AD-A070 050 STANFORD UNIV CALIF INFORMATION SYSTEMS LAB F/G 17/2
FAST ALGORITHMS FOR SPEECH MODELING.(U)
DEC 78 M MORF, D T LEE DCA100-77-C-0005
UNCLASSIFIED NL

1 OF 3
AD
A070050

10

INFORMATION SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

LELAND STANFORD JUNIOR UNIVERSITY · ORGANIZED 1891

LEVEL II

FINAL REPORT TO THE

DEFENSE COMMUNICATION AGENCY

FOR RESEARCH ON

FAST ALGORITHMS FOR SPEECH MODELING

Contract Number: DCA100-77-C-0005,

For a period of one Year

December 8, 1976 - December 8, 1977

Report Date: December 15, 1978

STANFORD UNIVER~~SITY~~ CA.

INFORMATION SYSTEMS LAB~~ORATORY~~

STANFORD, CA 94305

(9) FINAL REPORT, TO THE

8 Dec 76 - 8 Dec 77,

DEFENSE COMMUNICATION AGENCY

FOR RESEARCH ON

(6) FAST ALGORITHMS FOR SPEECH MODELING.

(15) Contract Number: DCA100-77-C-0005

(10) Martin/Morf D. T./Lee (11) 15 Dec 78

For a period of one Year (12) 231p.

December 8, 1976 - December 8, 1977

Report Date: December 15, 1978

406 720

LB

Abstract

This constitutes our final report on a research program aimed at the development of a high quality low data rate speech transmission system based on new types of speech modeling algorithms. Several such algorithms were developed and tested on simulated and real speech data. These algorithms have many desirable features including the capability of rapidly tracking time-varying model parameters.

The best algorithm was used as the basis of a speech transmission system in order to test the quality of the speech models. The model parameters (reflection coefficients) together with pitch information and speech energy form a speech parameter vector to be transmitted and used to reconstruct the original speech. Several parameter quantization methods were considered to achieve the desired low bit rates.

The various algorithms as well as the complete transmission system were coded and tested. Simulation results are very promising and the usefulness of our new approach for speech modeling has been successfully established.

M. Morf

D. T. Lee

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Stanford Electronics Laboratories Department of Electrical Engineering Stanford University, Stanford, CA 94305	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE
FINAL REPORT TO THE DEFENSE COMMUNICATION AGENCY FOR RESEARCH ON FAST ALGORITHMS FOR SPEECH MODELING

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
FINAL REPORT - December 8. 1976 - December 8, 1977 (For period of one year)

5. AUTHOR(S) (First name, middle initial, last name)
Dr. Martin Morf
D. T. Lee

6. REPORT DATE December 15, 1978	7a. TOTAL NO. OF PAGES 225	7b. NO. OF REFS 51
8a. CONTRACT OR GRANT NO. Contract #DCA100-77-C-0005 b. PROJECT NO. N/A c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) N/A	

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES N/A	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

This constitutes our final report on a research program aimed at the development of a high quality low data rate speech transmission system based on new types of speech modeling algorithms. Several such algorithms were developed and tested on simulated and real speech data. These algorithms have many desirable features including the capability of rapidly tracking time-varying model parameters.

The best algorithm was used as the basis of a speech transmission system in order to test the quality of the speech models. The model parameters (reflection coefficients) together with pitch information and speech energy form a speech parameter vector to be transmitted and used to reconstruct the original speech. Several parameter quantization methods were considered to achieve the desired low bit rates.

The various algorithms as well as the complete transmission system were coded and tested. Simulation results are very promising and the usefulness of our new approach for speech modeling has been successfully established.

UNCLASSIFIED

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Speech Modeling, Speech Coding, Linear Predictive Coding.						

UNCLASSIFIED
Security Classification

Acknowledgements

The content of this report is the result of a two year effort into the development of fast algorithms for real-time modeling of a class of signal sources that can be characterized by linear finite dimensional models, of the autoregressive (AR) and/or moving-average (MA) type, driven either by gaussian type noise (e.g. unvoiced speech) or a process containing jumps or impulses (e.g. voiced speech).

For the first year of this research the main emphasis was the application of these algorithms to speech modeling, supported by DCA. During the second year the development of multichannel versions of these algorithms with a fast parameter tracking capability was emphasized, as well as the use of on-line estimates of log-likelihood functions for jump process component or "event" detection. This effort was supported by the Defense Advanced Research Project Agency, who also supported the computer facility used for this project at the Stanford Artificial Intelligence Laboratory, and the production of this document.

Several people have contributed to this project. Augusto Vieira, now in Brazil, helped develop the early theoretical framework for the fast ladder algorithms. John Nickolls, now at AMPEX Corporation, Redwood City, designed and programmed the supporting modular framework as well as some of the earlier experiments. Ben Friedlander, now at Systems Control Inc., Palo Alto, has contributed in the writing of earlier versions of this report. Larry Stewart voluntarily spent many nights designing experiments, running algorithms and helping in the writing up of the results. R. Morf contributed in the editing of this document.

Table of Contents

1. Introduction

This investigation was concerned with the development of digital voice communication systems capable of low data transmission rates. Such systems construct a time-varying linear model of the speaker's vocal tract, and transmit the encoded model parameters over a digital network. The receiver reconstructs the model from the coded parameters and synthesizes an approximation of the original speech signal. The model is traditionally constructed by the method of *linear predictive coding* (LPC), which predicts future speech samples as linear combinations of past speech samples, where the linear combination is chosen to minimize the prediction error. This results in a vector A of coefficients which characterize the speech production mechanism in terms of an inverse filter $a(z)$ (the vocal tract is considered the filter $1/a(z)$). The A coefficients can be used to reconstruct the speech signal. In practice they must be encoded to achieve low bit rates, but this problem can be separated from the modeling problem proper.

Our goal was to investigate the application of new linear estimation algorithms to speech modeling. This involves both modeling and encoding issues. There are a large number of ways to approach the speech modeling problem, but here we restrict our attention to exact least-squares linear estimation procedures (there is currently no reason to examine sub-optimal or approximate methods). These estimation methods find linear models which fit the (speech) data optimally in terms of minimizing the sum of the squared errors--hence the term "least-squares". The LPC methods currently used in speech modeling are least-squares estimation procedures which find all-pole or *autoregressive* (AR) models. The assumption that an all-pole model is sufficient is valid for vowel sounds (disregarding sound radiation). However nasal sounds require zeros and a pole-zero or autoregressive-moving average (ARMA) model should produce a more efficient speech encoding. Another aspect of our modeling effort is the extensive use of ladder-form realizations and their reflection coefficient K for speech parametrizations. The *K* coefficients have many advantages over other model parametrizations, such as better numerical properties and fast convergence, and can be used directly in a ladder-form synthesis structure. Each representation can be converted to the other in a one-to-one fashion, but the *K* coefficients have physical significance in speech modeling because they correspond to acoustic reflection coefficients in a segmented tube model of the vocal tract. Actually, it is possible to modify the Levinson recursion to avoid the use of the prediction parameters A when computing the reflection coefficients, as mentioned in [MLNV]. See also [Vieira].

Various new speech modeling algorithms were developed using the techniques associated with our fast algorithms:

- Pre-windowed ladder-form (AR)
- Covariance ladder-form (AR)
- Pole-zero ladder-form (ARMA)

All of these techniques use exact recursive least-squares parameter estimation algorithms, i.e. they are ideal on-line modeling methods, with fast "adaptive" properties. Their computational requirements (per sample) are proportional only to n, the number of model parameters - again a feature that is well suited for hardware, parallel processing or pipeline implementation of our algorithms.

The implementation of the *pre-windowed* (PW) ladder-form has been enhanced with the introduction of tracking of time-varying parameters. This can only be done with an on-line method and meshes naturally with the PW ladder formulation. The effect is that the parameter estimates track the actual dynamics by weighting recent data more heavily than older data. The dynamics in model order can also be tracked. These tracking capabilites are necessary for estimates of transients (or transemes) or plosives. The weighted forms of the covariance ladder and the rational ladder algorithms were developed. We discovered that the tracking ability of our algorithm is actually even better for voiced speech, as the glottal pulses help the parameters to converge within a few samples virtually to a constant over a pitch period - a fact that leads to reduced data rates. For unvoiced speech (i.e.

Gaussian type residuals) the tracking is still fast but very smooth - another desirable feature.

This PW ladder-form algorithm provides the basis for our digital speech transmission system. The system consists of a speech analyzer which produces a (slowly) time varying parameter vector, an encoder that converts a single frame of speech parameter vectors into a binary data stream, and a decoder that converts the binary stream into parameter vectors which are used by the speech synthesizer to reconstruct a signal that sounds like the original speech. The speech parameter vector consists of the reflection coefficients, the pitch period (or time index of the beginning of the pitch period), and the energy contained in the speech frame (or equivalently in the residuals). The pitch information is obtained by a novel pitch detection method, resulting from our recursive ladder-form algorithm, using a log likelihood ratio parameter that is computed by the algorithm in order to separate out the jump process type pitch pulses from the residuals.

Several quantizing methods were considered, for moderate bit rates, (e.g. 4800 - 9600), single symbol quantization, i.e. independent quantization of each parameter is considered sufficient. For lower (e.g. 1200 and below) rates, a new parameter vector quantization scheme based on a minimum distortion measure was considered. Such methods are being developed by R. M. Gray at Stanford (under AFOSR sponsorship). (See [Buzo].) These new quantization schemes are still in the development stage; however they are sufficiently promising so that a short sample speech segment was quantized with approximately 3 db Itakura-Saito rate distortion deviation (from the unquantized reconstruction) at a rate of roughly 700 bits per second. In this new method the parameter space of the reflection coefficients is partitioned into a number of cells. Whenever the parameter vector falls within a given cell, the binary number representing that cell is transmitted. The partitioning of the parameter space is chosen so as to minimize a given, e.g. Itakura-Saito, distortion measure. In the actual (on-line) quantization, a mean-square error (Euclidean distance) criterion is used to pick the actual code transmitted. These methods have great potential to provide high quality low bit rate digital voice encoding for the future. In the simplest case, pitch period and (log) energy or gain are envisioned to be coded via a standard delta modulation type scheme; however, this is only considered in order to be able to compute an achievable lower bound on the transmission rate. A real implementation could use more sophisticated coding schemes, a task beyond the scope of this research. Using rate distortion encoding schemes, a given (low) rate can be achieved with a minimal loss in speech quality, once a suitable distortion measure (such as the Itakura-Saito) has been agreed apon. A number of simulations were performed on the complete transmission systems and the results so far are very promising.(See Section 5.)

This final project report presents the theoretical results of our research, the actual algorithm implementations, and the simulation results of the first year of an originally estimated two years worth of research. Sections 2 and 3 present all the analytical work that was performed. Section 4 and Appendix C discuss the software generated as a result of our analysis, and Section 5 describes the actual simulation results.

2. Algorithms for Speech Modeling

A number of new algorithms for speech modeling were developed in the course of our investigation, using our fast algorithm approach to estimation and system identification. The main features of these algorithms are

- They are *exact least-squares methods*

- *They are recursive in time and in the order of the model and thus capable of processing data as it comes along (i.e. not block-by-block, unless desired)*

- *Compute directly either the predictor coefficients (AR model) or the reflection coefficients (ladder form); provide true unbiased estimates*

- *Capability of tracking time-varying model parameters*

- *Good stability and convergence properties*

- *Computationally efficient*

In this section we describe in detail the development of several algorithms. For several reasons, the emphasis is on the development of ladder-form realizations. In particular the reflection coefficients appearing in these forms turn out to be an excellent way of parametrizing speech. Both autoregressive (AR) and pole-zero or autoregressive moving-average (ARMA) type algorithms are derived.

