median (L_1) Model, "Cats"

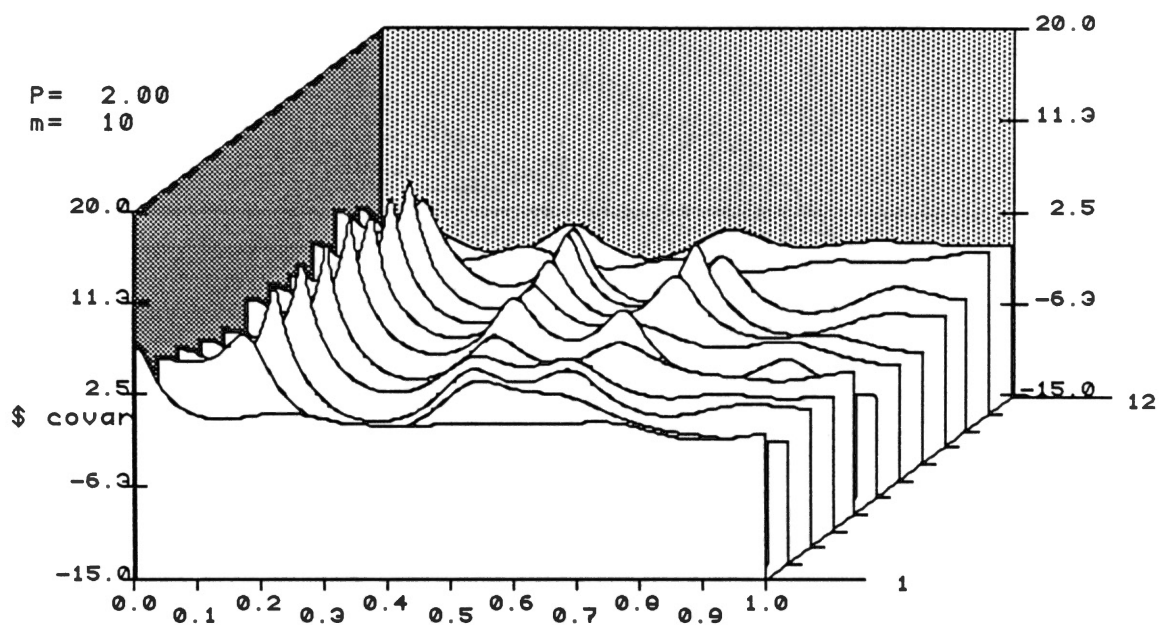


Figure 44: Spectral Surface, 20 dB SNR, Covariance Model, "Cats", $p=2$

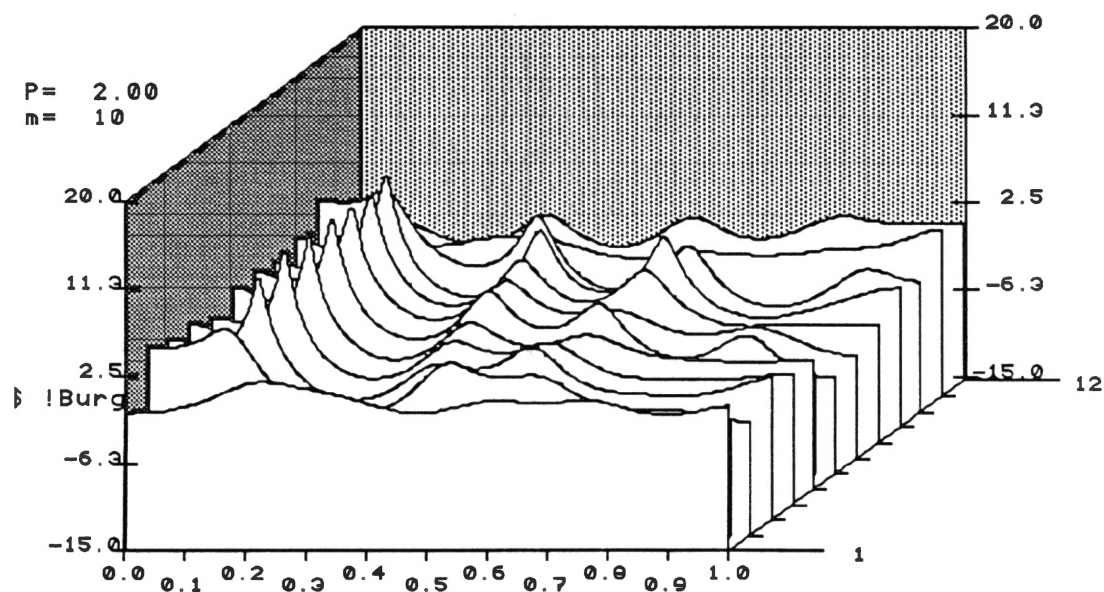


Figure 45: Spectral Surface, 20 dB SNR, Burg Model, "Cats", $p=2$

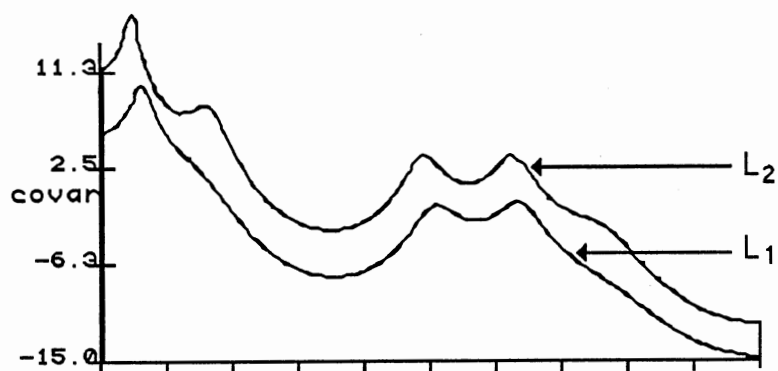


Figure 49: Comparison of L_2 Covariance and L_1 Covariance Spectra

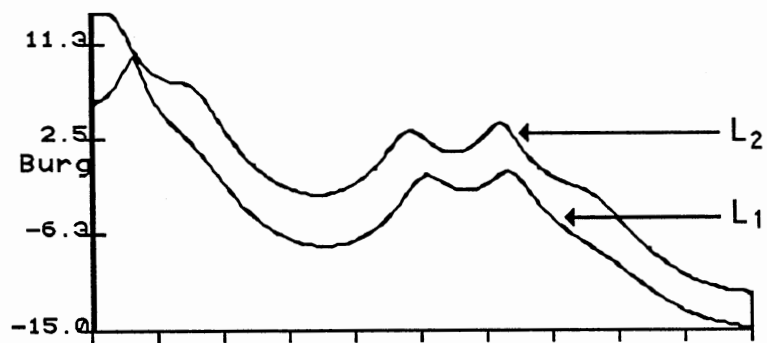


Figure 49: Comparison of L_2 Burg and L_1 Burg Spectra

CHAPTER VII

MARKOV CHAINS IN SPEECH RECOGNITION

Pattern Recognition Concepts

To properly discuss sequential pattern recognition as applied to speech recognition, a review of some background material is in order. This section will summarize relevant pattern recognition concepts and speech recognition research and will touch briefly on cochlear (inner ear) modeling. The key point here is that "classical" pattern recognition techniques need stationary features to function properly but speech and many other signals are not stationary; a technique that can parameterize time variations of the signal may be more appropriate.

Statistical pattern recognition techniques are based on the assumption that classification can be based on probabilistic measures and that these probabilities can be derived (Duda and Hart, 1973, p. 10). One of the most popular techniques for statistical pattern recognition is the Bayesian decision rule under the assumption of a Gaussian (normal) distribution. This decision rule can be described as follows:

Given, as an example, measurements made from two processes ω_1 and ω_2 . Bayes' rule says that (Duda and Hart, 1973, p. 11):

$$P(\omega_j|x) = P(x|\omega_j)P(\omega_j) / P(x) \quad j = 1,2 \quad (7.1)$$

Where:

$P(\omega_j|x)$ means the probability of class ω_j given an observation x (*a posteriori* probability)

$P(x|\omega_j)$ is the probability density function for x conditioned on class ω_j

$P(\omega_j)$ is the *a priori* probability of class ω_j

$P(x) = P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)$ for two class case

It can be shown (Duda and Hart, p. 17) that the decision that minimizes the probability of error, assuming a zero-one loss function, is given by:

If $P(\underline{x}|\omega_i)P(\omega_i) > P(\underline{x}|\omega_j)P(\omega_j)$ for all $i \neq j$, choose $\underline{x} \in \omega_i$. Choose $\underline{x} \in \omega_j$ otherwise.

Rearranging this, a decision function can be derived based on the ratio of the likelihood functions (again assuming a zero-one loss function):

$$d_{ij}(\underline{x}) = P(\underline{x}|\omega_i)P(\omega_i) / P(\underline{x}|\omega_j)P(\omega_j) \quad (7.2)$$

If $d_{ij}(\underline{x}) > 1$, choose class ω_i . Otherwise choose class ω_j . ($i \neq j$)

Figure 50 illustrates this concept in the one dimensional case.

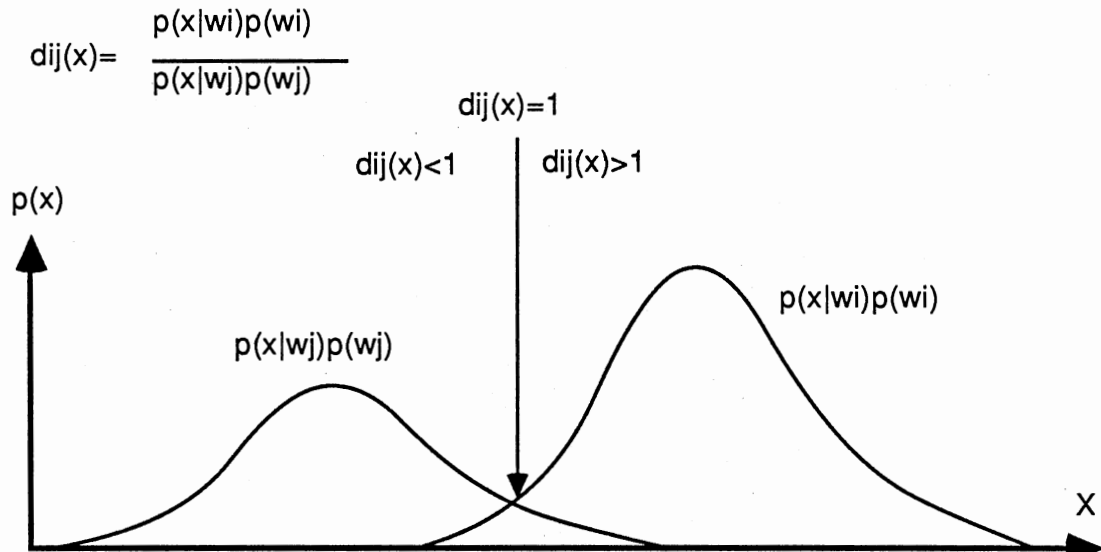


Figure 50: Minimum Probability of Error Decision Function

If the conditional densities are assumed to be multivariate Gaussian, then they will be of the form

$$P(\omega_i) = \frac{1}{(2\pi)^{n/2} |\mathbf{C}_i|^{1/2}} e^{-\left(\frac{1}{2}\right)(\mathbf{x}-\mathbf{m}_i)^t \mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)} \quad (7.3)$$

where \mathbf{C}_i is the covariance matrix and \mathbf{m}_i is the mean vector for the class.

Substituting this expression into the decision function and taking the natural logarithm, the minimum probability of error decision function d_i for M classes is (Tou and Gonzalez, 1974, p. 120):

$$d_i(\underline{x}) = \ln\{P(\omega_i)\} - (n/2) \ln\{2\pi\} - (1/2) \ln\{|\mathbf{C}_i|\} - (1/2)[(\underline{x} - \underline{m}_i)^T \mathbf{C}_i^{-1} (\underline{x} - \underline{m}_i)]$$

$$i = 1, 2, \dots, M \quad (7.4)$$

The overall decision surface is derived by resolving all pairwise dichotomies; for instance, $d_{12}(\underline{x}) = d_1(\underline{x}) - d_2(\underline{x})$. In general, $(M)(M+1)/2$ such dichotomies must be resolved during the classification process (Tou and Gonzalez, p. 43). Since the decision function is taken pairwise here, terms that do not depend on i can be dropped; thus the term $(n/2) \ln\{2\pi\}$ can be discarded.

The decision function d_i is a hyperquadric in n -space, where n is the dimension of the vector \underline{x} and \mathbf{C} has arbitrary entries (although recall that a covariance matrix is always symmetric and positive definite for $p(x) \neq 0$). In the two class case, $n=2$ so the decision function will either be a line, parabola, circle, ellipse or hyperbola. The special case of a linear (hyperplane) decision function occurs if the covariance matrices for all classes are equal because terms not depending on i can be dropped. In an even more special case, if the matrices are equal and diagonal, then the decision hyperplane lies perpendicular to the line connecting the cluster centroids (means). Finally, if all matrices are equal, diagonal, and each entry has the same value ($\mathbf{C} = \sigma^2 \mathbf{I}$, \mathbf{I} =identity matrix), then the decision hyperplane lies on the perpendicular bisector of the line connecting the centroids. Figure 51 shows the decision function and equiprobability contours for two of the cases described above.

Not all feature sets are normally distributed, however; other common distributions that are used in speech processing are the gamma and Laplacian densities. The gamma distribution is considered to be a better

approximation to the amplitude distribution of speech (Paez and Glisson, 1972).