Sections 2.1 - 2.4 present the detailed derivation of the algorithms for the pre-windowed and non-windowed (covariance-form) ladder-forms. The pre-windowed versions of these algorithms play a central role in our investigation, and Section 2.3 discusses the necessary modifications to make them capable of tracking time varying parameters. Pole-zero or autoregressive moving-average (ARMA) algorithms are derived in Section 2.5 .

The computational requirements of various algorithms are summarized in Section 2.6 and compared to currently used methods. Finally, some difficulties which arose during the algorithm implementation phase and the method by which they were rectified are briefly described in Section 2.7 .

For an overview of the various algorithms derived in this section and the way they are related to each other - see Appendix A. The importance of ladder form realizations in estimation and modeling is briefly summarized in Appendix B.

2.1 Autoregressive Models

In this section we introduce a framework which is later used for developing several exact least-squares algorithms for AR - type models. The basic problem is presented from an estimation theory approach which leads to the deterministic least-squares framework.

2.1.1 Basic Problem

We model speech as the output of a time-varying linear system, which, over a short time interval, can be approximated by a time-invariant filter of the form

$$y(z) = H(z)\ u(z) , \tag{1}$$

with $y(z)$ being the z-transform of the discretized speech signal, $H(z)$ the overall transfer function, and $u(z)$ the input driving function which consists of a periodic pulse train (approximating the glottal pulses for voiced sounds), and zero-mean white noise (for unvoiced sounds). Such a model is widely accepted by the speech research community as a good description of the speech generation process. Detailed discussions on this model can be found in [MKD] and [Fla]. A particularly popular model, see e.g. [MG], is one where $H(z)$ is a finite order all-pole filter

$$\begin{aligned} H(z) &= 1 \,/\, (1 + \sum_{k=1}^{P} A_P^{(k)} z^{-k}) \\ &= 1 \,/\, A_P(z) \quad . \end{aligned} \tag{2}$$

Such a model is equivalent to modeling $\{ y(\cdot) \}$ as an *autoregressive* (AR) process

$$y_t + A_P^{(1)} y_{t-1} + \ . \ . \ . + A_P^{(P)} y_{t-P} = u_t \quad . \tag{3}$$

Rewriting (3) as

$$y_t = - \sum_{k=1}^{P} A_P^{(k)} y_{t-k} + u_t \ , \tag{4}$$

the all-pole filter of (2) forms a one-step linear predictor for $\{ y(\cdot) \}$

$$\hat{y}_{t|[t-1,t-P]} = - \sum_{k=1}^{P} A_P^{(k)} y_{t-k} \quad . \tag{5}$$

A very common and practical criterion is to choose the predictor that minimizes the squares of the prediction errors, thus leading to a least-squares estimation problem.

We assume for the moment that $\{y(\cdot)\}$ (for generality we let y_t be m-dimensional vectors) is a stationary process with known covariance m by m matrix function

$$\begin{aligned} E\ y_t {y^{\mathsf{T}}}_{t-k} &= R_k , \\ R_{-k} &= R_k^{\mathsf{T}} , \quad k = 0, 1, \ldots, P \end{aligned} \qquad . \tag{6}$$

The innovations or the forward prediction errors are then given by

$$\begin{aligned} \epsilon_{P,t} &= y_t - \hat{y}_{t|[t-1,\, t-P]} \\ &= y_t + A_P^{(1)}\, y_{t-1} + \ldots + A_P^{(P)}\, y_{t-P}, \\ &= A^{\mathsf{T}}{}_P\ \mathcal{Y}_{[t:t-P]} , \end{aligned} \tag{7}$$

with

$$\begin{aligned} A^{\mathsf{T}}{}_P &= [\ \mathrm{I}_m\ ,\ A_P^{(1)}, \ldots, A_P^{(P)}\ ] , && (\,m \text{ by } (P+1)m\,) \\ {\mathcal{Y}_{[t:t-P]}}^{\mathsf{T}} &= [\ y_t^{\mathsf{T}}\ ,\ y_{t-1}^{\mathsf{T}}\ , \ldots ,\ y_{t-P}^{\mathsf{T}}\ ] , && (1 \text{ by } (P+1)m\,) \\ \mathrm{I}_m &= \textit{the } m \textit{ by } m \textit{ identity matrix.} \end{aligned} \tag{8}$$

The innovations should satisfy the following orthogonality property

$$E\ \ \epsilon_{P,t}\, y_k^{\mathsf{T}} = 0 , \qquad t - P \le k \le t - 1 \tag{9}$$

$$\begin{aligned} E\ \ \epsilon_{P,t}\, \epsilon_{P,t}^{\mathsf{T}} &= E\ \epsilon_{P,t}\, y_t^{\mathsf{T}} \\ &= R^{\epsilon}{}_P \end{aligned} \qquad . \tag{10}$$

From (9) and (10), we see that A_P can be obtained as the solution of the linear matrix equation called the *Normal Equation*

$$R_P\ A_P = \begin{bmatrix} R^{\epsilon}{}_P \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} , \tag{11}$$

where R_P is the $(P+1)m$ by $(P+1)m$ block Toeplitz matrix

$$R_P = E\, Y_{[t:t-P]}\, Y^{\mathsf{T}}_{[t:t-P]} \tag{12}$$

Writing it out in full

$$R_P = \begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_P \\ R_{-1} & R_0 & R_1 & \cdots & R_{P-1} \\ R_{-2} & R_{-1} & \cdots & & \cdot \\ \cdot & \cdot & \cdots & & R_{-1} \\ R_{-P} & \cdot & \cdots & & R_0 \end{bmatrix}. \tag{13}$$

We note that by virtue of stationarity of $\{ y(\cdot) \}$ that $R^{\epsilon}{}_P$ is independent of t and that the covariance matrix R_P is Toeplitz. This special structure makes it possible to solve the normal equation (11) with fewer than the $O(P^3)$ computations generally necessary to solve P linear equations in P unknowns. In fact, Levinson [Lev] and then, for vector processes, Whittle and independently Wiggins and Robinson [WR], derived a scheme for solving (11) with $O(P^2)$ computations. (Here a computation is taken as one multiplication of two real numbers.) This algorithm, which we call the *LWR algorithm*, involves the simultaneous solution of (11) and an auxiliary equation

$$R_P\, B_P = [\ 0,\ 0,\ \ldots,\ R^r{}_P\ ]^{\mathsf{T}},$$

where

$$B^{\mathsf{T}}{}_P = [\ B_P{}^{(P)},\ B_P{}^{(P-1)}, \ldots,\ B_P{}^{(1)},\ I_m\ ], \tag{14}$$

and is actually a backwards predictor, with backwards prediction errors defined by

$$\begin{aligned} r_{P,t} &= y_{t-P} - \hat{y}_{t-P|[t-P+1,\, t]} \\ &= B_P{}^{(P)} y_t + \ldots + B_P{}^{(1)} y_{t-P+1} + y_{t-P}, \\ &= B^{\mathsf{T}}{}_P\, Y_{[t:t-P]}, \end{aligned} \tag{15}$$

and

$$E\ r_{P,t}\, y_k{}^{\mathsf{T}} = 0, \qquad t-P+1 \le k \le t$$

$$E \; r_{P,t} \, r_{P,t}^{\mathsf{T}} = E \; r_{P,t} \, y_{t-P}^{\mathsf{T}} = R^r{}_P \tag{16}$$

The basic idea of the *LWR algorithm* is to compute A_m and B_m recursively in order from $n = 0$ to P . Here we give the recursiona of the algorithm, and a detailed discussion can be found in [K-74] and [Vie].

2.1.2 LWR Algorithm

Iterate on n=0 *until* n=P

$$\begin{bmatrix} R^{\epsilon}{}_{n+1} & 0 \\ A_{n+1} & B_{n+1} \\ 0 & R^r{}_{n+1} \end{bmatrix} = \begin{bmatrix} R^{\epsilon}{}_n & \Delta_{n+1}{}^{\mathsf{T}} \\ \begin{bmatrix} A_n \\ \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ \\ B_n \end{bmatrix} \\ \Delta_{n+1} & R^r{}_n \end{bmatrix} \Theta_{n+1} \tag{17}$$

where

$$\Theta_{n+1} = \begin{bmatrix} I_m & -R^{-\epsilon}{}_n \, \Delta_{n+1}{}^{\mathsf{T}} \\ -R^{-r}{}_n \, \Delta_{n+1} & I_m \end{bmatrix} \tag{18}$$

$$\Delta_{n+1} = \sum_{k=0}^{n-1} A_n{}^{(n-k)} \, R_k = E[\, \epsilon_{n,t} \, y^{\mathsf{T}}{}_{t-n-1} \,] = E[\, \epsilon_{n,t} \, r^{\mathsf{T}}{}_{n,t-1} \,] \tag{19}$$

with initial conditions:

$$R^{\epsilon}{}_0 = R^r{}_0 = R_0,$$
$$A_0 = B_0 = I_m \tag{20}$$

The Δ_{n+1} defined in (19) is known as the *partial correlation coefficient* (PARCOR) between the forward and backward prediction errors, and when appropriately normalized by $R^{\epsilon}{}_{n}$ and $R^{r}{}_{n}$, they become the so-called *reflection coefficients.*

A compact way to rewrite the algorithm is

$$\begin{bmatrix} R^{\epsilon}{}_{P} & 0 \\ & \\ A_P & B_P \\ & \\ 0 & R^{r}{}_{P} \end{bmatrix} = \begin{bmatrix} R_0 & \Delta_1{}^{T} \\ I_m & 0 \\ & \\ 0 & I_m \\ R_1 & R_0 \end{bmatrix} \Theta_1\, \Theta_2\, \Theta_3 \ldots \Theta_P \;. \tag{21}$$

The important point which (21) brings out is that both A_P and B_P can be completely characterized by $\{ \Theta_n ,\, n = 1, \ldots, P \}$, and therefore by $\{ R_0 ; \Delta_n ,\, n = 1, \ldots, P \}$. It turns out that this parametrization of the predictor offers many advantages such as "stability by inspection" property" [MLVN] and they form the basis for implementing the predictor in *ladder forms.* The development of various *ladder forms* will be presented in the subsequent sections.

In real time applications, no ensemble averages are available. The covariance functions usually are not directly obtainable and must be approximated by time averages from the given data. This leads to the following deterministic least-squares problem.

2.1.3 Deterministic Least-Squares Problem

Given a series of observations $\{ y(t),\ S \le t \le T \}$, where $\{ y(\cdot) \}$ can be m-dimensional vectors, we wish to find the linear one-step predictor of order P, parametrized by the (matrix) predictor coefficients $\{ -A_{P,S,T}^{(i)},\ i=1,\dots,P \}$, that minimizes the sum of the square of prediction errors $\epsilon_{P,S,T}(t)$, where

$$\begin{aligned}\epsilon_{P,S,T}(t) &= y_t - \hat{y}_{t|t-1,\dots,t-P} \\ &= y_t + A_{P,S,T}^{(1)}\, y_{t-1} + \dots + A_{P,S,T}^{(P)}\, y_{t-P}, \qquad S \le t \le T. \end{aligned} \tag{22}$$

In matrix notations, we have

$$\epsilon_{P,S,T}(t) = A^{\mathsf{T}}_{P,S,T}\ \mathcal{Y}_{[t:t-P]}\ , \tag{23}$$

with

$$\begin{aligned} A^{\mathsf{T}}_{P,S,T} &= [\ I_m\ ,\ A_{P,S,T}^{(1)},\ \dots\ ,\ A_{P,S,T}^{(P)}\ ], \\ \mathcal{Y}_{[t:t-P]} &= [\ y_t^{\mathsf{T}}\ ,\ y_{t-1}^{\mathsf{T}}\ ,\ \dots\ ,\ y_{t-P}^{\mathsf{T}}\ ]^{\mathsf{T}}. \end{aligned} \tag{24}$$

We can express the squared-error, $\xi_{P,S,T}$, by

$$\begin{aligned}\xi_{P,S,T} &= tr\{ \sum_{t=i}^{f} \epsilon_{P,S,T}(t)\ \epsilon^{\mathsf{T}}_{P,S,T}(t) \} \\ &= tr\{ A^{\mathsf{T}}_{P,S,T}\ R_{P,S,T}\ A_{P,S,T} \}\ , \end{aligned} \tag{25}$$

with $R_{P,S,T}$ being the sample covariance matrix given by

$$R_{P,S,T} = Y_{P,S,T}\ Y^{\mathsf{T}}_{P,S,T}\ , \tag{26}$$

and

$$Y_{P,S,T} = \begin{bmatrix} y_i & \cdots\cdots & y_f \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ y_{i-P} & \cdots\cdots & y_{f-P} \end{bmatrix}. \tag{27}$$

It is well-known in least-squares theory that the $A_{P,S,T}$ that minimizes $\xi_{P,S,T}$ is obtained by solving a linear matrix equation called the *Normal Equation* of the following form

$$R_{P,S,T}\,A_{P,S,T} = \begin{bmatrix} R^{\epsilon}_{P,S,T} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{28}$$

where

$$tr\ R^{\epsilon}_{P,S,T} = min_{A}\ \xi_{P,S,T} . \tag{29}$$

The solution for $A_{P,S,T}$ in (28), for the scalar case, would involve the inversion of $R_{P,S,T}$, an n *by* n matrix, and thus would require $O(n^3)$ computations. However, when $R_{P,S,T}$ carries some shift-invariance structure, for example a Toeplitz matrix, a reduction of computations is achieved. In the case where $R_{P,S,T}$ is Toeplitz, equation (28) can be solved via the *Levinson algorithm* [MG], [MVLK], requiring only $O(n^2)$ computations. The basic approach is to build up the predictor recursively in order, i.e. by recursively obtaining $\{A_{p,S,T},\ p = 1, \ldots, P\}$.

In our present problem, the structure of $R_{P,S,T}$ depends on the choice of i and f. Here we consider the following three cases of importance:

(1) $i = S,\ f = T$.

This is called the "*pre-windowed*" case, since one has to make the assumption that $y(t) = 0$, for $t < S$. Thus $Y_{P,S,T}$ becomes

$$Y_{P,S,T} = \begin{bmatrix} y_S & \cdots & y_{S+P} & \cdots & y_T \\ & \cdot & \cdot & & \cdot \\ & & \cdot & \cdot & \cdot \\ 0 & & \cdot & \cdot & \cdot \\ & & y_S & \cdots & y_{T-P} \end{bmatrix}. \tag{30}$$

(2) $i = S + P,\ f = T$.

This is called the "*non-windowed*" case, i.e. no windowing is applied to the observed data. This is also known as the "*covariance*" method. In this case $Y_{P,S,T}$

becomes

$$Y_{P,S,T} = \begin{bmatrix} y_{S+P} & \cdots\cdots\cdots & y_T \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ y_S & \cdots\cdots\cdots & y_{T-P} \end{bmatrix}. \qquad (31)$$

(3) $i = S,\ f = T + P.$

This is called the "*pre- and post-windowed*" case, since one must now assume that $y(t) = 0$, for $t < S$ and $t > T$. This is also known as the "*autocorrelation*" method. In this case zeros are added both before the first sample and after the last sample, and $Y_{P,S,T}$ takes the form

$$Y_{P,S,T} = \begin{bmatrix} y_S & \cdots & y_{S+P} & \cdots & y_T & & \\ & \cdot & \cdot & & \cdot & \cdot & 0 \\ & & \cdot & \cdot & \cdot & \cdot & \\ 0 & & \cdot & \cdot & \cdot & \cdot & \\ & & y_S & \cdots & y_{T-P} & \cdots & y_T \end{bmatrix}. \qquad (32)$$

The names "*covariance*" method and "*autocorrelation*" method are traditional in the speech processing literature, but from a statistical point of view such nomenclature is not completely justified.

We may note that only in the "*pre- and post-windowed*" or "*autocorrelation*" method, $R_{P,S,T}$ is a Toeplitz matrix, while in the other two case it is no longer Toeplitz. However, even though $R_{P,S,T}$ is non-Toeplitz when defined in (30) or (31), it is the product of two Toeplitz matrices and therefore still carries a certain shift-invariance structure. A class of algorithms are presented in [FMDK] and [MDKV] for inverting matrices which are sums of products of Toeplitz matrices and the algorithm as investigated here is a special case of that class.

2.2 The Pre-Windowed Ladder Form

In this section we present a detailed derivation of the *pre-windowed* (PW) ladder form. At the end of the section we will show that this particular form is actually a good approximation to a recursive maximum likelihood method for the autoregressive model.

2.2.1 Algorithm Development

For notational convenience, we let the observations start at time zero, i.e. $S = 0$, and thus from here on we simply drop the S index altogether. Thus in the *pre-windowed* case the covariance matrix for order p has the form

$$R_{p,T} = Y_{p,T}\, Y_{p,T}^{T}, \tag{1}$$

$$Y_{p,T} = \begin{bmatrix} y_0 & \cdots & y_p & \cdots & y_T \\ & \cdot & \cdot & & \cdot \\ & \cdot & \cdot & & \cdot \\ 0 & \cdot & \cdot & & \cdot \\ & & y_0 & \cdots & y_{T-p} \end{bmatrix}. \tag{2}$$

The matrix $R_{p,T}$ defined above satisfies the following shifting properties (or recursive identities).

Order-update (down-shift):

$$R_{p+1,T} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2^{T} \\ \mathbf{x}_2 & R_{p,T-1} \end{bmatrix} \tag{3}$$

Order-update (up-shift):

$$R_{p+1,T} = \begin{bmatrix} R_{p,T} & x_3 \\ x_3^{\mathsf{T}} & x_4 \end{bmatrix} . \tag{4}$$

Time-update:

$$R_{p,T+1} = R_{p,T} + \begin{bmatrix} y_{T+1} \\ \cdot \\ \cdot \\ y_{T-p+1} \end{bmatrix} [\, y^{\mathsf{T}}_{T+1} , \ldots , y^{\mathsf{T}}_{T-p+1} \,] , \tag{5}$$

where the x's represent unspecified matrix elements along the appropriate edges.

Define $A_{p,T}$, $B_{p,T}$, and $C_{p,T}$ for $p = 0, 1, \ldots, P$ by

$$R_{p,T} [\, A_{p,T} , B_{p,T} , C_{p,T} \,] = \left[\begin{array}{c|c|c} R^{\epsilon}{}_{p,T} & 0 & y_T \\ 0 & 0 & y_{T-1} \\ \cdot & \cdot & \cdot \\ 0 & 0 & y_{T-p+1} \\ 0 & R^{r}{}_{p,T} & y_{T-p} \end{array}\right] , \tag{6}$$

where $A_{p,T}$ and $B_{p,T}$ are respectively the forward and backward predictors of the form

$$A^{\mathsf{T}}{}_{p,T} = [\, I_m , A_{p,T}{}^{(1)} , \ldots , A_{p,T}{}^{(p)} \,] , \tag{7}$$

$$B^{\mathsf{T}}{}_{p,T} = [\, B_{p,T}{}^{(p)} , \ldots , B_{p,T}{}^{(1)} , I_m \,] , \tag{8}$$

and $C_{p,T}$ is an auxiliary vector which can also be expressed by

$$C_{p,T} = R_{p,T}{}^{-1} \, Y_{[T:T-p]} . \tag{9a}$$

and

$$C_{p,T}{}^{\mathsf{T}} = Y^{\mathsf{T}}{}_{[T:T-p]} \, R_{p,T}{}^{-1}. \tag{9b}$$

We define $\epsilon_{p,T}$, the forward prediction errors or innovations, and $r_{p,T}$, the backward prediction errors by

$$\begin{bmatrix} \epsilon_{p,T} \\ r_{p,T} \end{bmatrix} = \begin{bmatrix} A^{\mathsf{T}}_{p,T} \\ B^{\mathsf{T}}_{p,T} \end{bmatrix} \begin{bmatrix} y_T \\ y_{T-1} \\ \cdot \\ y_{T-p+1} \\ y_{T-p} \end{bmatrix} . \tag{10}$$

We define an auxiliary quantity $\gamma_{p,T}$ by

$$\gamma_{p,T} = C^{\mathsf{T}}_{p,T} \; Y_{[T:T-p]} \; . \tag{11a}$$

From the definition of $C_{p,T}$ given in (6), $\gamma_{p,T}$ can be interpreted as the weighted energy of the observations $\{ y_{T-p} , \ldots , y_T \}$, which can also be expressed as

$$\gamma_{p,T} = [y^{\mathsf{T}}_T , \ldots , y^{\mathsf{T}}_{T-p}] \; R^{-1}_{p,T} \begin{bmatrix} y_T \\ \cdot \\ y_{T-p} \end{bmatrix} . \tag{11b}$$

It also has an interpretation as a likelihood variable which we will discuss furthur in a later part of this section.

2.2.2 Order Update Recursions

Suppose we already have the predictors $A_{p,T}$ and $B_{p,T}$ and want to increase the predictors order to $p+1$. We therefore would like $A_{p+1,T}$ and $B_{p+1,T}$ to satisfy the normal equation

$$R_{p+1,T} \, [\, A_{p+1,T} \, , \, B_{p+1,T} \,] = \left[\begin{array}{c|c} R^{\epsilon}{}_{p+1,T} & 0 \\ 0 & 0 \\ . & . \\ 0 & 0 \\ 0 & R^{r}{}_{p+1,T} \end{array} \right] \tag{12}$$

and we start by using relation (4) :

$$R_{p+1,T} \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{p,T} & & x \\ x & x & x \end{bmatrix} \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} R^{\epsilon}{}_{p,T} \\ 0 \\ . \\ 0 \\ \Delta_{p+1,T} \end{bmatrix} \tag{13}$$

where

$$\Delta_{p+1,T} = [\, \textit{last block-row of } R_{p+1,T} \,] \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} = \sum_{t=0}^{T} y_{t-p-1} \, \epsilon^{T}{}_{p,T}(t) \quad . \tag{14}$$

Here, we can relate $\Delta_{p+1,T}$ to the partial correlations discussed in the previous section.