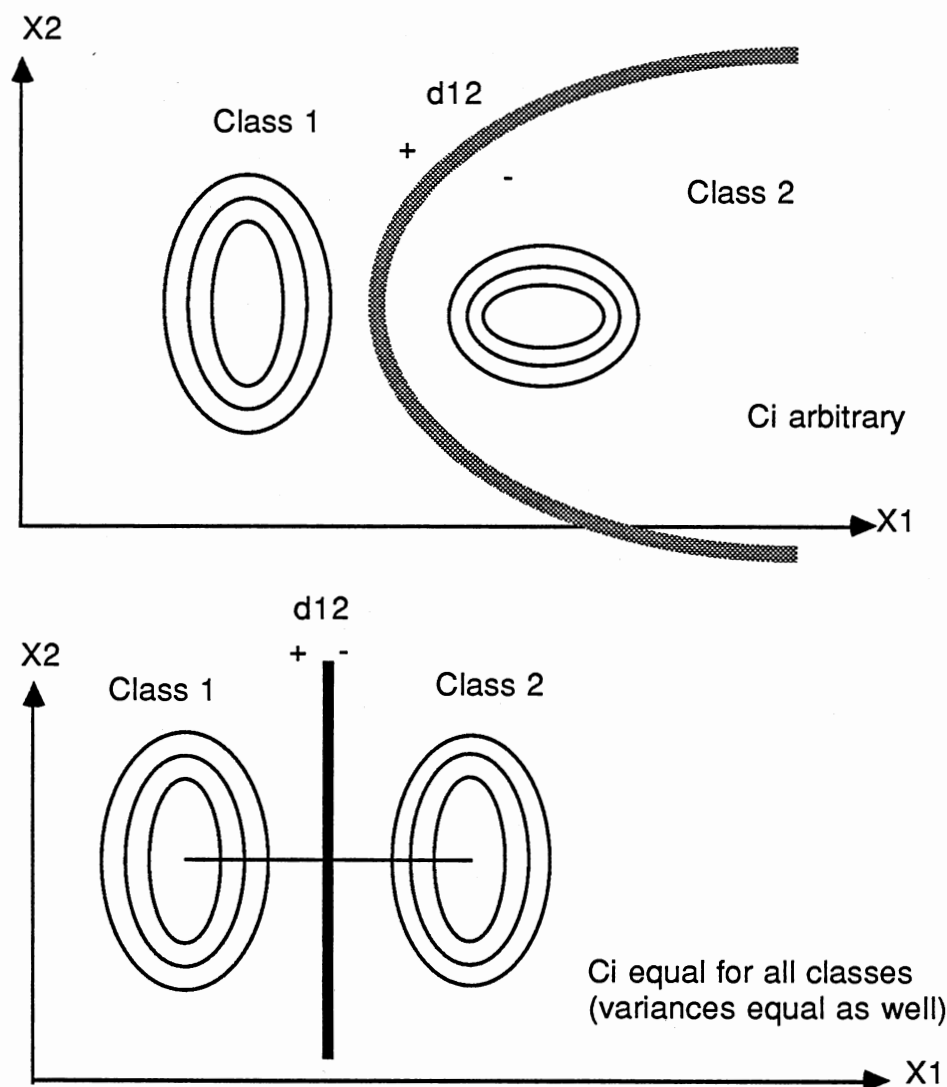


Figure 51: Decision Surfaces for Gaussian Distributions

The previous discussion implicitly assumes that the measurements are from a stationary process, although stabilization techniques have been used

with some signals to allow these functions to be used. In general, however, measurements of a non-stationary signal such as speech will yield time-varying features. (Stationarity will be defined here as wide sense or second order - the mean and covariance of a process are independent of time.) To handle this problem, a promising technique has received increased attention in recent years: sequential classification (Jelinek, 1976, Kaneko and Dixon, 1983, Kazakos, 1978). To illustrate the concept, consider Figure 52; a non-stationary signal can generally be considered stationary if analyzed over a sufficiently short time interval, which makes estimation of the mean and variance measurements meaningful. A sequential technique would somehow attempt to match the locus of the feature cluster (over time) to some reference locus. Thus the inclusion of time as a component of the feature vector increases the dimensionality of the pattern space by one, in a sense; but no assumption can generally be made about the probability distribution of the time component of the vector. Other problems arise which make extension of the previously described techniques impractical: 1) How can the features be computed? You generally can only measure a single vector at each point in time, so cluster means and moments are impractical to compute at each instant. 2) How can time similarity be measured? The non-stationary properties of the signal often lead to a distortion of the time axis from one observation to the next. For example, two different speakers might utter the same word at different rates, both the entire word as well as individual phonetic units within it.

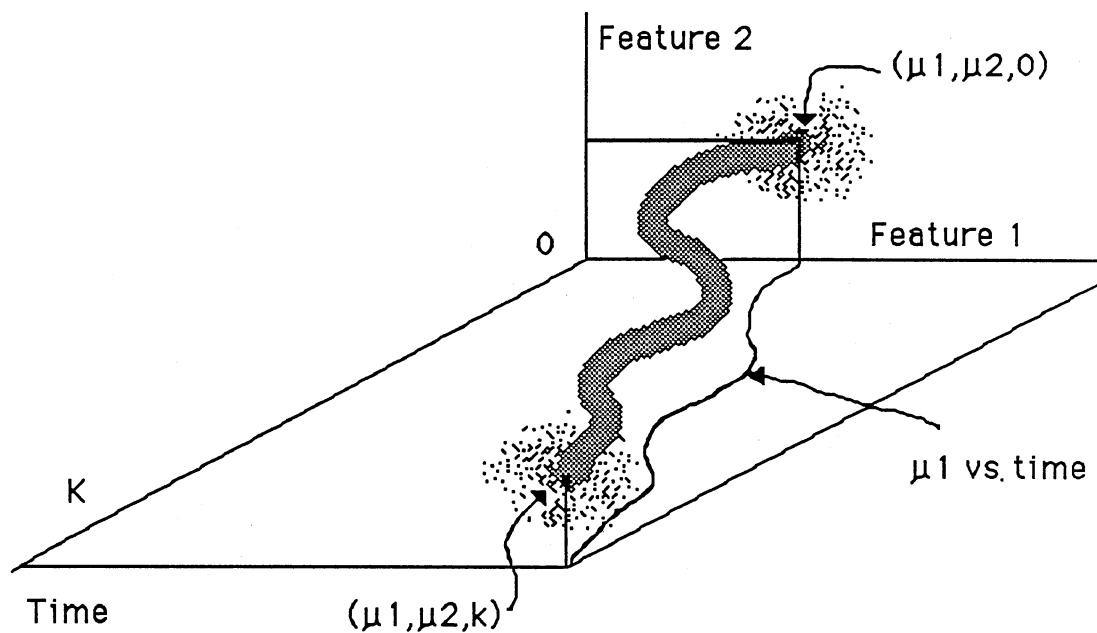


Figure 52: Variation of a feature Cluster Over Time

There are at least two ways to approach the problem: syntactical and statistical. Syntactical techniques attempt to parse the time variation into a set of primitive elements. Signal recognition is performed by analyzing the "sentence" formed from the primitive "alphabet" based on some underlying "grammar" or syntax. An example of how this might work is shown in Figure 53.

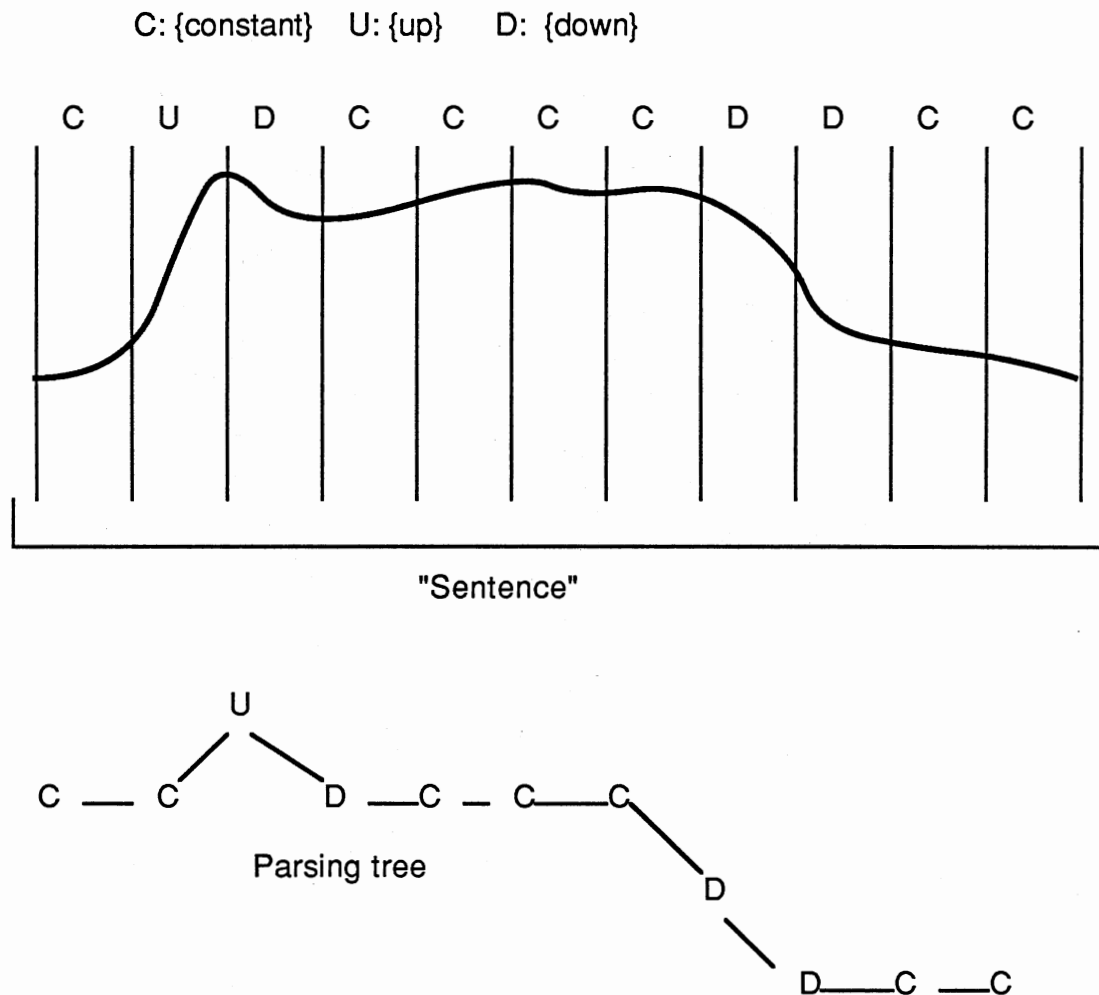


Figure 53: Syntactical Pattern Recognition

As noted by Tou and Gonzalez (1974, p. 340), a purely syntactical approach is often not suited for a sequential algorithm. Most signals have some statistical uncertainty in them; the locus shown in Figure 52 may in fact be a probabilistic function rather than a deterministic one. The technique introduced by Itakura (1975) was among the first to attempt to match an unknown locus (derived from linear prediction coefficients taken over time) to

a reference locus. The algorithm yields an optimal solution under a constraint on the "warping" of the unknown locus (Levinson, 1985). Itakura chose the allowable "warping" (called the continuity condition) to correspond to a linear mapping with a slope between 1/2 and 2. Dynamic programming techniques can then be used at each observation interval to pick the most likely value of the "true" measurement, taking into account the reference measurement at that time. The metric - Levinson (1985) calls it a "distortion function" because it is not a true norm since it does not always obey the triangle inequality - is based on the log of the ratio of an autocorrelation weighted vector distance. Formally, the metric is (Levinson, 1985)

$$\rho(\underline{x}, \underline{y}) = \text{Log} \left(\frac{\underline{a}_x \mathbf{R}_x \underline{a}_x^t}{\underline{a}_y \mathbf{R}_y \underline{a}_y^t} \right) \quad (7.5)$$

where: \underline{x} and \underline{y} are data segments

\underline{a}_x is a p-th order linear predictive coefficient (LPC) vector derived from \underline{x}

\underline{a}_y is a p-th order LPC vector derived from \underline{y}

\mathbf{R}_x is the autocorrelation matrix of \underline{x}

\mathbf{R}_y is the autocorrelation matrix of \underline{y}

This function ρ can be used to assign transition weights in a dynamic programming treillis, as will be seen when dynamic programming is examined in detail later.

Figure 54 shows a diagram of the dynamic time warping process under the $1/2 \leq \text{slope} \leq 2$ constraint. The two vectors are forced to be equal at the

endpoints, yielding a parallelogram with slopes corresponding to the constraints as the boundary of the allowed warping process. In terms of the original time locus, this diagram describes how much the individual measurements can be "stretched and pulled" (linearly) to make it line up with a reference.

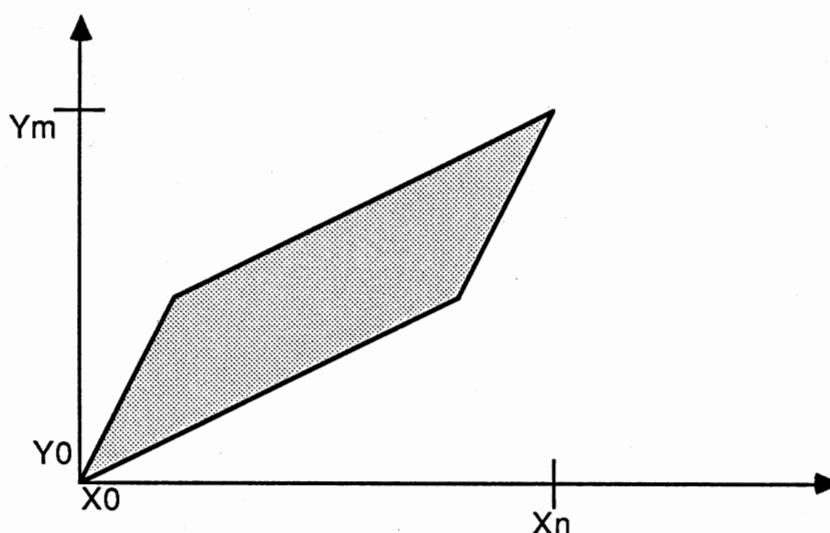


Figure 54: Dynamic Programming

Speech Recognition

A logical topic to examine next is the application of the pattern recognition techniques previously described to speech recognition. The field of automatic speech recognition (ASR) dates back at least to the early 1950's (Davis, 1952 cited by Flanagan *et al*, 1980). Rather than embark on a comprehensive history of the field, an attempt will be made to review here the

most recent techniques in the field of sequential classification, with the intent of identifying the "incremental" research areas alluded to earlier.

The most intensive speech recognition study was funded by the Defense Advanced Research Projects Agency (DARPA) in late 1971; after nearly 200 person-years of research (Flanagan *et al*, 1980), the study culminated with a demonstration of four systems in late 1976 (Klatt, 1977). Two systems were built by Carnegie-Mellon University (CMU); one was built by Bolt, Beranek, and Newman and the other was built by Systems Development Corp. The only system that met the DARPA study goals of 1000-word vocabulary from multiple speakers with less than ten percent error was the CMU Harpy system. Harpy used a dynamic time warping algorithm to compute the match between the incoming speech signal and the "acoustic segment" (phoneme or syllable level) templates. Once the segments were identified, they were analyzed by a syntax-directed search algorithm. In a sense, a model of allowed phoneme transitions was built during training which allowed the search algorithm to determine the most likely sequence of phonemes, which in turn determined the most likely word (after Baker [1975]). The most likely sequence was determined using a "best-few beam search" technique that has its roots in artificial intelligence and semantical network theory; Harpy used 15,000 states in its allowed transition network, making the search task a formidable one. Figure 55 shows a block diagram of the Harpy system.

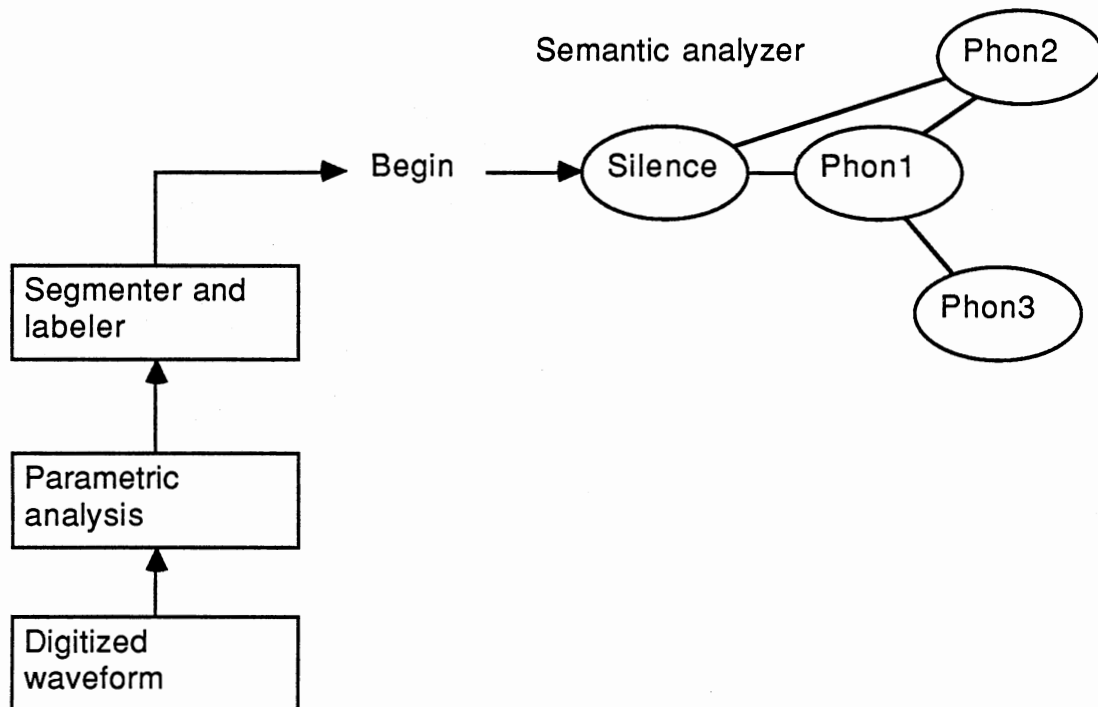


Figure 55: Block Diagram of Harpy System

Bell Laboratories (now AT&T Bell Laboratories) used the work of Baker (1975), Portiz (1982) and others to extend the concept of sequential models for recognition (Rabiner *et al*, 1983). Their approach to isolated word recognition used a statistical measure of similarity instead of the syntax-directed search of Harpy. This means the Bell system depends more on a probabilistic model for speech than does the CMU system. Both employ a linear predictive (autoregressive) spectral estimator, but the Bell system uses a technique called vector quantization (VQ) to reduce the dimensionality of the LPC vector. If each coefficient is k bits wide and there are M coefficients, a matrix model would require $(2^{Mk})^2$ entries, which is prohibitive. The VQ algorithm

compresses the vector dimension down to a manageable size by constructing a small "codebook" and assigning values to the entries in it. The algorithm assigns values to the codebook entries according to a distance preserving scheme (an L_2 metric), which optimizes the dispersion of the codebook entries as best it can to preserve the distance which existed in M-space. As the word is spoken, the compressed vector will trace out the locus of the spectrum (after a fashion) over time, yielding the time frequency locus that is desired. Bell Lab's parametric representation of this variation is called Hidden Markov Modeling (HMM); it will be described in detail later in this chapter. Bell has achieved results from the HMM that are only as good as existing isolated word recognition systems but throughput is about 17 times that of dynamic time warping based systems (Rabiner *et al*, 1983).

Before proceeding to a discussion of sequential models, an aside is in order. Both the dynamic time warping and the hidden Markov modeling techniques use LPC as the spectral pre-processor. Zue (1985) has observed that:

For the most part, current speech recognition strategies are based on a spectral representation of the speech signal that borrows heavily from models of speech production. Such models are very useful for characterizing the signal and for helping to identify the basic acoustic attributes that are important for phonetic identity.... Such models are also clearly appropriate in analysis/synthesis systems, where the goal is to produce as accurate a reconstruction as possible of the measured signal. Often, the form of spectral representation that is used for synthesis is assumed also to be appropriate for recognition. Yet the two tasks are really quite different, and there is no reason to believe that what works for synthesis is suitable for recognition.

These are words to ponder, especially since automated speech recognition systems perform so poorly compared to their human counterparts. While no claim can be made for modeling the human cognitive process as a probabilistic function of a Markov chain (although it can be argued that it is as good as any), it certainly seems reasonable that a recognition system based on a model of the ear's function would work at least as well as one using LPC derived features.

Overview of Cochlear Modeling

Modeling the function of the ear, particularly the transducer function of the inner ear or cochlea, is a difficult task, and has been studied for over 50 years (Zwislocki, 1980). Jont Allen (1985) summarized the current state of the art in his comprehensive paper in the *ASSP Magazine*, where he describes the currently accepted models for sound perception, particularly the cochlea. The cochlea, which acts as a complex mechanical to electrical transducer, has been the focus of the bulk of research in perception, since it is the critical link in the perceptual chain.

A simplified physical model of the ear is shown in Figure 56 (from Zweig *et al*, 1976); sound enters the ear and travels down the auditory canal where it vibrates the eardrum (tympanic membrane) which transmits the vibrations to the cochlea via the small bones of the inner ear, called the ossicles (commonly known as the hammer, anvil, and stirrup). The stapes (stirrup)

couples the vibrations into the fluid (perilymph) of the spiral-shaped cochlea, which sets up a traveling wave in the fluid (Zweig *et al*, 1976). As the wave propagates through the cochlea, hair cells (cilia) on the basilar membrane are switched "on" and "off". Surprisingly enough, this is usually modeled as a half-wave rectifier or zero crossing detector (Lyon, 1982; Allen, 1985; Pfeiffer *et al*, 1972). Allen's complete model is a transmission line model of the acoustic/mechanical properties of the auditory canal, eardrum, ossicles, and scala vestibuli combined with a transduction model (rectifier or zero crossing detector). This model, especially if the transmission line is two or three dimensional, has yielded reasonable experimental confirmation, including "capture effect" due to non-linear (limiting) amplitude response (Allen, 1985). Computationally, the model involves 1500 to 2000 bandpass filters of logarithmic spacing over the normal auditory range, with a zero crossing counter on the output of each - a formidable, if not prohibitive, scheme for a speech recognition system. A speech recognizer would not need the full bandwidth but research is needed to determine the number and characteristics of the filters that would perform effectively. To the author's knowledge, no extensive study has been made of this problem, although Zue (1985) alludes to a Ph.D. dissertation (by Seneff at MIT) which has analyzed the hair cell firing process to yield an "improved" spectral model during speech excitation. Application of this model to automatic speech recognition has apparently not been attempted, however.

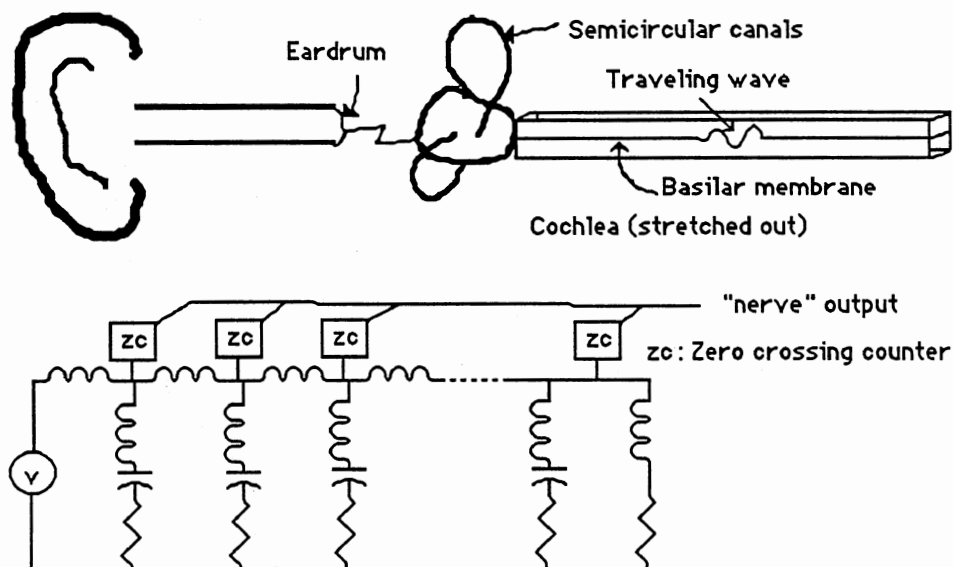


Figure 56: Cochlear Modeling

So reasonable cochlear models exist and are beginning to find their way into speech recognition studies; this brings us back to the original problem - speech recognition using sequential models. The next topic to be addressed will be properties of Markov chains, which will lay the groundwork for analysis of the Viterbi algorithm, which is a way of testing Markov processes for similarity. Some experiments in speech recognition follows this; finally, some areas for future research will be examined.

Properties of Markov Chains

To better understand the properties of sequential techniques, a more thorough analysis of the underlying theory must be undertaken. In particular, properties of stochastic processes called Markov chains will be examined. A first-order Markov chain is a discrete sequence of random variables so that for any n , X_{n+1} (the "state" of the process at time $n+1$) is conditionally independent of X_0, \dots, X_{n-1} given X_n (Çinlar, 1975). More formally:

$$P\{X_{n+1} = j \mid X_0, \dots, X_n\} = P\{X_{n+1} = j \mid X_n\} \quad (7.6)$$

for all j from a countably finite state space.

Which is to say that the next state is independent of all past states if the current state is known. A first order Markov process can also be shown to be a first order autoregressive process (Papoulis, 1965, p. 537). Note that no probability density is implicit in this definition.

One would be tempted to believe that this property would have limited usefulness, since many phenomena are conditionally dependent on more history than just the previous value. Higher order Markov processes can be defined but are rarely discussed. Why? An m -level (number of states), k -order (past dependency) Markov chain can always be rewritten as a first order one with m^k states for a finite state space (Kazakos, 1982). Consider an example of a two state (binary) Markov chain of order two - each bit depends both on the previous one and the one before that. It could be rewritten as a first order chain with four states, where each state represents not a single bit but two bits. This rewriting rule is widely used in digital communications systems (Viterbi, 1967; Forney, 1973; Hayes, 1975; Jelinek, 1969)

A useful way to visualize a Markov process is with a signal flow graph, which can also be called a state transition diagram. Figure 57 shows the flow graph of a Markov process, where the numbers on the branches indicate the probability of going from one state to another or staying in the same state. Note that the probability of making a state transition depends only on the current state, as indicated in the expression above. In this example, the system can only be in one of four states at any instant and can only change states at discrete time intervals (it can also remain in the current state, of course - if there is a non-zero probability). The transition probabilities can be written in matrix form as follows:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.7 & 0.0 & 0.0 \\ 0.45 & 0.1 & 0.45 & 0.0 \\ 0.0 & 0.45 & 0.1 & 0.45 \\ 0.0 & 0.0 & 0.7 & 0.3 \end{bmatrix}$$

where p_{ij} is the probability of a transition from state i to state j .

Time varying transition matrices can be defined, although the literature describes only the invariant case since, it is assumed, a time varying process can be described as a series of time invariant processes if the analysis interval is sufficiently short.

MARKOV CONNECTIVITY PLOT
 NUMBER OF STATES: 4
 FRAME SIZE: 10
 DATA SOURCE: TEST (0)

FILE NUMBER: 1017
 START, NUM FRAMES: 1, 1
 THRESHOLD: 0.000000
 FRAME: 1

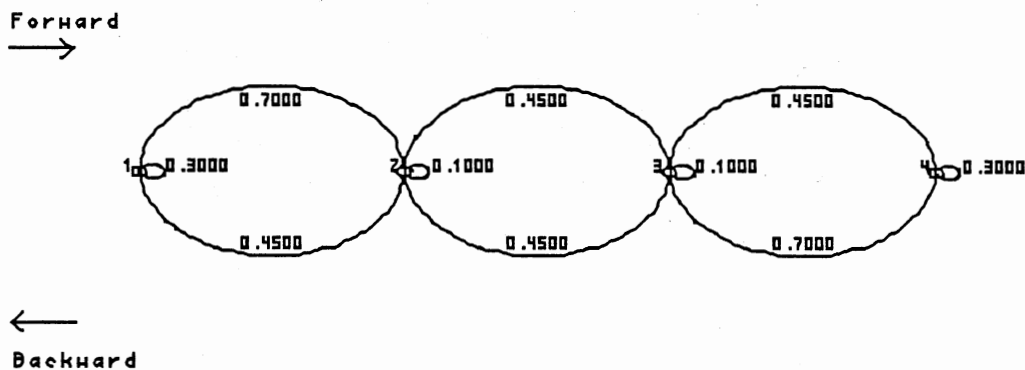


Figure 57: Signal Flow Graph for Example

The matrix \mathbf{P} is called a stochastic or Markov matrix because it satisfies the following conditions in a k -state chain (Çinlar, 1975):

$$p_{ij} \geq 0 \quad \text{for all } i \geq 1, j \leq k \quad (7.7)$$

$$\sum_{j=1}^k p_{ij} = 1 \quad \text{for all } i \geq 1, j \leq k \quad (7.8)$$

The states in a Markov chain can be described in terms of their transition properties. The primary classifications are (Kemeny and Snell, 1960, p. 35):

1. Ergodic (or irreducible) - any state or group of states in the chain that can be reached from any other state, either directly or through other states. An irreducible chain of only one state is called absorbing; the process can never leave it. Another special case of an ergodic process is called cyclic or periodic. The process moves through a set of states in a definite order, revisiting each with period δ . Figure 58 gives a diagram of the properties of ergodic chains.

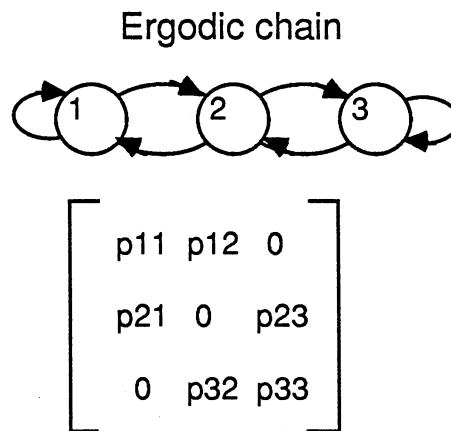


Figure 58: Example of an Ergodic Chain

2. Transient - a state or group of states that can only be visited once. When a process leaves a transient state, it never returns. Figure 59 illustrates this property.

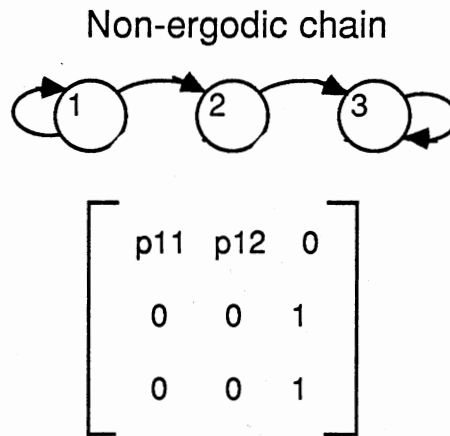


Figure 59: Example of Transient States

These classifications of the states allow some further properties to be derived, such as the limiting probabilities of ergodic states (the limiting probability of a transient state is zero).