Similarly, using the relation (3) :

$$R_{p+1,T} \begin{bmatrix} 0 \\ \\ B_{p,T-1} \end{bmatrix} = \begin{bmatrix} x & x & x \\ x & & \\ x & R_{p,T-1} & \end{bmatrix} \begin{bmatrix} 0 \\ \\ B_{p,T-1} \end{bmatrix} = \begin{bmatrix} \Gamma_{p+1,T} \\ 0 \\ 0 \\ 0 \\ 0 \\ R^r{}_{p,T-1} \end{bmatrix}, \qquad (15)$$

where

$$\Gamma_{p+1,T} = [\, \textit{first block-row of } R_{p+1,T} \,] \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix}$$

$$= \sum_{t=1}^{T} y_t \; r^{\mathsf{T}}{}_{p,T-1}(t-1) \; . \qquad (16)$$

We can show that $\Delta_{p+1,T} = \Gamma^{\mathsf{T}}{}_{p+1,T}$ by noting that $R_{p+1,T}$ is symmetric, and

$$\begin{bmatrix} A^{\mathsf{T}}{}_{p,T} & 0 \\ \\ 0 & B^{\mathsf{T}}{}_{p,T-1} \end{bmatrix} R_{p+1,T} \begin{bmatrix} A_{p,T} & 0 \\ \\ 0 & B_{p,T-1} \end{bmatrix}$$

$$= \begin{bmatrix} A^{\mathsf{T}}{}_{p,T} & 0 \\ \\ 0 & B^{\mathsf{T}}{}_{p,T-1} \end{bmatrix} \begin{bmatrix} R^{\epsilon}{}_{p,T} & \Gamma_{p+1,T} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta_{p+1,T} & R^r{}_{p,T-1} \end{bmatrix}$$

$$= \begin{bmatrix} R^{\epsilon}{}_{p,T} & \Gamma_{p+1,T} \\ \Delta_{p+1,T} & R^r{}_{p,T-1} \end{bmatrix} \; . \qquad (17)$$

By symmetry, we establish the identity

$$\Delta_{p+1,T} = \Gamma^{\mathsf{T}}{}_{p+1,T} \; . \qquad (18)$$

Thus postmultiplying (15) by $R^{-r}{}_{p,T-1}\,\Delta_{p+1,T}$, where R^{-r} is the inverse of R^r, and then subtracting the result from the right hand side of (13), we have the following order update recursions for $A_{p,T}$ and $R^{\epsilon}{}_{p,T}$:

$$A_{p+1,T} = \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix} R^{-r}{}_{p,T-1}\,\Delta_{p+1,T} \tag{19}$$

$$R^{\epsilon}{}_{p+1,T} = R^{\epsilon}{}_{p,T} - \Delta^{\mathsf{T}}{}_{p+1,T}\,R^{-r}{}_{p,T-1}\,\Delta_{p+1,T}\,. \tag{20}$$

A similar set of order update recursions for $B_{p,T}$ and $R^r{}_{p,T}$ are obtained as

$$B_{p+1,T} = \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix} - \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} R^{-\epsilon}{}_{p,T}\,\Delta^{\mathsf{T}}{}_{p+1,T} \tag{21}$$

$$R^r{}_{p+1,T} = R^r{}_{p,T-1} - \Delta_{p+1,T}\,R^{-\epsilon}{}_{p,T}\,\Delta^{\mathsf{T}}{}_{p+1,T}\,. \tag{22}$$

To obtain the order update recursions for $C_{p,T}$, we first observe that the last block row of $R^{-1}{}_{p,T}$ is equal to $R^{-r}{}_{p,T}\,B^{\mathsf{T}}{}_{p,T}$. (This can be obtained from the normal equation for $B_{p,T}$.) Thus from the definition of $C_{p,T}$, we can obtain the last block-row of $C_{p,T}$ as

$$\textit{last block row of } C_{p,T} = R_{p,T}{}^{-1}\,Y_{[T:T-p]} = R^{-r}{}_{p,T}\;r_{p,T}\,. \tag{23}$$

Using the relation (4) , we have

$$R_{p+1,T}\begin{bmatrix} C_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{p,T} & & x \\ x & x & x \end{bmatrix}\begin{bmatrix} C_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} y_T \\ y_{T-1} \\ \cdot \\ y_{T-p+1} \\ \delta C_{p,T} \end{bmatrix}, \tag{24}$$

where

$$\delta C_{p,T} = [\ \textit{last block row of}\ R_{p+1,T}\ ]\ [\ C_{p,T}^{\mathsf{T}}\ ,\ 0\ ]^{\mathsf{T}} \tag{25}$$

Here we want to show that

$$\delta C_{p,T} = \hat{y}_{T-p-1|[T-p:T]}\ .$$

Partitioning $R_{p+1,T}$ as given in (4), we write the *Normal Equation* for $B_{p+1,T}$ as follows

$$\begin{bmatrix} R_{p,T} & x \\ & \\ x^{\mathsf{T}} & z \end{bmatrix} \begin{bmatrix} B^*_{p+1,T} \\ \\ I_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ R^r_{p+1,T} \end{bmatrix} \tag{26}$$

where

$$B^*_{p+1,T} = [\ B^{(p)}{}_{p,T}^{\mathsf{T}}\ ,\ .\ .\ .\ .\ ,\ B^{(1)}{}_{p,T}^{\mathsf{T}}\ ]^{\mathsf{T}}$$

i.e., $B^*_{p+1,T}$ is the block vector formed by the first p elements of $B_{p+1,T}$.

Thus we can rewrite (26) into

$$\begin{aligned} R_{p,T}\, B^*_{p+1,T} + x &= 0 \\ x^{\mathsf{T}} B^*_{p+1,T} + z &= R^r_{p+1,T} \end{aligned}$$

From the first equation above we have

$$x^{\mathsf{T}} = -B^*{}_{p+1,T}^{\mathsf{T}}\ R_{p,T}$$

and recalling from (9b) that

$$C_{p,T} = R^{-1}_{p,T}\ Y_{[T:T-p+1]}$$

equation (25) becomes

$$\begin{aligned} \delta C_{p,T} &= x^{\mathsf{T}}\ C_{p,T} \\ &= -B^*{}_{p+1,T}^{\mathsf{T}}\ Y_{[T:T-p+1]} \\ &= \hat{y}_{T-p-1|[T-p:T]}\ . \end{aligned}$$

From the observation from (23) that the last block-row of $C_{p+1,T}$ is equal to $R^{-r}_{p+1,T}\ r_{p+1,T}$ and also that the last block-row of $B_{p+1,T}$ is I, we can obtain the order update recursion for $C_{p,T}$

$$C_{p+1,T} = \begin{bmatrix} C_{p,T} \\ 0 \end{bmatrix} + B_{p+1,T}\, R^{-r}_{p+1,T}\ r_{p+1,T}\ . \tag{27}$$

Also along the same lines an alternate update for $C_{p,T}$ is obtained as

$$C_{p+1,T} = \begin{bmatrix} 0 \\ C_{p,T-1} \end{bmatrix} + A_{p+1,T} R^{-\epsilon}{}_{p+1,T} \epsilon_{p+1,T} \quad . \tag{28}$$

Thus equations (19) – (22) , (26), (28) give a set of order update recursions for $A_{p,T}$, $B_{p,T}$ and $C_{p,T}$. These recursions for $A_{p+1,T}$ and $B_{p+1,T}$ are similar to the multichannel version of the Levinson algorithm [WR], [MVLK], and [Rob]. However, the recursions for $C_{p+1,T}$ are new results.

2.2.3 Time Update Recursions

Using the relation (5), we obtain

$$R_{p,T+1} A_{p,T} = \begin{bmatrix} R^{\epsilon}{}_{p,T} \\ 0 \\ . \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} y_{T+1} \\ y_T \\ . \\ y_{T-p} \\ y_{T-p+1} \end{bmatrix} \epsilon^{T}{}_{p,T}(T+1) \quad . \tag{29}$$

We then apply relation (4) to [0 , $C^{T}{}_{p-1,T}$]^{T} so that after post-multiplying the result by $\epsilon^{T}{}_{p,T}(T+1)$, we can force the right hand side of (29) to satisfy the normal equation (12). After some algebra, the time update recursion for $A_{p,T}$ is obtained as follows

$$A_{p,T+1} = A_{p,T} - \begin{bmatrix} 0 \\ C_{p-1,T} \end{bmatrix} \epsilon^{T}{}_{p,T}(T+1) \quad . \tag{30}$$

Furthermore, we premultiply (29) by [$y^{T}{}_{T+1}$, ... , $y^{T}{}_{T+1-p}$] and obtain

$$\epsilon^{T}{}_{p,T+1} = \epsilon^{T}{}_{p,T}(T+1) - \gamma_{p-1,T} \, \epsilon^{T}{}_{p,T}(T+1)$$

and recalling that $\gamma_{p-1,T}$ is a scalar process, we rearrange terms to get

$$\epsilon_{p,T+1} = \epsilon_{p,T}(T+1) \, (1 - \gamma_{p-1,T}) \quad . \tag{31}$$

Thus the time update recursion for $R^{\epsilon}{}_{p,T}$ is obtained as follows

$$R^{\epsilon}{}_{p,T+1} = R^{\epsilon}{}_{p,T} + \frac{\epsilon_{p,T+1}\,\epsilon^{T}{}_{p,T+1}}{1-\gamma_{p-1,T}} . \tag{32}$$

A similar set of time update recursions for $B_{p,T}$ and $R^{r}{}_{p,T}$ are obtained as

$$B_{p,T+1} = B_{p,T} - \begin{bmatrix} C_{p-1,T+1} \\ 0 \end{bmatrix} \frac{r^{T}{}_{p,T+1}}{1-\gamma_{p-1,T+1}} \tag{33}$$

$$R^{r}{}_{p,T+1} = R^{r}{}_{p,T} + \frac{r_{p,T+1}\,r^{T}{}_{p,T+1}}{1-\gamma_{p-1,T+1}} . \tag{34}$$

and for $C_{p-1,T}$ and $\gamma_{p,T}$ via

$$C_{p,T+1} = \begin{bmatrix} 0 \\ C_{p-1,T} \end{bmatrix} + A_{p,T+1}\,R^{-\epsilon}{}_{p,T+1}\,\epsilon_{p,T+1} , \tag{35}$$

$$\begin{aligned}\gamma_{p+1,T} &= \gamma_{p-1,T} + \epsilon^{T}{}_{p,T+1}\,R^{-\epsilon}{}_{p,T+1}\,\epsilon_{p,T+1} \\ &= \gamma_{p,T} + \epsilon^{T}{}_{p,T+1}\,R^{-\epsilon}{}_{p,T+1}\,\epsilon_{p,T+1} - r^{T}{}_{p,T}\,R^{-r}{}_{p,T}\,r_{p,T} .\end{aligned} \tag{36}$$

Equations (19) – (22), (26) – (28), (30), and (32) – (34) form a complete set of order and time update recursions for $A_{p,T}$, $B_{p,T}$ and $C_{p,T}$.

By using the same techniques, a time update recursion for $\Delta_{p+1,T}$, which will be useful in the ladder form implementation, is obtained as follows

$$\Delta_{p+1,T+1} = \Delta_{p+1,T} + \frac{r_{p,T}\,\epsilon^{T}{}_{p,T+1}}{1-\gamma_{p-1,T}} . \tag{37}$$

It is clear now that the time update of $\Delta_{p+1,T}$ is in fact a time-average of the cross-correlations between $r_{p,T}$ and $\epsilon_{p,T+1}$, except for the special gain factor $\frac{1}{1-\gamma_{p-1,T}}$. The significance of this gain factor is explained next.

2.2.4 Likelihood Variable

In this section we establish the significance of the variable $\gamma_{p,T}$ as a likelihood variable. Consider the Gaussian case where the joint distribution for $\{ y_T, y_{T-1}, \ldots, y_{T-p} \}$ is given by

$$p(y_T, \ldots, y_{T-p}) = |2\pi R_p|^{-1/2} \, exp \{ - \frac{1}{2} \, y^{\mathsf{T}}[T{:}T-p] R_p^{-1} y[T{:}T-p] \}. \tag{38}$$

It can be shown that $|R_p|$ is related to $\{ |R^{\epsilon}_i| , \, i = 0, \ldots, p \}$ which in turn are related to $\{ K_i , \, i = 1, \ldots, p \}$ (see [MG], for example) by

$$|R_p| = |R^{\epsilon}_0| \cdot \cdot \cdot \cdot \cdot \cdot |R^{\epsilon}_p| , \tag{39}$$

$$|R^{\epsilon}_{i+1}| = |R^{\epsilon}_i| (1 - |K_i|^2) . \tag{40}$$

Therefore the logarithm of (38), becomes a log-likelihood function

$$\begin{aligned} ll &= \ln |R_p| + ||y||^2{}_{R^{-1}_p} \\ &= \sum_{i=0}^{p} \ln |R^{\epsilon}_i| + \gamma_p \\ &= \ln |R_0| + \sum_{i=1}^{p} \ln (1 - |K_i|^2) + \gamma_p . \end{aligned} \tag{41}$$

We can indentify the variable $\gamma_{p,T}$ obtained from our exact least-squares recursions as the γ_p appearing in the log-likelihood function. Thus the $\gamma_{p,T}$ factor acts as a good detector for non-Gaussian components in the observations. Our simulation results indeed demonstrated that $\gamma_{p,T}$ would take high values (close to 1) at non-Gaussian components. It therefore also acts as an optimal gain factor in that the gain $\frac{1}{1 - \gamma_{p,T}}$ can adjust the gains immediately when non-Gaussian components are present in the observations. Simulation results are shown in a later section of this report.