Let π be some distribution vector over the finite space which has k states and let π_0 be the initial distribution of the states. The probability distribution of the states at time n will be (Kemeny and Snell, 1960, p. 33):

$$\pi_n = \pi_{n-1} \mathbf{P} \quad \text{or} \quad \pi_n = \pi_0 \mathbf{P}^n \quad (7.9)$$

(most authors define π as a row vector)

As an example, consider the transition matrix given earlier in Figure 57. If the process begins in state two, then $\pi_0 = (0 \ 1 \ 0 \ 0)$. In other words, the probability of being in state two is one, and all other probabilities are zero (since the probabilities must sum to one). The probability distribution for various values of n are then:

n	$\pi_n(1)$	$\pi_n(2)$	$\pi_n(3)$	$\pi_n(4)$
0	0.00000	1.00000	0.00000	0.00000
1	0.45000	0.10000	0.45000	0.00000
2	0.18000	0.52750	0.09000	0.20250
3	0.29137	0.21925	0.38812	0.10125
4	0.18608	0.40054	0.20835	0.20503
5	0.23607	0.26406	0.34460	0.15527
6	0.18965	0.34672	0.26198	0.20165
7	0.21292	0.28532	0.32338	0.17838
8	0.19227	0.32310	0.28560	0.19904
9	0.20307	0.29542	0.31328	0.18823
10	0.19386	0.31267	0.29603	0.19744
11	0.19886	0.30018	0.30851	0.19245
12	0.19474	0.30805	0.30064	0.19657
13	0.19704	0.30241	0.30628	0.19426
14	0.19520	0.30600	0.30270	0.19611
15	0.19626	0.30345	0.30524	0.19504
16	0.19543	0.30509	0.30361	0.19587
17	0.19592	0.30393	0.30476	0.19539
18	0.19555	0.30468	0.30402	0.19576
19	0.19577	0.30416	0.30454	0.19554

Note that the probability vector converges to some final value; for an ergodic chain, this limiting probability, denoted by π , is independent of the starting state of the system. In other words, the probability of being in a specific state after a long time does not depend on the initial state vector of an ergodic process.

The matrix \mathbf{P} can always be subdivided (after suitable permutations) into the following general block form (Kemeny and Snell, 1960, p. 44):

$$\begin{array}{ccc}
 \text{from:} & \begin{array}{cc} \text{ergodic} & \text{transient} \end{array} & \text{to:} \\
 \mathbf{P} = & \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{R} & \mathbf{Q} \end{bmatrix} & \begin{array}{c} \text{ergodic} \\ \text{transient} \end{array}
 \end{array} \quad (7.10)$$

where:

S is the set of ergodic states.

R causes transitions from transient (**Q**) to ergodic (**S**) states.

Q is the set of transient states.

0 is a zero matrix (since the process cannot leave the ergodic states).

As the matrix **P** is raised to successively higher powers, the probability of being in a set of ergodic states approaches one. It should be noted in passing that the matrix **S** may consist of several ergodic subchains and that the subchain the process is "captured" in as n becomes large depends on the initial probability distribution π_0 . For an ergodic chain or subchain, the limiting probability can be found by solving the following system of equations:

$$\pi = \pi \mathbf{P} \quad \text{and} \quad \pi \mathbf{1} = 1 \quad (\mathbf{1} \text{ is a vector of ones}) \quad (7.11)$$

where one of the equations in the first expression can be discarded since the system of equations is not of full rank (Çinlar, 1975, p. 154).

For the example given previously, solving for the limiting probability π yields:

$$\pi = (9/46 \quad 7/23 \quad 7/23 \quad 9/46) = (0.1957 \quad 0.3043 \quad 0.3043 \quad 0.1957).$$

There are many other interesting properties of Markov chains. One area that has, until very recently, received little attention since its early development (to the author's knowledge) is systems analysis of Markov chains using what we now call state variable techniques. The original work was done by at least three researchers at MIT during the late 1950's (Sittler, 1956; Huggins, 1957; Howard, 1960) and has been recently mentioned by Mullis and Steiglitz (1972) and Hershey and Yarlagadda(1986). The concept closely follows modern linear system theory: given the transition matrix \mathbf{P} and the initial probability vector $\pi_0 = (\pi_0(1) \quad \pi_0(2) \quad \pi_0(3) \quad \dots)$, the probability at time n given previously in equation (7.9) (modified to reflect $n+1$),

$$\pi_{n+1} = \pi_n \mathbf{P} \quad (7.12)$$

can be treated as a standard state variable equation where there is assumed to be no input to the system other than the initial state π_0 . Note that π_n is a row vector so this expression is not entirely in standard state variable form - using \mathbf{P}^T and rearranging the equation would fix this. We shall continue to use the row vector form to be consistent with the statistical literature.

Taking the Z transform of the above and solving for $\pi(z)$ yields:

$$z (\pi(z) - \pi_0) = \pi(z) \mathbf{P} \quad (7.13)$$

$$\pi(z) [z\mathbf{I} - \mathbf{P}] = z \pi_0 \quad (\mathbf{I} \text{ is the identity matrix}) \quad (7.14)$$

$$\pi(z) = z \pi_0 [z\mathbf{I} - \mathbf{P}]^{-1} \quad (7.15)$$

Howard (1960) proceeds with the analysis to yield the impulse response; we shall do the same here, using the example given previously.

The transform matrix can be computed efficiently using the technique described by Frame (1964), which is succinctly summarized by Hershey and Yarlagadda (1986). The algorithm can be derived from the Cayley-Hamilton theorem (Brogan, 1974) which gives a technique for computing the inverse of a matrix (and its characteristic equation, of course) recursively. Briefly, the procedure is as follows:

For a matrix \mathbf{A} , the scalar coefficients d_k in the characteristic polynomial

$$f(\lambda) = \sum_{k=0}^m d_k \lambda_k \quad (7.16)$$

and the matrix coefficients \mathbf{B}_k in

$$\mathbf{B}(\lambda) = \sum_{k=0}^{m-1} \mathbf{B}_k \lambda^{m-k} \quad (7.17)$$

are related by:

$$d_k = -1/k \text{ trace}[\mathbf{A}\mathbf{B}_{k-1}] \quad k = 1, 2, \dots, m \quad (7.18)$$

$$(d_0 = 1)$$

$$\mathbf{B}_k = \mathbf{A}\mathbf{B}_{k-1} + d_k \mathbf{I} \quad k = 1, 2, \dots, m \quad (7.19)$$

$$(\mathbf{B}_0 = \mathbf{I})$$

$$\mathbf{B}_m = \mathbf{0} \quad (\text{this can be used as a check})$$

where the operations and entries are over the real field and m is the dimension of the row (or column) space of \mathbf{A} .

Using this algorithm on the example matrix given earlier yields the following results:

$$\begin{array}{cccc} Z^3-0.5Z^2-0.448Z+0.89 & 0.7Z^2-0.28Z-0.2 & 0.315Z-0.095 & 0.142 \\ \mathcal{H}(Z) = Z\mathcal{H}_0 & 0.45Z^2-0.18Z-0.128 & Z^3-0.7Z^2-0.165Z+0.086 & 0.45Z^2-0.27Z+0.04 & 0.202Z-0.061 \\ 0.202Z-0.061 & 0.45Z^2-0.27Z+0.04 & Z^3-0.7Z^2-0.165Z+0.086 & 0.45Z^2-0.18Z-0.128 \\ 0.142 & 0.315Z-0.095 & 0.7Z^2-0.28Z-0.2 & Z^3-0.5Z^2-0.448Z+0.089 \end{array}$$

$$(Z - 1) (Z - 0.675) (Z + 0.15) (Z - 0.625)$$

Note that the numerator of $\mathcal{H}(Z)$ is usually a single row of the matrix since the initial state vector almost always has probability one for a single entry; the diagonal terms will be displayed here, however.

If the frequency response of the diagonal terms of $\mathcal{H}(Z)$ is computed, the spectra can be displayed as shown in Figure 60.

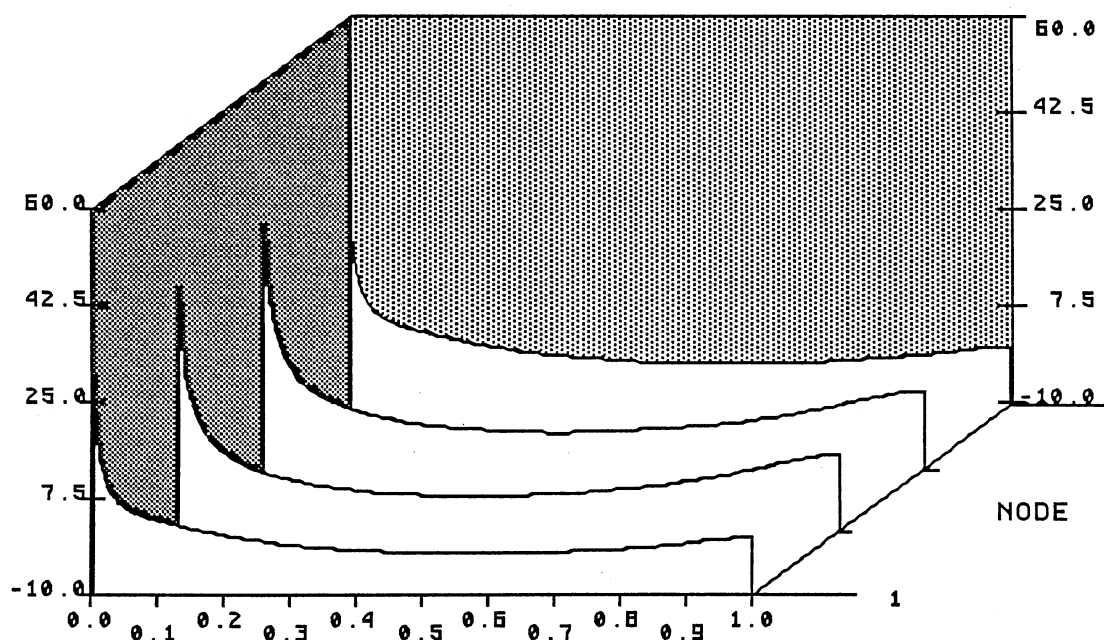


Figure 60: Frequency response of diagonal terms of $\mathcal{H}(z)$

The denominator is, in general, factorable for ergodic chains and one root will always be at $z=1$. Howard (1960, p. 10) states that the inverse transform of $\pi(Z)$ will be of the form:

$$H_n = \begin{bmatrix} \pi \\ \pi \\ \pi \\ \pi \end{bmatrix} + \sum_{i=1}^{m-1} \alpha_i^n T_i \quad n = 0, 1, \dots \quad (7.20)$$

where the first matrix on the right of the equal sign is comprised of row vectors of the limiting probability (each row sums to one). Note that this matrix is merely the residue from the pole at $Z=1$, which always exists for a stochastic matrix. The α_i term is the i -th eigenvalue of the characteristic equation (other than the eigenvalue at one) and the T_i term is a matrix of the partial fraction residues for the given eigenvalue. The rows of this matrix sum to zero. For a completely ergodic process, the expression above can be expressed as:

$$H_n = S + T_n' \quad (7.21)$$

where:

S is a stochastic matrix of limiting probabilities as defined above
and

T_n' is the sum of the transient matrices, also called differential matrices, which tend to zero as n becomes large.

Thus, the process can be viewed as a steady state response \mathbf{S} which is perturbed by a transient response \mathbf{T}_n' . The steady state response yields the zero frequency response in the spectrum while the transient terms cause the general spectral shape in Figure 60. This property is also discussed in Mullis and Steiglitz (1972).

What is the physical interpretation of the frequency and impulse responses (if any)? Huggins (1957) gives an example of a random telegraph signal and, using the spectral representation of the Markov matrix, derives the autocorrelation of the process. More research is needed to fully explore the relationship between the autocorrelation function, the Markov process spectrum and the transition matrix.

Historical Notes

The Markov chain was named after A. A. Markov, a Russian mathematician who wrote a series of papers around 1907 that outlined the basic theory (from Çinlar [1975], p. 143) and applied his theory to a study of the pattern of vowels and consonants in Russian literature, the most famous being an analysis of *Eugene Onegin* (Markov, 1913, cited in Çinlar). Later the idea of predicting the next event based on conditional probabilities and the current event was used by Shannon (1950) to predict missing letters in normal English text, particularly *Jefferson the Virginian* by Dumas Malone, although Shannon did not specifically refer to Markov's work. The theory of Markov chains has been used in a multitude of areas since that time (see Billingsley [1960] for a comprehensive bibliography); the theory of Markov processes is

important in the development of the Kalman filter, although the process must also be Gaussian (Brown, 1983, Ch. 5).

The derivation of the Z-transform and impulse response of the Markov chain given in this section differs from Howard (1960) because the operator Z was generally defined as a left shift in the late 1950's but is now defined as a right shift operator (for electrical engineers). Sittler (1956) did a significant amount of work in the area of systems analysis of Markov chains; his Ph.D dissertation at MIT was entitled "Analysis and Design of Simple Nonlinear Noise Filters" (Sittler, 1954). This work, while somewhat difficult to obtain, is quite readable and offers insight into a significant research area.

The Viterbi Algorithm

So far, this report has addressed some background issues in pattern/speech recognition and has addressed some topics in Markov processes. What has not been examined is how one would perform a sequential classification based on a Markov model. To motivate the discussion, consider the following: The entries in the matrix \mathbf{P} , $(\mathbf{P})_{ij}$, describe the probability of state transitions and in particular the probability of the process going to state j given that the process is currently in state i . If a set of observations is made that is assumed to be from the same process, it should be constrained to the same transition probabilities if the process is ergodic. To determine the likelihood that this observed process came from the original process, one could examine all the possible transitions at each node in the original chain and condition them with the transitions in the observations. The objective would then be to find the most likely "path" that the combined process could have taken around the flow graph. While this explanation is admittedly imprecise, it emphasizes that the goal of sequential detection is to see if a given set of observations are from a known process.

The Viterbi algorithm (VA) (Viterbi, 1967) is an efficient technique for calculating a likelihood measure; it is based on the dynamic programming algorithm pioneered by Bellman in the 1940's (see Bellman, 1957). The derivation given here is loosely taken from Forney (1973), which is a lucid and comprehensive review of the VA.

The VA is a general solution to the maximum *a posteriori* (MAP) problem; it estimates the most likely probability of a finite state, discrete time

Markov process observed in memoryless noise. This problem is "formally identical to finding the shortest route through a certain graph" (Forney, 1973). This graph is a special representation of the state transition matrix called a trellis. Figure 61 shows what the trellis would look like for the example we have been using (note the similarity to the DTW graph in Figure 54). "Each node corresponds to a state of the process at a given time and each branch represents a transition to some new state at the next instant of time" (Forney, 1973 - emphasis added). The trellis begins at state x_0 and ends at state x_k ; in the example, $x_0 = x_k$ but this is obviously not true in general. Branches are only drawn if the state transition matrix allows the process to go from one node to the other; in the example, the initial state (1) at $k = 0$ can only go to state 2 or stay in state 1 at time $k = 1$. Thus, the trellis branches describe every possible sequence for the process that begins at x_0 and ends at x_k . Given a sequence of observations \underline{Z} , each branch of the trellis can be assigned a "length" proportional to the probability of making that transition times the probability that the observation made a transition at that time, given that the observation was actually from the process. More formally, if \underline{X} is a vector from the known process where $\underline{X} = (x_0, \dots, x_k)$, then :

$$P(\underline{X}, \underline{Z}) = P(\underline{X}) P(\underline{Z} | \underline{X}) \quad (\text{joint distribution is separable if memoryless}). \quad (7.22)$$

Because the process is Markov, the total probability depends only on the prior state or transition; thus:

$$P(\underline{X}, \underline{Z}) = \prod_{k=0}^{K-1} P(X_{k+1} | X_k) \prod_{k=0}^{K-1} P(Z_k | X_{k+1}, X_k) \quad (7.23)$$

where K is the number of observations.

If we assign each branch (transition) the "length"

$$\lambda(\xi_k) = -\ln P(X_{k+1} | X_k) - \ln P(Z_k | \xi_k) \quad (7.24)$$

where $\xi_k = (X_{k+1}, X_k)$ (a transition), then the total path length is given by:

$$-\ln P(\underline{X}, \underline{Z}) = \sum_{k=0}^{K-1} \lambda(\xi_k) \quad (\text{from Forney [1973]}). \quad (7.25)$$

The first term in the branch length depends on the node probabilities of the underlying process while the second depends on transitions in the observed sequence \underline{Z} .

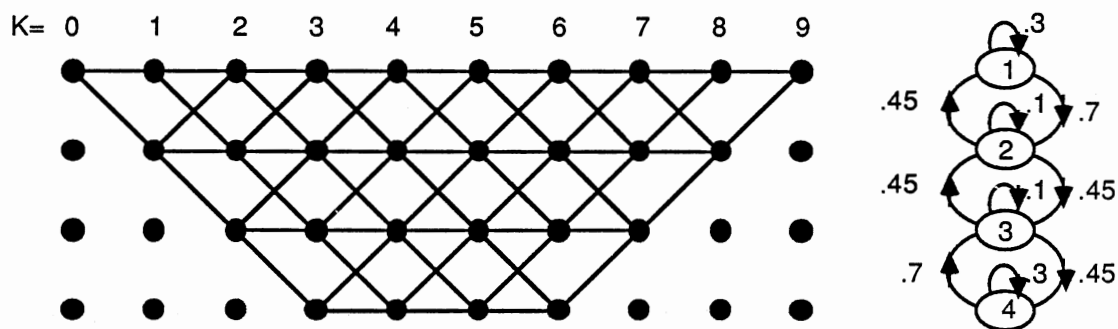


Figure 61: Trellis for Example

What is desired is the shortest path through the trellis since the maximum probability corresponds to the minimum of $-\ln P(\bullet)$. From dynamic programming theory, it can be shown that this distance through the trellis can be computed recursively; as the trellis is traversed, the minimum distance path

to a given node must lie on one of the possible final solution paths. In other words, if you have to go through Guthrie to get to Oklahoma City, the shortest path to Oklahoma City must have the shortest path to Guthrie as a segment. If one imagines a number of intermediate stops, the shortest path from Stillwater to Oklahoma City, given that these intermediate stops must be made, will be the sum of the minimum distances between those stops. For the trellis, the minimum distance to a given node will only depend on the accumulated distance to the previous node and the transition probabilities at the current node; no other information needs to be kept. A formal definition of the algorithm and an example will help illustrate these concepts.

VITERBI ALGORITHM

M: number of states in chain

k: time index

K: number of observations

$\Gamma(X_k)$: accumulated distance to node x_k , $1 \leq x_k \leq M$

$Z = (z_0, \dots, z_k)$: observation vector

Initial values

$k=0$

$\Gamma(x_0) = 0$

$\Gamma(i) = \infty$, $i \neq x_0$

Recursion

$$\Gamma(x_{k+1}, x_k) = \Gamma(x_k) + [-\ln P(x_{k+1} | x_k) - \ln P(z_k | \xi_k)] \quad (7.26)$$

(this must be computed for all possible transitions $x_k = (x_{k+1} | x_k)$)

$$\Gamma(x_{k+1}) = \min_{x_k} \{ \Gamma(x_{k+1}, x_k) \} \quad (7.27)$$

x_k

(for each x_k)

repeat until $k = K$.

The desired minimum distance solution is $\underline{\Gamma}(x_K)$. The path taken through the trellis can also be generated simultaneously but was omitted here.

As an example, consider the Markov matrix used previously:

$$\mathbf{P} = \begin{bmatrix} 0.3 & 0.7 & 0.0 & 0.0 \\ 0.45 & 0.1 & 0.45 & 0.0 \\ 0.0 & 0.45 & 0.1 & 0.45 \\ 0.0 & 0.0 & 0.7 & 0.3 \end{bmatrix}$$

and define

$$\mathbf{P}' = (\ln\{\mathbf{P}_{ij}\}) = \begin{bmatrix} 1.204 & 0.357 & \infty & \infty \\ 0.8 & 2.3 & 0.8 & \infty \\ \infty & 0.8 & 2.3 & 0.8 \\ \infty & \infty & 0.357 & 1.204 \end{bmatrix}$$

Given an observed sequence $\underline{Z} = [1 \ 2 \ 3 \ 4 \ 4 \ 3 \ 2 \ 2 \ 1]^T$, the maximum likelihood distance can be calculated as follows :

$$\underline{\Gamma}(x_0) = \begin{bmatrix} 0 \\ \infty \\ \infty \\ \infty \\ \infty \end{bmatrix} = \begin{bmatrix} \gamma_{01} \\ \gamma_{02} \\ \gamma_{03} \\ \gamma_{04} \end{bmatrix} \quad (\gamma_{01} = 0 \text{ because } z_0 = 1)$$

(γ_{ki} is the i -th node at time k)

For $k = 1$,

$$\begin{aligned} \gamma_{11} &= \min [P'_{j1} + P'_1 Z_1 Z_2 + \gamma_{0j}] \quad j = 1, 2, 3, 4 \\ &= \min [(P'_{11} \ P'_{21} \ P'_{31} \ P'_{41}) + P'_{12} + (0 \ \infty \ \infty \ \infty)] \\ &\quad (z_1=1, z_2=2) \\ &= \min [(P'_{11} \ \infty \ \infty \ \infty) + P'_{12}] = P'_{11} + P'_{12} \\ &= 1.204 + 0.357 = 1.56 \quad (\text{note that the path taken was from 1} \end{aligned}$$

to 1)

Continuing this process, the sequence of distances and trellis paths can be generated, yielding the final path and distance shown in Figure 62

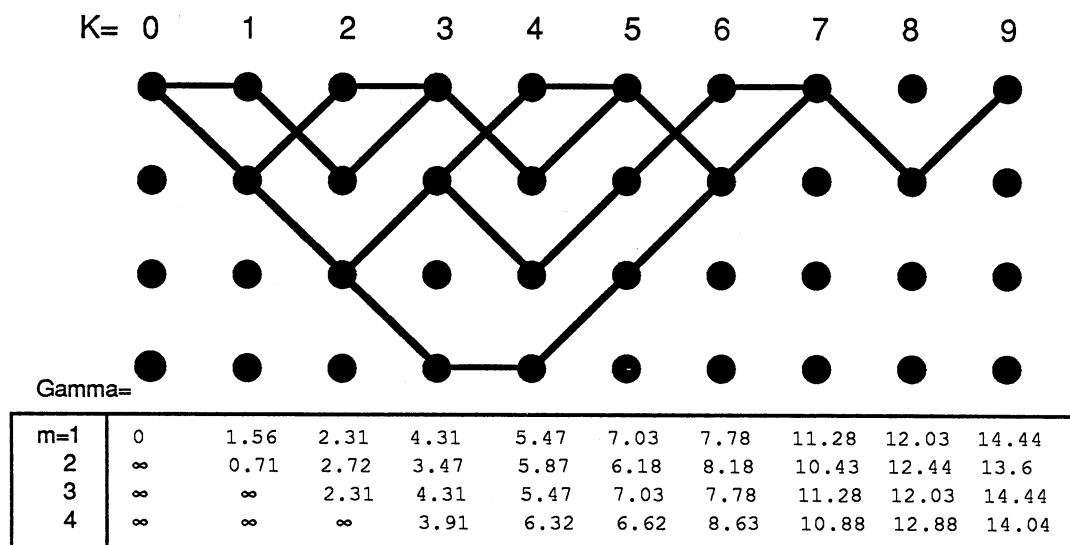


Figure 62: Solution path and distances for example

To see how this might be useful as a pattern recognition tool, consider changing a single entry in \underline{Z} ; for example, let $\underline{Z}=(1\ 2\ 3\ 4\ 4\ 3\ \underline{4}\ 2\ 1\ 1)^T$. Using the VA, the final distance will be infinity because a transition from state 4 to state 2 is not allowed in the transition matrix \mathbf{P} . This brings up an interesting point, however – since the transition matrix is "trained" from a known process of necessarily finite length, a value can be calculated to be zero when the underlying process may allow such a transition. This problem has been studied by at least two groups (Jelinek, 1976; Levinson *et al*, 1983) and the solution involves what are called reestimation formulas, generally implemented using the Baum-Welch or forward-backward (F-B) algorithm

(Baum *et al*, 1970) although Jelinek (1976) states that the VA can be used in some cases. Use of the reestimation algorithm is "beyond the scope of this thesis". An easy fix to the problem as suggested by Rabiner, *et al* (1983) is to compute an initial estimate of the transition probabilities using a training sequence and to change all zero entries of the resulting matrix to some $\epsilon > 0$. An added benefit of this approach is that it avoids computation of $\ln\{0\}$.

A pattern recognition system block diagram using Markov modeling and a Viterbi classifier is given in Figure 63; it incorporates most of the techniques described to this point, although the modified reestimation formulas have not been implemented. It is this general approach that will be used in the following section to demonstrate a preliminary recognition system that uses the VA for maximum *a posteriori* classification.

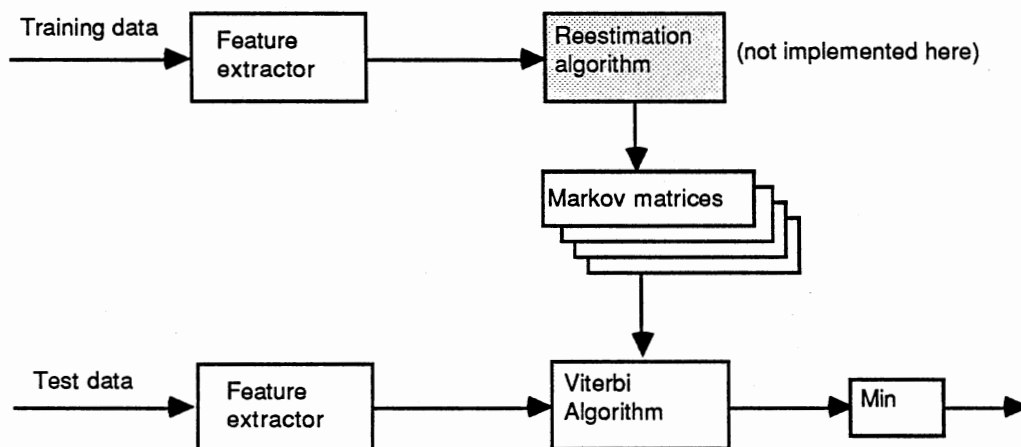


Figure 63: Recognition System Block Diagram

Use of the Viterbi algorithm on functions of Markov chains

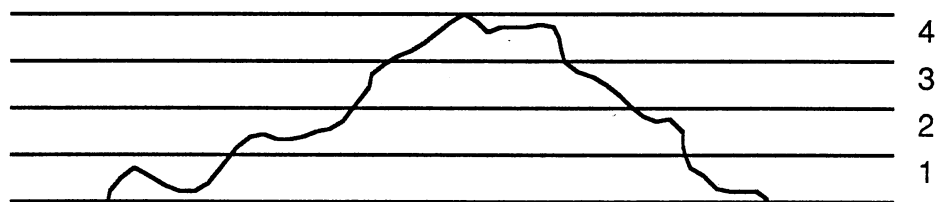
Given a combination of Markov modeling and sequential classification, one final issue must be resolved before recognition studies can be attempted: feature extraction. While the bulk of the existing literature discusses use of LPC spectral models, other features could certainly be used. Three feature extraction strategies were examined for this thesis: time derived, reflection coefficient derived and log spectral model derived.

Most researchers using Markov models constrain the state transitions so that the process can never return to a state once it leaves; the process is further constrained to only stay in the current state, move ahead one state, or move ahead two states. This is called a left to right or Bakis model (cited by Jelinek, 1976) and results in a Markov matrix that has entries only along the diagonal and the first two superdiagonals. The author has been unable to determine the theoretical justification for such a highly constrained model, although Jelinek (1975) shows that the constraints make the classification easier. No restrictions will be made on the state transition matrix in this study, however.

Time derived features

This technique is a significant departure from the spectral models; it represents an attempt to statistically measure time variations of the signal. The algorithm takes a segment of the waveform of a specified length (with a

rectangular window) and calculates the maximum and minimum value over that interval. This amplitude range is divided into n intervals, where n is the desired number of states in the Markov chain. The waveform segment is then analyzed point by point and the amplitude state corresponding to the sample value is calculated. The current state and the previous state are treated as an ordered pair that is hashed into an $n \times n$ matrix - the current state is the row index and the previous state is the column index. The value of that location in the matrix is incremented by one and the routine moves to the next point (see Figure 64a). When the entire segment has been analyzed in this manner, the rows of the matrix are normalized so that their sum is one, making it a stochastic matrix that is an estimate of the transition probabilities. Figure 64b) shows how the time structure of a waveform affects the structure of the Markov matrix for this method. A series of Markov matrices will be stored for each word, where the analysis interval will be based on the assumption of the short term stationarity of the speech signal (approximately 15ms will be used here).



Requantize the signal into n levels, then record the number of times the signal moves to a new state or stays in the same state.

Figure 64a: Time based Feature Extraction

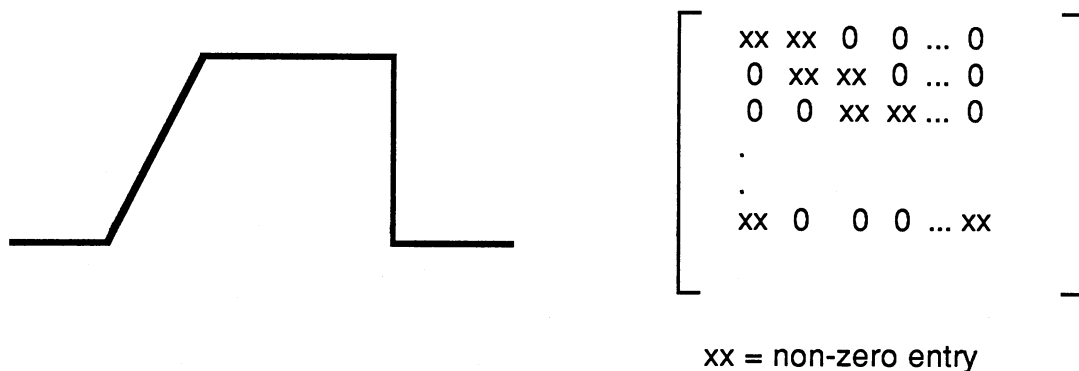


Figure 64b:Waveform Encoding Effects

Recognition is performed by encoding an unknown waveform segment into a state sequence using the same analysis interval . The state sequence and the Markov matrices derived from the training data are passed to the Viterbi algorithm, which computes the minimum path length for each reference matrix. The smallest of these path lengths is chosen as the solution distance and the class corresponding to that reference matrix is chosen as the most likely class.

Before proceeding, it should be stressed that this technique will work on time data in any form - not just the "raw" signal. One technique that dates back at least as far as 1967 is counting the number of zero crossings over some analysis interval (Reddy, 1967). This is equivalent to using a frequency discriminator on the signal which, in the case of speech signals, can be used to determine if the segment is voiced or unvoiced (Rabiner and Schafer, 1978, p. 128). For the test cases in this thesis, the Markov model/VA will be used with zero crossing information as a feature.