2.2.5 Exact Least-Squares Ladder Recurrsions

Premultiplying (30), (33), and (26) by $[y^T{}_T , \ldots , y^T{}_{T-p+1}]$ we obtain the following order update recursions for $\epsilon_{p,T}$, $r_{p,T}$ and $\gamma_{p,T}$:

$$\epsilon_{p+1,T} = \epsilon_{p,T} - K^r{}_{p+1,T}\, r_{p,T-1} \tag{42}$$

$$r_{p+1,T} = r_{p,T-1} - K^\epsilon{}_{p+1,T}\, \epsilon_{p,T} \tag{43}$$

$$\gamma_{p+1,T} = \gamma_{p,T} + r^T{}_{p+1,T}\, R^{-r}{}_{p+1,T}\, r_{p+1,T} \ , \tag{44}$$

where $K^\epsilon{}_{p+1,T}$ and $K^r{}_{p+1,T}$ are the *reflection or PARCOR coefficients* given by

$$K^\epsilon{}_{p+1,T} = \Delta_{p+1,T}\ R^{-\epsilon}{}_{p,T} \tag{45}$$

$$K^r{}_{p+1,T} = \Delta_{p+1,T}\ R^{-r}{}_{p,T-1} \tag{46}$$

$$\Delta_{p+1,T+1} = \Delta_{p+1,T} + \frac{r_{p,T}\ \epsilon^T{}_{p,T+1}}{1 - \gamma_{p-1,T}} \ . \tag{47}$$

The initial conditions are given by

$$\epsilon_{0,T} = r_{0,T} = y_T \ ; \qquad \gamma_{-1,T} = 0 \ ;$$

$$R^\epsilon{}_{0,T} = R^r{}_{0,T} = \sum_{t=0}^{T} y_t\, y_t{}^T$$

$$= R^r{}_{0,T-1} + y_T\, y^T{}_T \ ;$$

for $p \geq T$:

$$\epsilon_{p,T} = \epsilon_{T,T} \ ; \quad r_{p,T} = r_{T,T} \ ; \quad \gamma_{p,T} = \gamma_{T,T} \ ;$$

$$R^\epsilon{}_{p,T} = R^\epsilon{}_{T,T} \ ; \quad R^r{}_{p,T} = R^r{}_{T,T} \ ;$$

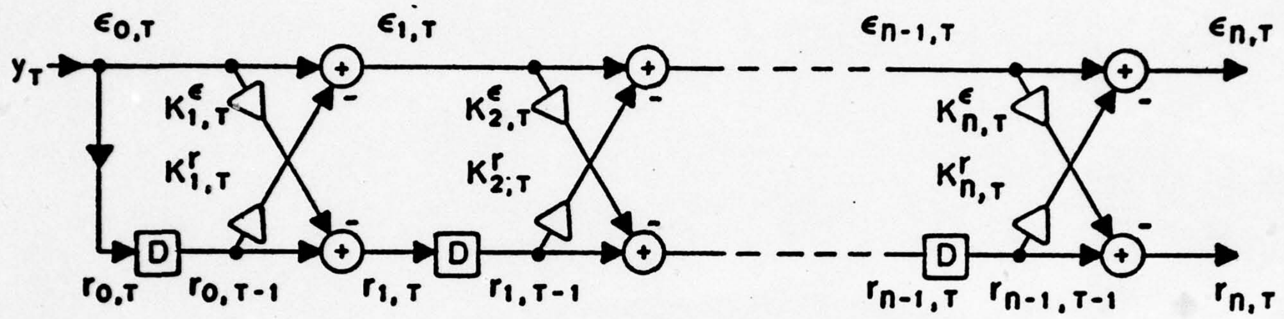
$$\Delta_{p+1,T} = 0 \ ;$$

$$\Delta_{p+1,p+1} = y_0\ \epsilon^T{}_{p+1,p+1} . \tag{48}$$

The recursions (45) – (47) compute the *sample cross-covariance* of the *forward and backward innovations*, using the *optimal weighting* $1 / (1 - \gamma_{\cdot,\cdot})$, in contrast to other suboptimal schemes [SV].

As the dual to the stochastic forms in [IS], [Wak], [Mo], [SKM], equations (42) – (47) are a complete set of *order and time update recursions* to obtain the *exact least-squares* ladder form predictor, which is shown in Figure 1.

Figure 1. Ladder realization of exact one-step least-squares predictor.



2.2.6 Invertibility of $R_{p,T}$.

Here we establish a criterion to guarantee that $R_{p,T}$ is invertible. From relation (5) we have

$$\begin{aligned} R_{p,T} &= R_{p,T-1} + y_T\, y_T^{\mathsf{T}} \\ y_T^{\mathsf{T}} &= [\, y_T^{\mathsf{T}}, \ldots, y_{T-p}^{\mathsf{T}}\,]. \end{aligned} \tag{49}$$

So, using the well-known matrix inversion lemma, we have

$$R^{-1}_{p,T} = R^{-1}_{p,T-1} - R^{-1}_{p,T-1}\, y_T\, [\, I + y_T^{\mathsf{T}} R^{-1}_{p,T-1}\, y_T\,]^{-1}\, y_T^{\mathsf{T}}\, R^{-1}_{p,T-1} \tag{50}$$

pre- and post-multiplying the above by y_T^{T} and y_T, we have

$$\gamma_{p,T} = \alpha_{p,T} - \frac{\alpha^2_{p,T}}{1+\alpha_{p,T}} = \frac{\alpha_{p,T}}{1+\alpha_{p,T}}\,, \tag{51}$$

where

$$\alpha_{p,T} = y_T^{\mathsf{T}}\; R^{-1}_{p,T-1}\; y_T \;>\; 0 \tag{52}$$

or

$$0 < \gamma_{p,T} = \frac{\alpha_{p,T}}{1+\alpha_{p,T}} < 1\,. \tag{53}$$

Thus when $\gamma_{p-1,T} = 1$, recursion will stop indicating that $R_{p,T}$ is not invertible, or equivalently that the columns of $Y_{p,T}$ are linearly dependent. However,in the scalar case $R_{p,T} > 0$ if $y_0 \neq 0$ and $R_{p,T}$ is always invertible. If $m > 1$, ($m = dim(y_t)$) we require $T \geq p + m$. These singularities can be avoided by including a priori estimates of the covariance R_p, or equivalently including a weighted norm of the predictor A_p in the error criterion $\xi_{p,T}$. Several such modifications have been proven useful in actual implementations. See also the use of the special quantity γ in our pitch detection algorithm on Section 2.7 and 5.4.

2.2.7 Maximum Likelihood Estimate

In this section we show the close connection between the *pre-windowed* method and *recursive maximum likelihood* method for autoregressive models.

Suppose we have a p-*th* order stationary Gaussian autoregressive process (for simplicity we consider the scalar case here)

$$y_t + A_p^{(1)} y_{t-1} + \ . \ . \ . + A_p^{(p)} y_{t-p} = \epsilon_t , \tag{54}$$

where $\{ \epsilon_t \}$ is an independent identically distributed zero-mean Gaussian random process with variances σ^2. Given the observations $\mathcal{Y}_T = [y_0, \ . \ . \ . \ , y_T]^{\mathsf{T}}$, from this process, the likelihood function is given as [BJ]

$$\Lambda = (2\pi)^{-(T+1)/2} \ | R_T |^{-1/2} \ exp \{ - \frac{y^{\mathsf{T}}{}_T R_T{}^{\mathsf{T}} \ y_T}{2} \} , \tag{55}$$

with

$$R_T = E \quad \mathcal{Y}_T \mathcal{Y}_T{}^{\mathsf{T}}. \tag{56}$$

Conditioning on $\{ y_0, \ . \ . \ . \ , y_{p-1} \}$, we express Λ as

$$\Lambda = (2\pi\sigma^2))^{-(T+1)/2} \ | M_p |^{1/2} \ exp\{ - \frac{\sigma^2}{2} (\sum_{t=p}^{T} \epsilon_t{}^2 + \mathcal{Y}[0, p-1]^{\mathsf{T}} M_p \ \mathcal{Y}[0, n-1])\}, \tag{57}$$

where

$$(M_p{}^{-1})_{i,j} = E (y_{t+i} y_t) / \sigma^2 \quad , \quad 0 \le i, \ j, \ \le p \ . \tag{58}$$

Collecting all the terms in observations into a matrix Σ_p, we have

$$\Lambda = (2\pi\sigma^2)^{-(T+1)/2} \ | M_p |^{1/2} \ exp \{ - \frac{\sigma^2}{2} (\mathbf{a}_p{}^{\mathsf{T}} \ \Sigma_p \ \mathbf{a}_p) \} \tag{59}$$

$$\mathbf{a}_p = [1, \ A_p^{(1)}, \ A_p^{(2)}, \ . \ . \ . \ . \ , \ A_p^{(p)} \ ]^{\mathsf{T}}$$

Then it can be shown that Σ_p is related to the *pre-windowed* covariance matrix as defined in (4) as follows

$$\Sigma_p = Y_{p,T} Y^{\mathsf{T}}{}_{p,T} - L_p L_p^{\mathsf{T}} \tag{60}$$

where

$$L_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & y_0 & 0 & 0 \\ 0 & \cdot & y_0 & 0 \\ 0 & y_{p-1} & \cdot & y_0 \end{bmatrix} \tag{61}$$

Moreover the maximum likelihood estimates for $\mathbf{a}_p$ is obtained by setting the derivative of Λ with respect to $\mathbf{a}_p$ to zero, and neglecting the data record length independent terms (justifications of such approximation can be found in [BJ]), we have the following *normal equation* for such estimates

$$\Sigma_p A_p = [\, \hat{\sigma}_p^2, 0, \ldots, 0 \,]^{\mathsf{T}} \tag{62}$$

$$A_p = [\, 1, \hat{a}_p^{(1)}, \hat{a}_p^{(2)}, \ldots, \hat{a}_p^{(p)} \,]^{\mathsf{T}} \tag{63}$$

$$\hat{\sigma}^2{}_p = A_p^{\mathsf{T}} \Sigma_p A_p \tag{64}$$

We can see now that the *Pre-windowed* method is a good approximation to *recursive maximum likelihood* (RML) estimates, both L_p and $| M_p |$ are independent of data record length, and asymptotic properties of the maximum likelihood estimates are preserved, when $T >> p$.