Reflection coefficient derived features

As mentioned previously, the most recent studies of Markov models for speech recognition use LPC spectral models as a basis. An equally valid way to encode the LPC model is to use the reflection coefficients (RC) that are naturally computed along with the LPC parameters in Levinson's recursion (Markel and Gray, 1976, p. 51). The advantage of using reflection coefficients is that they are always in the range $[-1,1]$, which would intuitively seem to make encoding easier.

The LPC analysis is followed by a vector quantizer to reduce the dimensionality of the LPC vector (Bell Labs study [Rabiner *et al*, 1983] based on the work of Portiz [1982]). Vector quantization (VQ) is optimum in the sense that the distance between the output vectors is maximized under a transformation on the input vectors. The VQ technique is powerful but requires a great deal of computation during training and produces transformation equations that are data dependent (Juang *et al*, 1982). It would seem that a choice should be considered between an optimal but data dependent technique and a suboptimal but data independent alternative. One such suboptimal approach, which will be used here, takes a vector of reflection coefficients derived from an LPC algorithm of order m and encodes it into an m bit word that represents the current state of the process (see Figure 65). This encoding is performed by observing the sign of the k -th coefficient ($1 \leq k \leq m$); if it is positive, 2^{k-1} is added to a summing variable, otherwise nothing is added. This encoded value will then be in the range $[0, 2^{k-1}]$, which will be subdivided into n levels, where n is the number of states in the chain model,

and the matrix will be constructed as before. Likewise, the classifier operates by building an observation sequence from the unknown and then determines the class it was most likely to have come from using the VA. An important difference between the two techniques is that fewer samples are available for training using this approach; an analysis frame of 128 samples would generate 128 observations for use in building the time based model but only one value for the LPC and spectral techniques. A half second utterance would only generate 31 observations (for 8kHz sampling frequency and 128 sample frame size), making confident parameter estimation difficult.

Given the reflection coefficient vector RC:

RC = (rc₁ rc₂ rc₃, rc_m) -1 ≤ rc_i ≤ 1

The feature is calculated as follows:

SUM = 0

DO i = 1 to i_max

IF rc_i > 0 THEN SUM = SUM + 2(i-1)**

END DO

The quantity SUM will be the feature value.

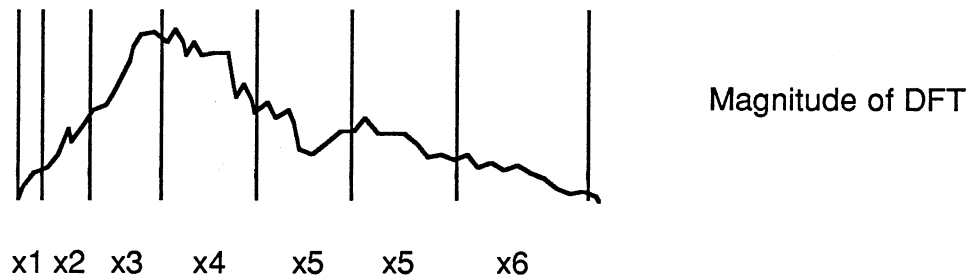
Figure 65: Reflection coefficient feature extraction

Spectral model derived features

As mentioned earlier, there is merit in studying cochlear models for use in Markov chain models; to the author's knowledge, no results have been published in the literature in this area, although Zue (1985) hints that the area bears investigation. Unfortunately, "state of the art" cochlear modeling techniques are rather complex, as noted previously. For this thesis, a first step in that direction will be made by using a log amplitude – log spaced spectral

estimate; while the technique has some merit (Zue, 1985), it is not a particularly accurate model of the evoked response of the cochlea but it has been widely used, nevertheless. The technique is to perform a discrete Fourier transform over an analysis interval and generate a log frequency spacing of m total cells by summing bins together before taking the log of the spectral magnitude. The spacing of the summing interval will grow wider at higher frequencies, giving decreased resolution at those frequencies. This will yield a log amplitude – log frequency vector that must be encoded to yield an m bit number as with the other feature extraction techniques. The elegant approach, as with the reflection coefficient technique, would be to use vector quantization; instead we shall use an encoding strategy based on the average value across the vector. A feature value is generated by adding 2^{k-1} to a sum if the k -th component ($1 \leq k \leq m$) of the vector is larger than the average value. This sum can be scaled into one of the n levels for generation of the observation sequence (see Figure 66), yielding either a Markov matrix (training) or an input sequence to the VA (classification).

This technique, like the RC algorithm given previously, is amplitude independent, which has its merits as well as its limitations - more work is needed to determine the suitability of these techniques as encoding schemes.



Let μ = average (x_i)

```
SUM=0
DO i = 1 to i_max
  IF  $x_i > \mu$  THEN SUM = SUM + 2**(i-1)
END DO
```

Figure 66: Spectral Model Feature Extraction

Summary of computer tools

Two speech sets were digitized for analysis and testing. A number of computer programs were written that create, display, analyze and classify Markov models derived from these data; they are:

MKVXTR Generates a Markov matrix based on reflection coefficient, spectral model or block zero crossing (not discussed) features. One feature generated for each analysis frame.

MKVTIM Generates time signal based Markov model. Signal may be preprocessed to allow variations on the time based technique. One matrix

is generated for each analysis frame

MKVPRT Prints Markov model matrices.

MKVPLT Plots signal flow graph of Markov matrix.

MKVIMP Prints impulse response of Markov matrix.

MKVXFN Plots frequency response of Markov process transfer function (diagonal terms only).

MKVXFP Prints transfer function values.

MKVATM Time signal classifier. Uses Viterbi algorithm to test a data file against a set of reference matrices.

MKVCLS Block feature classifier (RC, log-log spectral model, etc.).

DATA

ONES.DAT The word "one" spoken ten times by the same speaker

DIGITS.DAT The digits "one" through "ten" spoken by the same speaker

Results

Rather than inundate the reader with output, we will attempt to summarize three test cases here, leaving the mound of plots and tables for the appendices. The three test cases are actually variations of the same general problem - digit recognition. The first test case will use time derived features based on a zero crossing representation of the speech signal; the second will use reflection coefficient features; and the third will use the log-log spectral model. In the first case, the classifier will be trained on the word "five" and tested against the words "one" through "ten". In the second case, the classifier will be trained on the words "one", "two", and "three" and tested against all ten

words. The third case will be the same as the second except for the change of feature. While none of these tests is rigorous - testing a classifier against its training set only indicates gross robustness of the feature set - the tests do illustrate the relative merits of the approaches.

Test case 1 - Time derived features

For this test, the word "five" was analyzed after preprocessing by a zero crossing discriminator of width 32. The speech signal and the zero crossing signal for this word are shown in Figure 67 (as noted earlier, a rectangular window was used). If a frame size of 128 samples is chosen, the word is approximately 32 frames in length; for the time derived features, this results in 32 Markov matrices. After some experimentation, a chain length of 8 was chosen for the model. Of the 32 matrices, the fifth was chosen (arbitrarily) for display. The values in the matrix are shown in Figure 68 while the signal flow graph, which displays the same information graphically, is given in Figure 69. From this diagram, states three and six are never visited at all, because when the signal is requantized to eight levels, those amplitude values do not occur. In Figure 70, these empty states translate into a zero level for the transfer function of those nodes. The other states show relatively uninteresting transfer functions, although the limiting probability of states one and two appear to be less than that of states four, five, and eight; from MKVIMP, these limiting values are: $\pi = (0.01 \ 0.033 \ 0.0 \ 0.243 \ 0.291 \ 0.0 \ 0.265 \ 0.159)$.

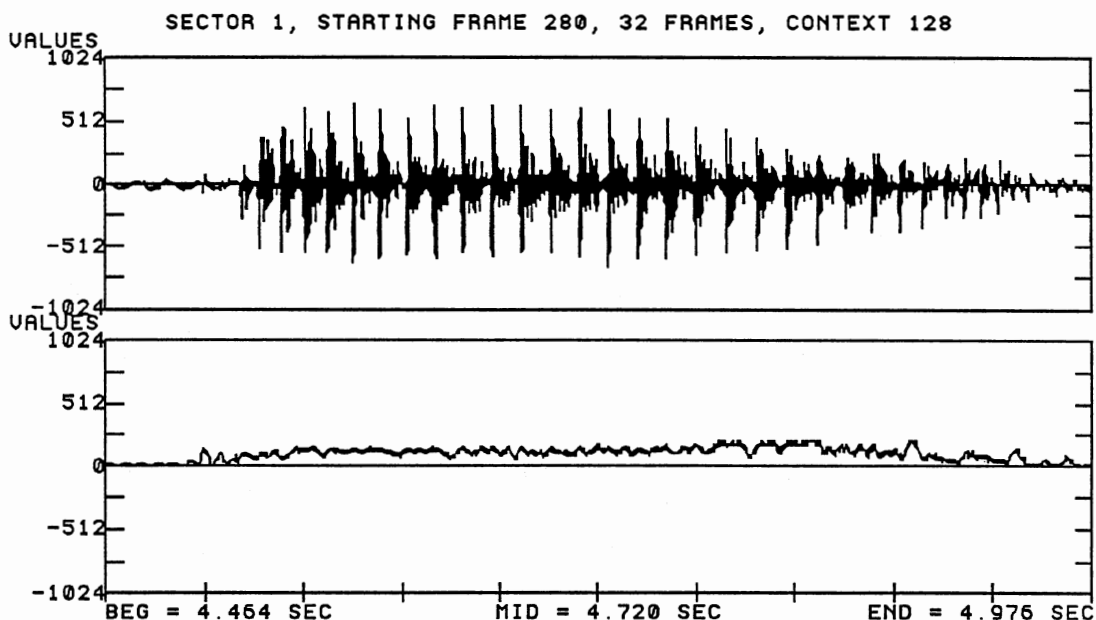


Figure 67: Time and Zero Crossing Signals – "FIVE"

MARKOV TRANSITION MATRIX**FILE NUMBER: 3022****NUMBER OF STATES IN CHAIN: 8****STARTING FRAME, NUMBER OF FRAMES USED: 280, 32****FRAME SIZE: 128 DATA SOURCE: TIME****FRAME: 284**

1	0.66667	0.33333	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.10000	0.70000	0.00000	0.20000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.00000	0.02703	0.00000	0.81081	0.16216	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.13514	0.72973	0.00000	0.13514	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000	0.00000	0.14815	0.00000	0.66667	0.18519
8	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.30769	0.69231

Figure 68: Markov Transition Matrix for "FIVE"

MARKOV CONNECTIVITY PLOT
 NUMBER OF STATES: 8
 FRAME SIZE: 128
 DATA SOURCE: TIME (0)

FILE NUMBER: 3022
 START, NUM FRAMES: 280, 32
 THRESHOLD: 0.000000
 FRAME: 5

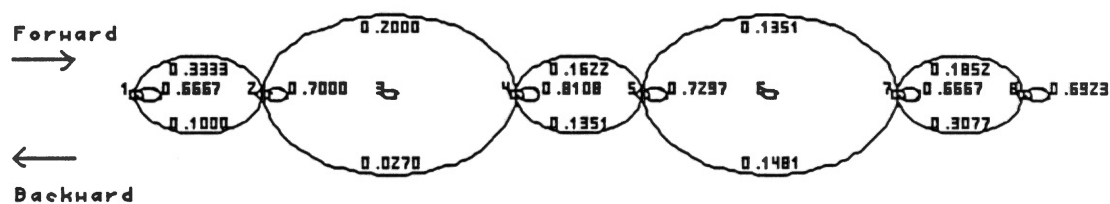


Figure 69: Signal flow Graph – “FIVE” (zero crossing)

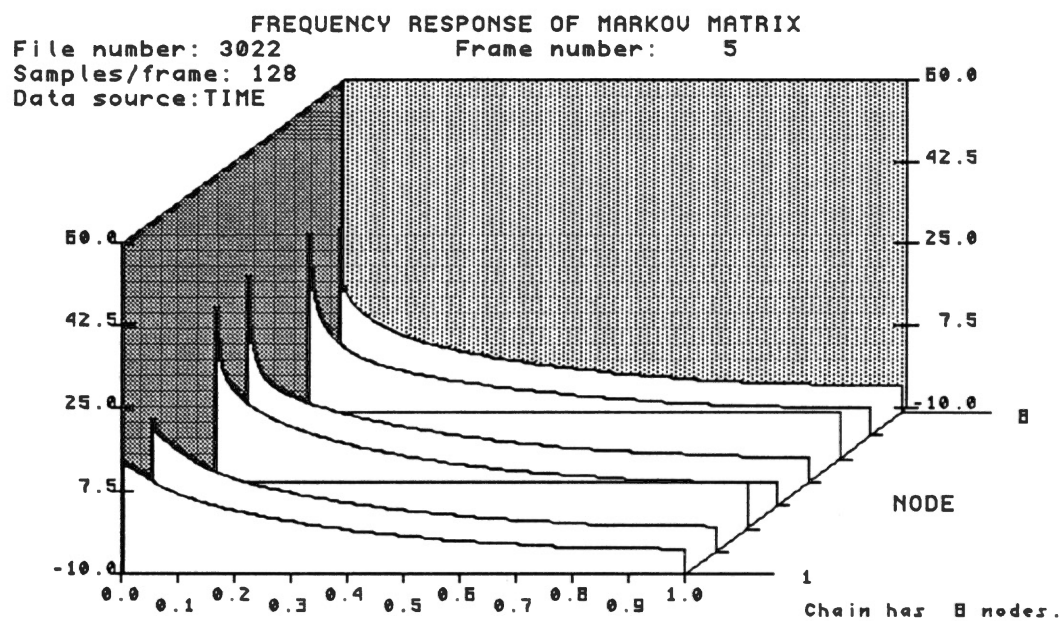


Figure 70: Frequency Response – “FIVE” (zero crossing)

Using the time classifier, the 32 "phonetic units" (frames) of the word "one" were tested against the entire set of the digits "one" through "ten" (622 frames at 128 samples/frame). Correct recognition of the digit should be indicated by a match with successively higher reference matrices; in other words, the classifier should count up through the reference sets as the correct word is analyzed. Words not corresponding to the reference set should show no such pattern (technically, this should be considered word spotting rather than digit recognition). Instead of printing 622 distance scores, we shall instead plot the class chosen by the classifier as a function of input frame number, which should indicate linear trends in the results. Figure 71 shows this recognition sequence, indicating that the algorithm correctly identified its training sequence, except for a very few frames, and rejected the other digits. This indicates that the algorithm has promise but the reader is cautioned that this test is not rigorous and further testing is needed.

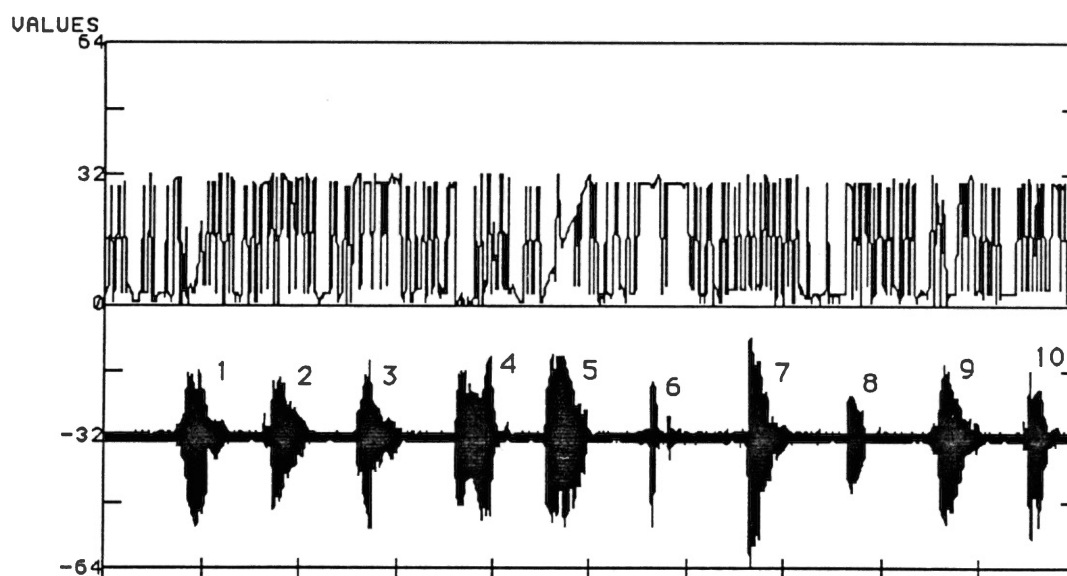


Figure 71: Recognition Sequence for Example

Test case 2 - Reflection coefficient features

As mentioned before, this test will train on the words "one", "two", and "three" from the set of digits, then test against against the same set, using the reflection coefficient method described previously. As an example of the difference in the structure of the matrices, consider Figures 72 and 73 which show the flow graph and transfer function, respectively, of the word "one". For these frame based features (few matrices instead of many), the plot of recognizer performance is not as useful; instead, these preliminary results can be summarized by a confusion matrix to show how well the set of reference matrices scored against the training set, which is given in Figure 74. Obviously, this is not a rigorous test, but it does indicate the consistency of the procedure since no classification procedure can work if it cannot recognize its own reference set.

MARKOV CONNECTIVITY PLOT
 NUMBER OF STATES: 5
 FRAME SIZE: 128
 DATA SOURCE: REFL (10)

FILE NUMBER: 3024
 START, NUM FRAMES: 1, 3
 THRESHOLD: 0.000000
 FRAME: 1

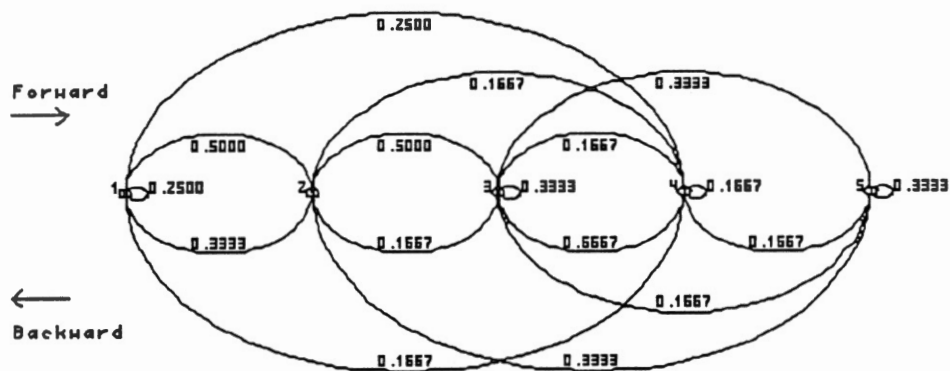


Figure 72: Signal Flow Graph – “ONE” (Reflection coefficient)

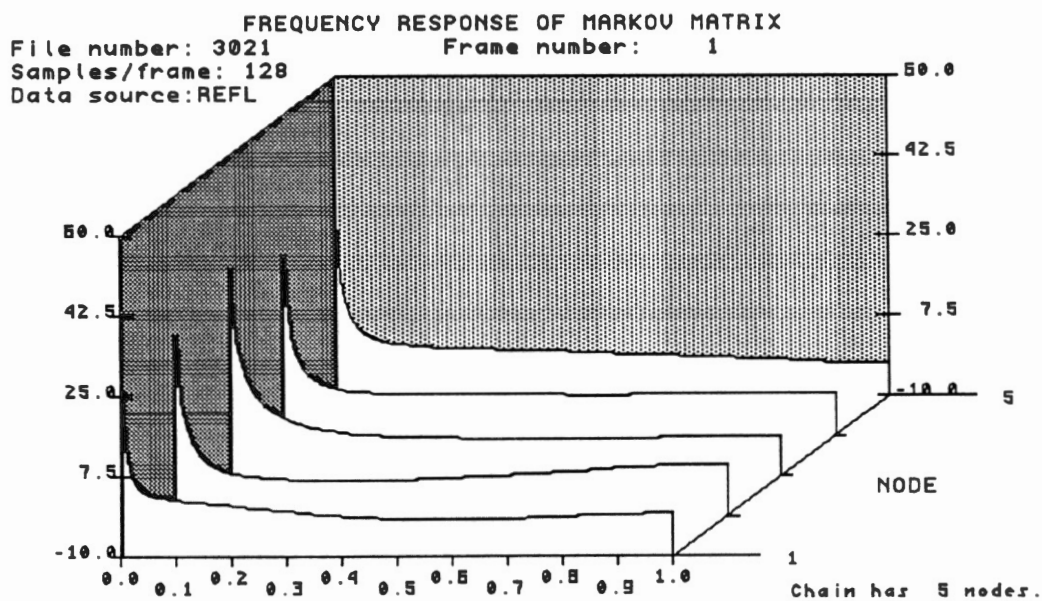


Figure 73: Frequency Response – “ONE” (Reflection coefficient)

CONFUSION MATRIX (# OCCURRENCES)				
TEST \ REFERENCE	1	2	3	#obs
	1	2	3	
1	23			35
2		7		35
3			12	30
4				28
5				35
6				15
7				35
8				15
9				30
10				25

Size of AR model: 10
 Size of Markov matrix: 5
 Trellis length: 20
 Frame size: 128 samples at 8kHz

Figure 74: Confusion Matrix – “ONE” (Reflection coefficient)

Test case 3 - log-log spectral model features

This test is identical to the previous except the log amplitude - log frequency spectral model (a crude cochlear model) is used. The flow graph is shown in Figure 75 and the transfer function is shown in Figure 76. Classifier performance was evaluated in the same fashion as before by testing the reference set against its own training data. In this case, given in Figure 77, performance was not as good – it is hypothesized that an improved cochlear model and better probability estimation techniques would improve these results.

MARKOV CONNECTIVITY PLOT
 NUMBER OF STATES: 5
 FRAME SIZE: 128
 DATA SOURCE: SPEC (16)

FILE NUMBER: 3023
 START, NUM FRAMES: 1, 3
 THRESHOLD: 0.000000
 FRAME: 1

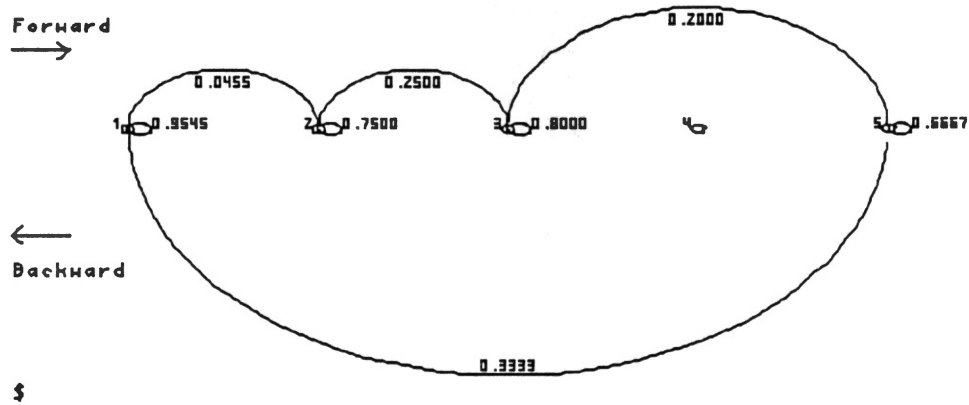


Figure 75: Signal Flow Graph – “ONE” (Spectral Model)

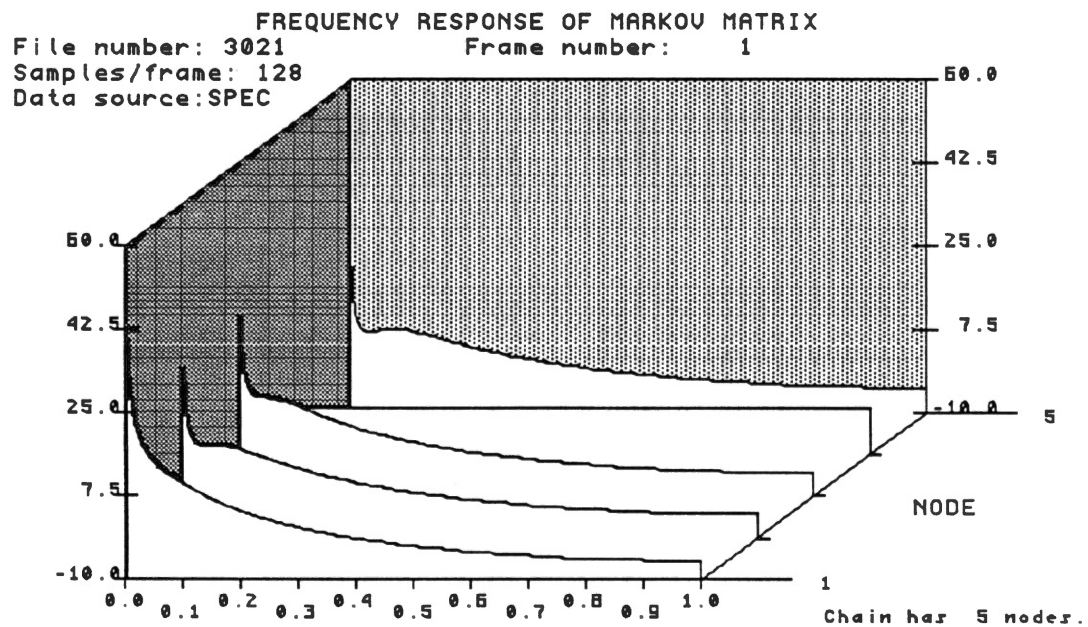


Figure 76: Frequency Response – “ONE” (Spectral Model)

CONFUSION MATRIX (# OCCURRENCES)				
TEST	REFERENCE			#obs
	1	2	3	
1	27	4		35
2		27		35
3	3	8	17	30
4		2		28
5		12		35
6				15
7		12		35
8				15
9		3		30
10		17		25

Size of spectral model: 16
 Size of Markov matrix: 5
 Trellis length: 30
 Frame size: 128 samples at 8kHz

Figure 75: Confusion Matrix – “ONE” (Spectral Model)

CHAPTER VIII

CONCLUSIONS AND AREAS FOR FUTURE RESEARCH

This thesis has introduced a number of new and significant ideas in the areas of speech coding and recognition. In particular, new methods for linear predictive (autoregressive) coding of speech using a p -normed error criterion (L_p model) have been developed, as well as some experiments in speech recognition using Markov chains.

One of the significant results in this thesis is an optimal, adaptive scheme for coding speech using an L_p model. The kurtosis, which is a measure of the "flatness" of a distribution, of the residual from a traditional autoregressive model is calculated for each frame of speech. This kurtosis is then used to map to the best p value; this optimal value of p is then used to compute the L_p model using the residual steepest descent algorithm. Here, the best value of p is determined by using the fact that the L_p model is the maximum likelihood estimator for the corresponding p -Gaussian distribution, a general class of pdf's. Specifically, a kurtosis measurement of six requires an L_1 model (Laplacian residuals), a kurtosis of three requires an L_2 model (Gaussian residuals) and a kurtosis below three requires a higher value of p – as the kurtosis nears 1.8, p is required to go to infinity, the limiting value for a uniform distribution (L_∞).

This optimal, adaptive scheme has been incorporated into the government standard vocoder algorithm, LPC-10. These modifications have a minimal impact on LPC-10 complexity but increase computation time by about

a factor of 3, although no efforts have been made to fine-tune the speed of the L_p algorithms.

Even though L_p models are more likely to be unstable than the L_2 model, no remedial measures were taken in LPC-10 other than the existing procedure of taking the parameters from the previous frame and using them again.

Another method has been historically to calculate the L_p model for $p=1$ and $p \rightarrow \infty$: linear programming. While linear programming gives an exact answer, it has traditionally been too slow and consumed too much memory to be useful in practical problems, especially when high-speed performance is needed. The simplex method requires calculations that grow exponentially with the number of parameters while the technique used in this thesis, RSD, is only quadratic in the parameters. Newer implementations of linear programming have narrowed the gap but the RSD procedure still has the advantage of being able to calculate the more general L_p model.

Another significant result in this thesis is the L_p Burg algorithm; it has all the traditional advantages of the L_2 Burg method – stability, direct calculation of the reflection coefficients, etc. It too, can be used to generate models for speech synthesis. In addition, the L_p Burg method, which is based on the IRLS algorithm, will converge to a stable model for any p between one and three, although the L_1 model may not be unique.

Since the spectrum of speech is of great importance, the spectrum of the L_p model was evaluated for both the covariance method and the newer Burg method. These algorithms were evaluated both on “clean” and noisy speech. The L_1 estimator appears to preserve formants quite well; other values of p also seem to offer no great surprises. At fairly modest signal-to-noise ratios

(approximately 20dB), the L_p estimator, especially for p near one, seemed to have problems retaining the formant structure of speech for white noise. The robustness of the L_p model for p near one should make it ideal for coding in environments that have a great deal of noise, especially impulsive noise.

Finally, some experimental results were gathered using Markov models as an isolated word recognition technique. The uniqueness of the approach was in that the models were not the so-called "hidden" Markov models but were explicit models that were derived directly from parametric data. Of particular promise is the use of a cochlear model for the pre-processing function of the model extraction process.

There are a number of research areas that remain to be explored, such as:

- How does L_p speech perform in listening tests? Would the diagnostic rhyme test (DRT) show improvements in intelligibility?
- Are there other ways to constrain the L_p model to be stable without use of the Burg algorithm? Will reflection of unstable poles to inside the unit circle harm intelligibility?
- What desirable properties of the covariance or autocorrelation L_p solution are lost by using the Burg algorithm? In other words, what does the guarantee of stability cost?
- Can procedures be used to speed up RSD to make it a practical alternative to Cholesky's method in real-time applications such as LPC-10?
- In detection theory, can these results be applied to develop an optimum detector for signals in non-Gaussian noise?
- Can the L_p Burg algorithm be expanded to two dimensions?

- Why does the Burg algorithm, as implemented here, appear to have problems when using values of p other than two in noisy signals? Is it inherent in the algorithm or can remedial measures be taken.
- Would shaping of the noise or glottal pulses at the synthesizer to match the appropriate p -Gaussian density improve intelligibility?
- Can L_p models be applied to other non-Gaussian signals, such as neural firings?
- Are there alternatives to dynamic programming for classification of Markov processes?
- Is there a relationship between Markov spectral models and maximization techniques such as dynamic programming?