2.2.8 Summary of Algorithm

```
                    initialization
          $\epsilon_{0,0} \leftarrow r_{0,0} \leftarrow 0$;
          $R^{\epsilon}_{0,0} \leftarrow R^{r}_{0,0} \leftarrow y_0\, y_0$;
      for $i \leftarrow 1$ upto $pmax$ do $\Delta_{i,i} \leftarrow 0$;
                    main loop
for $T \leftarrow 1$ upto $Nsamples$ do
 begin   $\epsilon_{0,T} \leftarrow r_{0,T} \leftarrow y_T$;   $\gamma_{-1,T-1} \leftarrow 0$;
         $R^{\epsilon}_{0,T} \leftarrow R^{r}_{0,T} \leftarrow R^{\epsilon}_{0,T-1} + y_T\, y_T$;
         Orderupdate;
 end;
 procedure Orderupdate;
   for $p \leftarrow 0$ upto ( $pmax$ min $T$ ) do
    begin  $\Delta_{p+1,T} \leftarrow \Delta_{p+1,T-1} + r_{p,T-1}\, \epsilon_{p,T} / (1 - \gamma_{p-1,T-1})$;
           $\gamma_{p,T} \leftarrow \gamma_{p-1,T} + r_{p,T}\, r_{p,T} / R^{r}_{p,T}$;
           $K^{\epsilon}_{p+1,T} \leftarrow \Delta_{p+1,T} / R^{\epsilon}_{p,T}$;
           $K^{r}_{p+1,T} \leftarrow \Delta_{p+1,T} / R^{r}_{p,T-1}$;
           $\epsilon_{p+1,T} \leftarrow \epsilon_{p,T} - K^{r}_{p+1,T}\, r_{p,T-1}$;
           $r_{p+1,T} \leftarrow r_{p,T-1} - K^{\epsilon}_{p+1,T}\, \epsilon_{p,T}$;
     if $T \le pmax$ then begin
           $R^{\epsilon}_{p+1,T} \leftarrow R^{\epsilon}_{p,T} - K^{r}_{p+1,T}\, \Delta_{p+1,T}$;
           $R^{r}_{p+1,T} \leftarrow R^{r}_{p,T-1} - K^{\epsilon}_{p+1,T}\, \Delta_{p+1,T}$;
        end
       else begin
           $R^{\epsilon}_{p+1,T} \leftarrow R^{\epsilon}_{p+1,T-1} + \epsilon_{p,T}\, \epsilon_{p,T} / (1 - \gamma_{p-1,T-1})$;
           $R^{r}_{p+1,T} \leftarrow R^{r}_{p+1,T-1} + r_{p,T}\, r_{p,T} / (1 - \gamma_{p-1,T-1})$;
        end;
   end;
```

2.3 Tracking Time-Varying Parameters

So far we have assumed that our observations $\{ y(t), 0 \le t \le T \}$, are generated from a constant parameter model. When the parameters are changing with time, the general algorithms must be modified to track these changes. Using time update recursions of the algorithms, we apply a simple approach to the tracking problem by including an exponential weighting factor, w, or the so-called fading memory factor, into the error criteria.

2.3.1 Exponentially Weighted Algorithms

We define the squared error criterion for the pre-windowed case ($s=0$, $f=T$) as

$$\xi_{p,T} = \sum_{t=0}^{T} w^{T-t} \; \epsilon^{T}{}_{p,T}(t) \; \epsilon_{p,T}(t) \quad , \tag{1}$$

where w is a constant ≤ 1, (e.g. $w = 0.99$) so that the past prediction errors, being weighted with w, will have smaller influence on the estimates. Other weighting schemes such as using time-varying weights, $w(t)$, are more complicated (see Ljung[Lju], for example). For simplicity, we only consider the case of constant w.

We introduce a simple procedure to obtain the ladder form for this exponentially weighted case by first considering the simple time-weighted case. The *Normal Equation* is weighted with respect to the length of observations, i.e. the time index,

$$R^{*}{}_{p,T} \, A_{p,T} = [\, R^{\epsilon *}{}_{p,T} \, , 0 \, , 0 \, , \ldots , 0 \,]^{T} , \tag{2}$$

where

$$R^{*}{}_{p,T} = \frac{1}{T+1} \, R_{p,T} \quad , \tag{3}$$

$$R^{\epsilon *}{}_{p,T} = \frac{1}{T+1} \, R^{\epsilon}{}_{p,T} \quad . \tag{4}$$

This time-weighted *Normal Equation* (2) retains the same form as the non-weighted one, and the least-squares predictor for the two cases must be identical.

Thus both the forward and backward predictors would remain the same while only the auxiliary quantity $C_{p,T}$ needs to be modified to

$$C^*_{p,T} = (T+1)\, C_{p,T} \quad , \tag{5}$$

and the defining relation for them also retains the same form

$$R^*_{p,T}\,[\, A_{p,T} \,,\; B_{p,T} \,,\; C^*_{p,T} \,] = \left[\begin{array}{c|c|c} R^{\epsilon *}_{p,T} & 0 & y_T \\ 0 & 0 & y_{T-1} \\ . & . & . \\ 0 & 0 & y_{T-p+1} \\ 0 & R^{r*}_{p,T} & y_{T-p} \end{array}\right] , \tag{6}$$

with $R^{r*}_{p,T}$ defined by $\frac{1}{T+1}\, R^r_{p,T}$.

Similarly, the forward innovations, $\epsilon_{p,T}$, and backward residuals, $r_{p,T}$, remain unchanged, while $\gamma_{p,T}$, the auxiliary quantity, is modified to

$$\gamma^*_{p,T} = (T+1)\,\gamma_{p,T} \,. \tag{7}$$

The defining relations for them again retain the same form

$$\begin{bmatrix} \epsilon_{p,T} \\ r_{p,T} \\ \gamma^*_{p,T} \end{bmatrix} = \begin{bmatrix} A^{\mathsf{T}}_{p,T} \\ B^{\mathsf{T}}_{p,T} \\ C^{*T}_{p,T} \end{bmatrix} \begin{bmatrix} y_T \\ y_{T-1} \\ . \\ y_{T-p+1} \\ y_{T-p} \end{bmatrix} . \tag{8}$$

We next consider the time-update recursion identities for the matrix $R^*_{p,T}$ which is given by

$$R^*_{p,T} = \frac{T}{T+1}\, R^*_{p,T-1} + \frac{1}{T+1} \begin{bmatrix} y_T \\ . \\ y_{T-p} \end{bmatrix} [\, y^{\mathsf{T}}_T \,, \; . \; . \; . \; . \;,\; y^{\mathsf{T}}_{T-p} \,] \tag{9}$$

Thus if T of the weighting factor is set to any constant T_0, we can interpret the ratio $\frac{T_0}{T_0+1}$, which is ≤ 1, to be the exponential weighting factor, w, and this T_0

can also be interpreted as the *a priori* time constant for the underlying time-varying model.

Recursion (9) can be rewritten into

$$R^*_{p,T} = R^*_{p,T-1} + \{ \begin{bmatrix} y_T \\ \cdot \\ y_{T-p} \end{bmatrix} [y^T_T, \; . \; . \; , y^T_{T-p}] - R^*_{p,T-1} \} / (T + 1), \tag{10}$$

and we can identify $\frac{1}{T+1}$ as some time-varying parameter, $\lambda(T)$, that is related to a time-varying weighting factor, $w(T)$. Indeed, it can be easily shown that $w(t)$ and $\lambda(t)$ are related by

$$\lambda(t+1) = \frac{\lambda(t)}{w(t+1) + \lambda(t)}; \qquad \lambda(0) = 1 . \tag{11}$$

The order-update recursion for $R^*_{p,T}$ retains the same form as the non-weighted case and is given by

$$R^*_{p,T} = \begin{bmatrix} R^*_{p-1,T} & & x \\ x & x & x \end{bmatrix}, \tag{12}$$

while the time-shifted order update is modified to

$$R^*_{p,T} = \begin{bmatrix} x & x & x \\ x & \frac{T}{T+1} R^*_{p-1,T-1} & \end{bmatrix}. \tag{13}$$

2.3.2 Ladder Recursions

Having set up the time, order, and time-shifted order update recursions for $R^*_{p,T}$ in (10), (12) and (13), various recursions for other quantities can be obtained in the same fashion as described in Task Report I, therefore we will only give the important results and also drop the superscript "*" from here on so that all quantities are defined in the time-weighted context.

Again we obtain $\Delta_{p+1,T}$, $\Gamma_{p+1,T}$ by the following

$$R_{p+1,T} \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{p,T} & & x \\ x & x & x \end{bmatrix} \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix} = \begin{bmatrix} R^{\epsilon}_{p,T} \\ 0 \\ 0 \\ 0 \\ \Delta_{p+1,T} \end{bmatrix} , \tag{14}$$

where

$$\Delta_{p+1,T} = [\,\mathit{last\ block\text{-}row\ of}\ R_{p+1,T}\,] \begin{bmatrix} A_{p,T} \\ 0 \end{bmatrix}$$

$$= \sum_{t=0}^{T} y_{t-p-1}\ \epsilon^{T}_{p,T}(t) \quad . \tag{15}$$

and similarly for $\Gamma_{p+1,T}$

$$\Gamma_{p+1,T} = [\,\mathit{first\ block\text{-}row\ of}\ R_{p+1,T}\,] \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix}$$

$$= \sum_{t=1}^{T} y_t\ r^{T}_{p,T-1}(t-1) \ . \tag{16}$$

with

$$\Delta_{p+1,T} = \Gamma^{T}_{p+1,T} \tag{17}$$

as given by the so-called Burg-type lemma.

The forward and backward *reflection or PARCOR coefficients* are given by

$$K^{\epsilon}_{p+1,T} = \Delta_{p+1,T}\ R^{-\epsilon}_{p,T} \quad , \tag{18}$$

$$K^{r}_{p+1,T} = \Delta^{T}_{p+1,T}\ R^{-r}_{p,T-1}\ \{\frac{T+1}{T}\} . \tag{19}$$

Notice the $\{\frac{T+1}{T}\}$ factor in the $K^r{}_{p+1,T}$ due to the shifted time index of $R^r{}_{p,T-1}$.

The ladder recursions are given by

$$\epsilon_{p+1,T} = \epsilon_{p,T} - K^r{}_{p+1,T}\, r_{p,T-1} \qquad (20)$$

$$r_{p+1,T} = r_{p,T-1} - K^\epsilon{}_{p+1,T}\, \epsilon_{p,T} \qquad (21)$$

$$\gamma_{p+1,T} = \gamma_{p,T} + r^\top{}_{p+1,T}\, R^{-r}{}_{p+1,T}\, r_{p+1,T} \ , \qquad (22)$$

$$R^\epsilon{}_{p+1,T} = R^\epsilon{}_{p,T} - \Delta^\top{}_{p+1,T}\, R^{-r}{}_{p,T-1}\, \{\frac{T+1}{T}\}\, \Delta_{p+1,T} \ . \qquad (23)$$

$$R^r{}_{p+1,T} = R^r{}_{p,T-1}\, \{\frac{T}{T+1}\} - \Delta_{p+1,T}\, R^{-\epsilon}{}_{p,T}\, \Delta^\top{}_{p+1,T} \ . \qquad (24)$$

The time update for $\Delta_{p,T}$ requires some similar algebraic manipulations and is given by

$$\Delta_{p+1,T+1} = \Delta_{p+1,T} + [\ \frac{r_{p,T}\ \epsilon^\top{}_{p,T+1}}{1-\gamma_{p-1,T}/(T+1)} - \Delta_{p+1,T}\ ]\{\frac{1}{T+1}\} \ . \qquad (25)$$

Initial Conditions

The initial conditions remain the same as the unweighted case, and a time constant is needed to set up the desired weighting factor.

$$\epsilon_{0,T} = r_{0,T} = y_T \ ; \qquad \gamma_{-1,T} = 0 \ ;$$

$$R^\epsilon{}_{0,T} = R^r{}_{0,T} = \frac{1}{T+1}\sum_{t=0}^{T} y_t\, y_t^\top$$

$$= R^r{}_{0,T-1} + [\, y_T\, y^\top{}_T - R^r{}_{0,T-1}\,]\ \{\frac{1}{T+1}\} ;$$

for $p \geq T$:

$$\epsilon_{p,T} = \epsilon_{T,T} ; \quad r_{p,T} = r_{T,T} ; \quad \gamma_{p,T} = \gamma_{T,T} ;$$

$$R^\epsilon{}_{p,T} = R^\epsilon{}_{T,T} ; \quad R^r{}_{p,T} = R^r{}_{T,T} ;$$

$$\Delta_{p+1,T} = 0 ;$$

$$\Delta_{p+1,p+1} = y_0\ \epsilon^\top{}_{p+1,p+1} .$$

(26)

Other aspects of the ladder form like invertibility of $R_{p,T}$ *remain unchanged* from the unweighted case.

2.3.3 Summary of Algorithm

```
                         initialization

              $\epsilon_{0,0}$ ← $r_{0,0}$ ← 0 ;
              $R^{\epsilon}_{0,0}$ ← $R^{r}_{0,0}$ ← $y_0\, y_0$ ;
        for $i$ ← 1 upto $pmax$ do
              $\Delta_{i,i}$ ← 0 ;
                         main loop
for  $T$ ← 1  upto  $Nsamples$  do
  begin        $\epsilon_{0,T}$ ← $r_{0,T}$ ← $y_T$ ;

    if $tracking$ = true then
      begin    $ttau$ ← $T$ min $tau$;
               $Ti$ ← 1 / $ttau$ ;
               $T1i$ ← 1 / ($ttau$ + 1) ;
               $T0T1$ ← $ttau$ / ($ttau$ + 1) ;
               $T1T0$ ← 1 / $T0T1$ ;
      end
   else
      begin    $Ti$ ← 1 / $T$ ;
               $T1i$ ← 1 / ($T$ + 1) ;
               $T0T1$ ← $T$ / ($T$ + 1) ;
               $T1T0$ ← 1 / $T0T1$ ;
      end;
           $\gamma_{-1,T-1}$ ← 0 ;
           $R^{\epsilon}_{0,T}$ ← $R^{r}_{0,T}$
                  ← $R^{\epsilon}_{0,T-1}$ + $T1i$ [ $y_T\, y_T$ - $R^{\epsilon}_{0,T-1}$ ];
           $OrderUpdate$;
end;
```

```
procedure OrderUpdate;
  for $p \leftarrow 0$ upto ( $pmax$ min $T$ ) do
   begin
        $\Delta_{p+1,T} \leftarrow \Delta_{p+1,T-1} + T1i\ [r_{p,T-1}\, \epsilon_{p,T} / (1 - (Ti)\, \gamma_{p-1,T-1}) - \Delta_{p+1,T-1}]$ ;
        $\gamma_{p,T} \leftarrow \gamma_{p-1,T} + r_{p,T}\, r_{p,T} / R^{r}_{p,T}$;
        $K^{\epsilon}_{p+1,T} \leftarrow \Delta_{p+1,T} / R^{\epsilon}_{p,T}$ ;
        $K^{r}_{p+1,T} \leftarrow T1T0\ \Delta_{p+1,T} / R^{r}_{p,T-1}$ ;
        $\epsilon_{p+1,T} \leftarrow \epsilon_{p,T} - K^{r}_{p+1,T}\, r_{p,T-1}$ ;
        $r_{p+1,T} \leftarrow r_{p,T-1} - K^{\epsilon}_{p+1,T}\, \epsilon_{p,T}$ ;
   if $T \le pmax$ then begin
        $R^{\epsilon}_{p+1,T} \leftarrow R^{\epsilon}_{p,T} - K^{r}_{p+1,T}\, \Delta_{p+1,T}$ ;
        $R^{r}_{p+1,T} \leftarrow T0T1\ R^{r}_{p,T-1} - K^{\epsilon}_{p+1,T}\, \Delta_{p+1,T}$ ;
      end
     else begin
        $R^{\epsilon}_{p+1,T} \leftarrow R^{\epsilon}_{p+1,T-1} + T1i\ \epsilon_{p,T}\, \epsilon_{p,T} / (1 - (Ti)\gamma_{p-1,T-1})$ ;
        $R^{r}_{p+1,T} \leftarrow R^{r}_{p+1,T-1} + T1i\ r_{p,T}\, r_{p,T} / (1 - (Ti)\gamma_{p-1,T-1})$ ;
      end;
  end;
```

end of algorithm

2.4 The Covariance Ladder Form

As mentioned earlier, the covariance ladder form is even better suited for speech modeling than the pre-windowing form, since it does not require any windowing of the data. This is of importance when the analysis involves short data segments as is the case in speech modeling. Strictly speaking, the covariance or non-windowed form uses a rectangular window on the data, i.e. it uses only "the available data set". The weighted pre-windowing form uses an exponential window (on the error sequence) that acts like a *sliding window* of a covariance type. Therefore the behavior of the weighted pre-windowed form is expected to be similar to a fixed sliding window covariance form (a slightly more complex covariance type form). To derive the covariance form algorithms we use the notation introduced in Section 2.1 .

2.4.1 Basic Definitions

The covariance matrix, $R_{p,S,T}$, satisfies the following recursive identities:

$$R_{p,S,T} = R_{p,S,T-1} + \begin{bmatrix} y_T \\ \cdot \\ y_{T-p} \end{bmatrix} [y^T_T , \; . \; . \; . \; . \; , \; y^T_{T-p}] \tag{1}$$

$$R_{p,S,T} = R_{p,S+1,T} + \begin{bmatrix} y_{S+p} \\ \cdot \\ y_S \end{bmatrix} [y^T_{S+p} , \; . \; . \; . \; . \; , \; y^T_S] \tag{2}$$

$$R_{p,S,T} = \begin{bmatrix} x & x & x \\ & x & R_{p-1,S,T-1} \end{bmatrix} \tag{3}$$

$$R_{p,S,T} = \begin{bmatrix} R_{p-1,S+1,T} & & x \\ x & x & x \end{bmatrix} . \tag{4}$$

where the x's represent unspecified elements along the appropriate edges.

Define $A_{p,S,T}$, $B_{p,S,T}$, $C_{p,S,T}$ and $D_{p,S,T}$ by

$$R_{p,S,T}\,[A_{p,S,T},\ B_{p,S,T},\ C_{p,S,T},\ D_{p,S,T}] = \left[\begin{array}{c|c|c|c} R^{\epsilon}{}_{p,S,T} & 0 & y_T & y_{S+p} \\ 0 & 0 & y_{T-1} & y_{S+p-1} \\ . & . & . & . \\ 0 & 0 & y_{T-p+1} & y_{S+1} \\ 0 & R^{r}{}_{p,S,T} & y_{T-p} & y_S \end{array}\right] \qquad (5)$$

where $A_{p,S,T}$ and $B_{p,S,T}$ are respectively the forward and backward predictors, with $A_{p,S,T}$ and $B_{p,S,T}$ by

$$A^{T}{}_{p,S,T} = [\, I_m\, ,\ A_{p,S,T}{}^{(1)}\, ,\ .\ .\ .\ .\ ,\ B_{p,S,T}{}^{(p)}\ ]\ ,$$

$$B^{T}{}_{p,S,T} = [\, B_{p,S,T}{}^{(p)}\, ,\ .\ .\ .\ .\ ,\ B_{p,S,T}{}^{(1)}\, ,\ I_m\ ] \qquad (6)$$

and $C_{p,S,T}$ and $D_{p,S,T}$ are auxiliary quantities.

We define $\epsilon_{p,S,T}$, the forward prediction errors or innovations, and $r_{p,S,T}$, the backward prediction residuals, and auxiliary scalar quantities $g_{p,S,T}$ and $h_{p,S,T}$ by

$$\begin{bmatrix} \epsilon_{p,S,T} \\ r_{p,S,T} \\ g_{p,S,T} \\ h_{p,S,T} \end{bmatrix} = \begin{bmatrix} A^{T}{}_{p,S,T} \\ B^{T}{}_{p,S,T} \\ C^{T}{}_{p,S,T} \\ D^{T}{}_{p,S,T} \end{bmatrix} \begin{bmatrix} y_T \\ y_{T-1} \\ . \\ . \\ y_{T-p+1} \\ y_{T-p} \end{bmatrix} . \qquad (7)$$

and one more auxiliary scalar quantity $f_{p,T}$ by

$$f_{p,S,T} = D^{T}{}_{p,S,T}\ Y_{[\,S+p\,:\,S\,]}\ . \qquad (8)$$

The quantities $f_{p,S,T}$, $g_{p,S,T}$ and $h_{p,S,T}$ are related by

$$\begin{bmatrix} g_{p,S,T} & h_{p,S,T} \\ h_{p,S,T} & f_{p,S,T} \end{bmatrix} = \begin{bmatrix} y^T_T & y^T_{T-1} & \cdot & \cdot & y^T_{T-p} \\ y^T_{S+p} & y^T_{S+p-1} & \cdot & \cdot & y^T_S \end{bmatrix} R^{-1}_{p,S,T} \begin{bmatrix} y_T & y_{S+p} \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ y_{T-p+1} & y_{S+1} \\ y_{T-p} & y_S \end{bmatrix}$$

$$= \begin{bmatrix} y^T_T & y^T_{T-1} & \cdot & \cdot & y^T_{T-p} \\ y^T_{S+p} & y^T_{S+p-1} & \cdot & \cdot & y^T_S \end{bmatrix} [\ C_{p,S,T}\ ,\ D_{p,S,T}\ ]$$

$$\geq 0 \qquad (9)$$

We now let the basic observation record span $0 = S \leq t \leq T$, and drop subscript 0 when no confusion is created.

2.4.2 Order Update Recursions

We would like $A_{p+1,T}$ and $B_{p+1,T}$ to satisfy the normal equation

$$R_{p+1,T} [A_{p+1,T} , B_{p+1,T}] = \begin{bmatrix} R^{\epsilon}_{p+1,T} & \vert & 0 \\ 0 & \vert & 0 \\ 0 & \vert & R^{r}_{p+1,T} \end{bmatrix} \qquad (10)$$

and we start by using relation (4) :

$$R_{p+1,T} \begin{bmatrix} A_{p,1,T} \\ \\ 0 \end{bmatrix} = \begin{bmatrix} R_{p,1,T} & & x \\ & & \\ x & x & x \end{bmatrix} \begin{bmatrix} A_{p,1,T} \\ \\ 0 \end{bmatrix} = \begin{bmatrix} R^{\epsilon}_{p,1,T} \\ 0 \\ \Delta_{p+1,T} \end{bmatrix} , \qquad (11)$$

where

$$\Delta_{p+1,T} = [\textit{last block-row of } R_{p+1,T}] \begin{bmatrix} A_{p,1,T} \\ 0 \end{bmatrix} . \qquad (12)$$

Similarly, using the relation (3) :

$$R_{p+1,T} \begin{bmatrix} 0 \\ \\ B_{p,T-1} \end{bmatrix} = \begin{bmatrix} x & x & x \\ & & \\ x & R_{p,T-1} & \end{bmatrix} \begin{bmatrix} 0 \\ \\ B_{p,T-1} \end{bmatrix} = \begin{bmatrix} \Gamma_{p+1,T} \\ 0 \\ R^{r}_{p,T-1} \end{bmatrix} , \qquad (13)$$

where

$$\Gamma_{p+1,T} = [\textit{first block-row of } R_{p+1,T}] \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix} . \qquad (14)$$

Applying the so-called Burg-type lemma,

$$\begin{bmatrix} A^{T}_{p,1,T} & 0 \\ 0 & B^{T}_{p,T-1} \end{bmatrix} R_{p+1,T} \begin{bmatrix} A_{p,1,T} & 0 \\ 0 & B_{p,T-1} \end{bmatrix}$$

$$= \begin{bmatrix} A^{T}_{p,1,T} & 0 \\ & \\ 0 & B^{T}_{p,T-1} \end{bmatrix} \begin{bmatrix} R^{\epsilon}_{p,1,T} & \Gamma_{p+1,T} \\ 0 & 0 \\ \Delta_{p+1,T} & R^{r}_{p,T-1} \end{bmatrix}$$

$$= \begin{bmatrix} R^{\epsilon}_{p,T} & \Gamma_{p+1,T} \\ \Delta_{p+1,T} & R^{r}_{p,T-1} \end{bmatrix},$$

note that these expressions are all symmetric matrices. Therefore we get the important identity

$$\Delta_{p+1,T} = \Gamma^{\top}_{p+1,T} \tag{15}$$

Thus postmultiply (13) by $R^{-r}_{p,T-1}\,\Delta_{p+1,T}$, where R^{-r} is the inverse of R^{r}, and then subtract the result from (11), to obtain the following order update recursions for $A_{p,T}$ and $R^{\epsilon}_{p,T}$

$$A_{p+1,T} = \begin{bmatrix} A_{p,1,T} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix} R^{-r}_{p,T-1}\,\Delta_{p+1,T} \tag{16}$$

$$R^{\epsilon}_{p+1,T} = R^{\epsilon}_{p,1,T} - \Delta^{\top}_{p+1,T}\,R^{-r}_{p,T-1}\,\Delta_{p+1,T}\ . \tag{17}$$