The traditional view has always been that Gaussian noise is a “good enough” first order approximation because better approximations made the analysis untenable. The p -Gaussian family of pdf's and their maximum likelihood estimators, the p -normed solutions to the linear prediction equations, offer a powerful extension to existing theory that should find application wherever non-Gaussian noise is found, which is, of course, everywhere.

CITED REFERENCES

- Allen, Jont B., "Cochlear Modeling", *ASSP Magazine*, Jan 1985, pp. 3-29.
- Asphall, Bengt, and Richard Stone, "Khachiyan's Linear Programming Algorithm", Stanford University Report STAN-CS-79-776, 1976
- Atal, B. S., and Suzanne L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", *Jou. Acoust. Soc. Am.*, **50**, 1971, pp. 637-655.
- Baker, James K., "The DRAGON System - An Overview", *IEEE Transactions on Acoustics , Speech and Signal Processing*, Feb 1975, pp. 24-29.
- Barrodale, I., and F.D.K. Roberts, "An Improved Algorithm for Discrete L_1 Linear Approximation", *SIAM J. Numer. Anal.*, V. 10 No. 5 (1973), pp. 839-848.
- Barrodale, I., and F.D.K. Roberts, "Solution of an Overdetermined System of Equations in the L_1 Norm", *Comm ACM*, V. 17 No. 6, June 1974, pp. 319-320.
- Baum, Leonard E., Ted Petrie, George Soules, and Norman Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabalistic Functions of Markov Chains", *Annals of Mathematical Statistics*, v41 No. 1, 1970, pp. 164-171.
- Bednar, J.Bee, Rao Yarlagadda and Terry L. Watt, " L_1 Deconvolution and Its Application to Seismic Signal Processing", *IEEE Trans. ASSP*, ASSP-34, Dec 1986, pp. 1655-1658.
- Bellman, Richard, *Dynamic Programming*, Princeton University Press

- (Princeton NJ), 1957.
- Billingsley, Patrick, "Statistical Methods in Markov Chains", *Ann. Math. Stat.*, v32, Mar 1961, pp. 12-40.
- Brogan, William L., *Modern Control Theory*, Quantum Publishers (New York), 1974.
- Brown, Robert G., *Introduction to Random Signals and Kalman Filtering*, Wiley and Sons (New York), 1983.
- Burg, J., "A New Analysis Technique for Time Series Data", *Proc. NATO Advanced Study Institute on Signal Proc.*, Enschede, Netherlands, 1968.
- Byrd, Richard H., and David A. Pyne, "Convergence of the Iteratively Reweighted Least Squares Algorithm for Robust Regression", The Johns Hopkins Univ., Baltimore, MD, Technical Report N0. 313, June 1979.
- Çinlar, Erhan, *Introduction to Stochastic Processes*, Prentice-Hall (Englewood Cliffs NJ), 1975.
- Croxtan, Frederick E., *Elementary Statistics*, Dover Press (New York), 1953.
- Denöel, Etienne and Jean-Phillipe Solvay, "Linear Prediction of Speech with a Least Absolute Error Criterion", *IEEE Trans. ASSP*, ASSP-33, Dec 1985, pp. 1397-1403.
- Duda, R.O., and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley and Sons (New York), 1973.
- Dwyer, Roger F., "A Statistical Frequency Domain Signal Processing Method", *Statistical Signal Processing*, Marcel Dekker (New York), 1984, pp. 79-90.
- Flanagan, James L., "Voices of Men and Machines", *Jou. Acoust. Soc. Am.*, V

51, No 5 (Part 1), 1972, pp. 1375–1387.

Flanagan, James L., "Computers That Talk and Listen: Man - Machine Communication by Voice", *Proceedings of the IEEE*, April 1976, pp. 405-415.

Flanagan, James L., S.E. Levinson, L.R. Rabiner, and A.E. Rosenberg, "Techniques for Expanding the Capabilities of Practical Speech Recognizers", *Trends in Speech Recognition*, Wayne A. Lea, ed., 1980, pp. 425-444.

Forney, G. David, "The Viterbi Algorithm", *Proc. IEEE*, Mar 1973, pp. 268-278.

Frame, J.S., "Matrix Functions and Applications", *IEEE Spectrum*, Mar-July 1964.

Geary, R.C. "Testing for Normality", *Biometrika*, Vol. 34(1947), pp. 209-242.

Graybill, Franklin A., *Theory and Application of the Linear Model*, Wadsworth (Belmont, CA), 1976.

Hamming, R. W., *Numerical Methods for Scientists and Engineers*, 2nd ed, McGraw-Hill (New York), 1973.

Hayes, J.F., "The Viterbi Algorithm Applied to Digital Data Transmission", *IEEE Communication Society Magazine*, Mar 1975, pp. 15-20.

Hershey, John, and Rao Yarlagadda, *Data Transportation and Protection*, Plenum Press (New York), 1986.

Hogg, Robert V., "An Introduction to Robust Procedures", *Comm. Stat - Thry and Meth.*, A6(9), 1977, pp. 789-794.

Howard, Ronald A., *Dynamic Programming and Markov Processes*, Wiley and Sons (New York), 1960.

Huggins, William H., "Signal Flow Graphs and Random Signals", *Proceedings of the IRE*, 45, 1957, pp. 74-86.

- Itakura, Fumitada, and Shuzo Saito, "A Statistical Method for Estimation of Speech Spectral Density and Formant Frequencies", *Electron. Commun. Japan*, **53-A**, 1970, pp. 36–43.
- Itakura, Fumitada, and Shuzo Saito, "On the Optimum Quantization of Feature Parameters in the PARCOR Speech Synthesizer", *Proc. Conf. Speech Commun. Process*, 1972, pp. 434–437.
- Itakura, Fumitada, "Minimum Prediction Residual Principle Applied to Speech Recognition", *IEEE Trans. ASSP*, Feb 1975, pp. 67-72.
- Jayant, N.S., and Peter Noll, *Digital Coding of Waveforms*, Prentice-Hall (Englewood Cliffs, NJ), 1984.
- Jelinek, Frederick, "Fast Sequential Decoding Algorithm Using a Stack", *IBM Journal of Research and Development*, Nov 1969, pp. 675-685.
- Jelinek, Frederick, L.R. Bahl, and R.L. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech", *IEEE Trans. Information Theory*, May 1975, pp. 250-256.
- Jelinek, Frederick, "Continuous Speech Recognition by Statistical Methods", *Proc IEEE*, April 1976, pp. 532-556.
- Jeter, Melvyn W., *Mathematical Programming*, Marcel Dekker (New York), 1986.
- Juang, Biing-Huang, David Y. Wong, and A.H. Gray, Jr., "Distortion Performance of Vector Quantization for LPC Voice Coding", *IEEE Trans. ASSP*, April 1982, pp. 294-304.
- Karmakar, Nerendra K., "A New Polynomial Time Algorithm for Linear Programming", *Combinatorica*, Vol. 4 (1984), pp. 373–395.
- Kazakos, Dimitri, "The Bhattacharyya Distance and Detection Between Markov Chains", *IEEE Trans. Info. Theory*, Nov 1978, pp. 747-754.

- Kazakos, Dimitri, "Statistical Discrimination Using Inaccurate Models", *IEEE Trans. Info. Theory*, Sep 1982, pp. 720-728.
- Kemeny, John G. and J. Laurie Snell, *Finite Markov Chains*, Van Nostrand (Princeton NJ), 1960.
- Klatt, Dennis H., "Review of the ARPA Speech Understanding Project", *Journal of the Acoustic Society of America*, Dec 1977, pp. 1345-1366.
- Kaneko, Toyohisa and N. Rex Dixon, "A Hierchical Decision Approach to Large Vocabulary Discrete Utterance Recognition", *IEEE Trans. ASSP*, Oct 1983, pp. 1061-1066.
- Levinson, S.E., L.R. Rabiner and M.M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition", *Bell System Technical Journal*, April 1983, pp. 1035-1074.
- Levinson, S.E., "Structural Methods in Automatic Speech Recognition", *Proc. IEEE*, Nov 1985, pp. 1625-1650.
- Lyon, Richard F., "A Computational Model of Filtering, Detection, and Compression in the Cochlea", *Proc. Intl. Conf. ASSP*, May 1982, pp. 1282-1285.
- Machell, Frederick W. , and Clark S. Penrod, "Probability Density Functions of Ocean Acoustic Noise Processes", *Statistical Signal Processing*, Marcel Dekker (New York), 1984, pp. 211-221.
- Makhoul, J., "Stable and Efficient Lattice Methods of Linear Prediction", *IEEE Trans. ASSP*, ASSP-25, Oct 1977, pp. 423-428.
- Markel, J.D. and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag (Berlin), 1976.
- Marple, S. Lawrence, *Digital Spectral Analysis with Applications*, Prentice-

- Hall (Englewood Cliffs, NJ), 1987.
- McCormick, G.F., and V.A. Sposito, "Using the L_2 Estimator in L_1 Estimation", *SIAM J. Numer. Anal.*, V. 13 No. 3(1976), pp. 337-343.
- Mendel, Jerry M., *Lessons in Digital Estimation Theory*, Prentice-Hall (Englewood Cliffs, NJ), 1987.
- Miller, James H., and John B. Thomas, "Detectors for Discrete-Time Signals in Non-Gaussian Noise", *IEEE Trans. Info. Thry.*, IT-18, March 1972, pp. 241-250.
- Mullis, Clifford T. and Kenneth Steiglitz, "Circulant Markov Chains as Digital Signal Sources", *IEEE Trans. on Audio and Electroacoustics*, Oct 1972, pp. 246-248.
- Olkin, I., L.J. Gleser and C. Derman, *Probability Models and Applications*, MacMillan (New York), 1980.
- Paez, M.D., and T.H. Glisson, "Minimum Mean-Squared-Error Quantization in Speech PCM and DPCM Systems", *IEEE Trans. Comm.*, April 1972, pp. 225-230.
- Patel, Jagdish K., C. H. Kapadia, and D. B. Owen, *Handbook of Statistical Distributions*, Marcel Dekker (New York), 1976.
- Pfeiffer, R.R. and Duck On Kim, "Response Patterns of Single Cochlear Nerve Fibers to Click Stimuli: Descriptions for Cat", *Jou. Acoust. Soc. Am.*, v52 No 6 (2), pp. 1669-1677.
- Papoulis, Athanasios, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill (New York), 1965.
- Pham, Trung T., and Rui J. P. deFigueiredo, "Maximum Likelihood Estimation of a Class of Non-Gaussian Densities with Application to Deconvolution", *Proc. ICASSP 1987*, pp. 49-52.

- Pierce, J.R., "Whither Speech Recognition", *Jou. Acoust. Soc. Am.*, v46 No 4 (2), 1969, pp. 1049-1051.
- Portiz, Alan B., "Linear Predictive Hidden Markov Models and the Speech Signal", *Proc. IEEE Conf. ASSP*, May 1982, pp. 1291-1294.
- Rabiner, L.R. and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall (Englewood Cliffs, NJ), 1978.
- Rabiner, L.R., S.E. Levinson and M. M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker Independent, Isolated Word Recognition", *Bell Sys. Tech. Jou.*, April 1983, pp. 1075-1105.
- Reddy, D.R., "Computer Recognition of Connected Speech", *Jou. Acoust. Soc. Am.*, v42 No 2, pp. 329-347.
- Rice, John R. and John S. White, "Norms for Smoothing and Estimation", *SIAM Review*, Vol. 6, No. 3, July 1964, pp. 243-256.
- Rockett, Andrew M., and John C. Stephenson, "Karmakar's Algorithm", *Byte*, Sept. 1987, pp. 146-160.
- Schroeder, James and Rao Yarlagadda, "Two Dimensional Linear Predictive Spectral Estimation Via the L_1 Norm", to be published in *Signal Processing*.
- Schroeder, James, "Linear Predictive Spectral Analysis via the L_p Norm", Ph.D. Dissertation, Oklahoma State University, Stillwater, OK, 1985.
- Shannon, C.E., "Prediction and Entropy of Printed English", *Bell Sys. Tech. Jou.*, Jan 1951, pp. 50-64.
- Sittler, R.W., "Systems Analysis of Discrete Markov Processes", *IRE Trans. on Circuit Theory*, Dec 1956, pp. 257-266.
- Sittler, R.W., "Analysis and Design of Simple Non-linear Noise Filters", Ph.D.

- Dissertation, Massachusetts Institute of Technology, Cambridge, Mass., 1954.
- Strang, G., *Introduction to Applied Mathematics*, Wellesley-Cambridge Press (Wellesley, MA), 1986,.
- Taylor, H.L., S.C. Banks and J.F. McCoy, "Deconvolution with the L_1 Norm", *Geophysics*, V. 44 No. 1, Jan 1979, pp. 37-52.
- Thie, Paul R., *An Introduction to Linear Programming and Game Theory*, Wiley and Sons (New York), 1979.
- Tou, J.T. and R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley (Reading Mass), 1974.
- Tremain, Thomas E., "The Government Standard Linear Predictive Coding Algorithm: LPC-10", *Speech Technology*, April 1982, pp. 40-49.
- Viswanathan, R., and John Makhoul, "Efficient Lattice Methods for Linear Prediction", *Programs for Digital Signal Processing*, IEEE Press, Wiley (New York), 1979, Section 4.2.
- Viterbi, Andrew J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Trans. Info. Theory*, April 1967, pp. 260-269.
- Wilson, Gary R., and Dennis R. Powell, "Experimental and Modeled Density Estimates of Underwater Acoustic Returns", *Statistical Signal Processing*, Marcel Dekker (New York), 1984, pp. 223-239.
- Yarlagadda, Rao, J. Bee Bednar and Terry L. Watt, "Fast Algorithms for L_p Deconvolution", *IEEE Trans. ASSP*, ASSP-33, Feb 1985, pp. 174-182.
- Zue, Victor W., "The Use of Speech Knowledge in Automatic Speech Recognition", *Proc. IEEE*, Nov 1985, pp. 1602-1615.
- Zweig, G., R. Lipes, and J. R. Pierce, "The Cochlear Compromise", *Jou.*

Acoust. Soc. Am., April 1976, pp. 975-982.

Zwislocki, J.J., "Five Decades of Research on Cochlear Mechanics", *Jou.*

Acoust. Soc. Am., May 1980, pp. 1679-1685.

VITA

James Lowell Lansford

Candidate for the Degree of

Doctor of Philosophy

Thesis: Lp MODELS IN SPEECH CODING AND
MARKOV CHAINS IN SPEECH RECOGNITION

Major field: Electrical Engineering

Biographical:

Personal Data: Born in Huntland Tennessee, June 9, 1957, the son of Rev. Ewell W. and Ellene Bramblett Lansford. Married to Lynn Milburn on September 9, 1987

Education: Graduated from Scottsboro High School, Scottsboro, Alabama, in May 1975; received Bachelor of Science in Electrical Engineering from Auburn University in March of 1980. Received Master of Science in Electrical Engineering from Georgia Institute of Technology in June of 1982; completed requirements for Doctor of Philosophy degree at Oklahoma State University in May, 1988.

Professional Experience: Lead Engineer, Harris Corp., 1982 to 1985; teaching assistant, Oklahoma State University, 1985 to 1988. Senior Research Engineer, Georgia Tech Research Institute, 1988 to present