A similar set of order update recursions for $B_{p,T}$ and $R^{r}_{p,T}$ is obtained as

$$B_{p+1,T} = \begin{bmatrix} 0 \\ B_{p,T-1} \end{bmatrix} - \begin{bmatrix} A_{p,1,T} \\ 0 \end{bmatrix} R^{-\epsilon}_{p,1,T}\,\Delta^{\top}_{p+1,T} \tag{18}$$

$$R^{r}_{p+1,T} = R^{r}_{p,T-1} - \Delta_{p+1,T}\,R^{-\epsilon}_{p,1,T}\,\Delta^{\top}_{p+1,T}\ . \tag{19}$$

To obtain the order update recursions for $C_{p,T}$, we first observe that the last block row of $R^{-1}_{p,T}$ is equal to $R^{-r}_{p,T}\,B^{\top}_{p,T}$. Thus from the definition of $C_{p,T}$, we can obtain the last block-row of $C_{p,T}$ i.e.

$$C_{p,T} = R^{-1}_{p,T} \begin{bmatrix} y_T \\ \cdot \\ y_{T-p} \end{bmatrix} \tag{20}$$

inplies

$$[\ \textit{last block-row of}\ C_{p,T}\ ] = R^{-r}_{p,T}\ {}^{r}_{p,T} \ . \tag{21}$$

Using the relation (4) , we have

$$R_{p+1,T}\begin{bmatrix} C_{p,1,T} \\ \\ 0 \end{bmatrix} = \begin{bmatrix} R_{p,1,T} & & x \\ & & x \\ x & x & x \end{bmatrix}\begin{bmatrix} C_{p,1,T} \\ \\ 0 \end{bmatrix} = \begin{bmatrix} y_T \\ y_{T-1} \\ \cdot\,\cdot \\ y_{T-p} \\ x \end{bmatrix} \tag{22}$$

and from the observation (21) that the last block-row of $C_{p+1,T}$ is equal to $R^{-r}{}_{p+1,T}\, r_{p+1,T}$ and also that the last block-row of $B_{p+1,T}$ is I_m, we can obtain the order update recursion for $C_{p,T}$ as

$$C_{p+1,T} = \begin{bmatrix} C_{p,1,T} \\ 0 \end{bmatrix} + B_{p+1,T}\, R^{-r}{}_{p+1,T}\, r_{p+1,T}(T) \quad . \tag{23}$$

Similarly, the order update recursion for $D_{p,T}$ is obtained

$$D_{p+1,T} = \begin{bmatrix} 0 \\ D_{p,T-1} \end{bmatrix} . + A_{p+1,T}\, R^{-\epsilon}{}_{p+1,T}\, \epsilon_{p+1,T}(p+1) \; . \tag{24}$$

Thus equations (16) - (19) , (23) and (24) give a set of order update recursions for $A_{p,T}$, $B_{p,T}$, $C_{p,T}$ and $D_{p,T}$. These recursions are very similar to the multichannel version of the Levinson algorithm ([WR], [MVLK], and [Rob]).

2.4.3 Time Update Recursions

Using the relation ($R1$), we obtain

$$R_{p,T+1}\ A_{p,T} = \begin{bmatrix} R^{\epsilon}{}_{p,T} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} y_{T+1} \\ \cdot \\ y_{T+1-p} \end{bmatrix} \epsilon^{T}{}_{p,T}(T+1) \ . \tag{25}$$

We then apply relation (3) to $[\ 0\ ,\ C^{T}{}_{p-1,T}\ ]^{T}$ so that after post-multiplying the result by $\epsilon^{T}{}_{p,T}(T+1)$, we can force the right hand side of (23) to satisfy the normal equation (10). After some algebra, the time update recursion for $A_{p,T}$ is obtained as follows

$$A_{p,T+1} = A_{p,T} - \begin{bmatrix} 0 \\ C_{p-1,T} \end{bmatrix} \epsilon^{T}{}_{p,T}(T+1) \ . \tag{26}$$

Furthermore, we premultiply (25) by $[\ y^{T}{}_{T+1}\ ,\ .\ .\ .\ .\ ,\ y^{T}{}_{T+1-p}\ ]$ and obtain

$$\epsilon^{T}{}_{p,T+1} = \epsilon^{T}{}_{p,T}(T+1) - g_{p-1,T}\ \epsilon^{T}{}_{p,T}(T+1) \tag{27}$$

and recalling that $g_{p-1,T}$ is a scalar, we rearrange terms to get

$$\epsilon_{p,T+1} = \epsilon_{p,T}(T+1)\ (\ 1 - g_{p-1,T}\ ) \ . \tag{28}$$

Thus the time update recursion for $R^{\epsilon}{}_{p,T}$ is obtained as follows

$$R^{\epsilon}{}_{p,T+1} = R^{\epsilon}{}_{p,T} + \frac{\epsilon_{p,T+1}\ \epsilon^{T}{}_{p,T+1}}{1 - g_{p-1,T}} \ . \tag{29}$$

A similar set of time update recursions for $B_{p,T}$ and $R^{r}{}_{p,T}$ is obtained as

$$B_{p,T+1} = B_{p,T} - \begin{bmatrix} C_{p-1,1,T+1} \\ 0 \end{bmatrix} \frac{r^{T}{}_{p,T+1}}{1-g_{p-1,1,T+1}} \tag{30}$$

$$R^{r}{}_{p,T+1} = R^{r}{}_{p,T} + \frac{r_{p,T+1}\ r^{T}{}_{p,T+1}}{1 - g_{p-1,1,T+1}} \ . \tag{31}$$

Equations (16) – (19), (23), (24) and (26) – (31) form a complete set of order and time update recursions for $A_{p,T}$, $B_{p,T}$, $C_{p,T}$ and $D_{p,T}$.

2.4.4 Time-Shifted Recursions and Scalar Updates

By using the same techniques for obtaining the order and time updates, we can also obtain the time-shifted updates of various quantities and updates for the scalar quantities. The time-shifted updates for $R^{\epsilon}{}_{p,T}$, $R^{r}{}_{p,T}$, $A_{p,T}$, $B_{p,T}$, $C_{p,T}$ and $D_{p,T}$ are given by:

$$R^{\epsilon}{}_{p,1,T} = R^{\epsilon}{}_{p,T} \left[I + \frac{R^{-\epsilon}{}_{p,T}\, \epsilon_{p,T}(p)\, \epsilon^{T}{}_{p,T}(p)}{1 - f_{p,T}} \right]^{-1} .$$

$$R^{r}{}_{p,T-1} = R^{r}{}_{p,T} \left[I + \frac{R^{-r}{}_{p,T}\, r_{p,T}(T)\, r^{T}{}_{p,T}(T)}{1 - g_{p,T}} \right]^{-1} .$$

$$A_{p,1,T} = \left[A_{p,T} + \frac{D_{p,T}\, \epsilon^{T}{}_{p,T}(p)}{1 - f_{p,T}} \right] R^{-\epsilon}{}_{p,T} R^{\epsilon}{}_{p,1,T}$$

$$B_{p,T-1} = \left[B_{p,T} + \frac{C_{p,T}\, r^{T}{}_{p,T}(T)}{1 - g_{p,T}} \right] R^{-r}{}_{p,T} R^{r}{}_{p,T-1}$$

$$C_{p,1,T} = C_{p,T} + D_{p,T} \frac{h_{p,T}}{1 - f_{p,T}}$$

$$D_{p,T-1} = D_{p,T} + C_{p,T} \frac{h_{p,T}}{1 - g_{p,T}}$$

The scalar updates for $f_{p,T}$, $g_{p,T}$ and $h_{p,T}$ are given by

$$\begin{aligned} f_{p+1,T} &= f_{p,T-1} + \epsilon^{T}{}_{p+1,T}(p+1)\, R^{-\epsilon}{}_{p+1,T}\, \epsilon_{p+1,T}(p+1) \\ &= f_{p,T} + \frac{h^{2}{}_{p,T}}{1-g_{p,T}} + \epsilon^{T}{}_{p+1,T}(p+1)\, R^{-\epsilon}{}_{p+1,T}\, \epsilon_{p+1,T}(p+1) \end{aligned}$$

$$\begin{aligned} g_{p+1,T} &= g_{p,1,T} + r^{T}{}_{p+1,T}(T)\, R^{-r}{}_{p+1,T}\, r_{p+1,T}(T) \\ &= g_{p,T} + \frac{h^{2}{}_{p,T}}{1-f_{p,T}} + r^{T}{}_{p+1,T}(T)\, R^{-r}{}_{p+1,T}\, r_{p+1,T}(T) \end{aligned}$$

$$\begin{aligned} h_{p+1,T} &= h_{p,1,T} + r^{T}{}_{p+1,T}(T)\, R^{-r}{}_{p+1,T}\, r_{p+1,T}(p+1) \\ &= h_{p,T-1} + \epsilon^{T}{}_{p+1,T}(p+1)\, R^{-\epsilon}{}_{p+1,T}\, \epsilon_{p+1,T}(T) . \end{aligned}$$

A time update recursion for $\Delta_{p+1,T}$, which will be useful in the ladder form implementation, can be verified to be given by

$$\Delta_{p+1,T+1} = \Delta_{p+1,T} + \frac{r_{p,T}\ \epsilon^{T}{}_{p,1,T+1}(T+1)}{1 - g_{p-1,1,T}} . \qquad (32)$$

2.4.5 Ladder Type Realization

Premultiplying (26), (30) , and (23) by $[\, y^{\mathsf{T}}{}_T \, , \; . \; . \; . \; , \; y^{\mathsf{T}}{}_{T-p+1} \,]$ we obtain the following order update recursions for $\epsilon_{p,T}$ and $r_{p,T}$.

$$\epsilon_{p+1,T}(T) = \epsilon_{p,1,T}(T) - K^{r}{}_{p+1,T}\, r_{p,T-1}(T-1) \qquad (33)$$

$$r_{p+1,T}(T) = r_{p,T-1}(T-1) - K^{\epsilon}{}_{p+1,T}\, \epsilon_{p,1,T}(T) \qquad (34)$$

$$\epsilon_{p,T}(p) = \epsilon_{p,T-1}(p) - \epsilon_{p,T}(T) \frac{h_{p-1,T-1}}{1-g_{p-1,T-1}}$$

(time update) (35)

$$\epsilon_{p,1,T}(T) = \epsilon_{p,T}(T) + \epsilon_{p,T}(p) \frac{h_{p-1,T-1}}{1-f_{p-1,T-1}} \qquad (36)$$

where $K^{\epsilon}{}_{p+1,T}$ and $K^{r}{}_{p+1,T}$ are the ***reflection or PARCOR coefficients*** given by

$$K^{\epsilon}{}_{p+1,T} = \Delta^{\mathsf{T}}{}_{p+1,T}\; R^{-\epsilon}{}_{p,1,T} \qquad (37)$$

$$K^{r}{}_{p+1,T} = \Delta_{p+1,T}\; R^{-r}{}_{p,T-1} \qquad (38)$$

The recursions (33) – (38) compute the ***sample cross-covariance*** of the ***forward and backward innovations.***

In contrast to the approximate schemes ([SV], [MG]) that have been presented to solve the stochastic ladder forms given in [IS], [Wak], [Mo], equations (33) – (38) form a complete set of ***order and time update recursions*** to obtain the ***exact least-squares*** ladder form predictor, which is shown in Figure 1. .

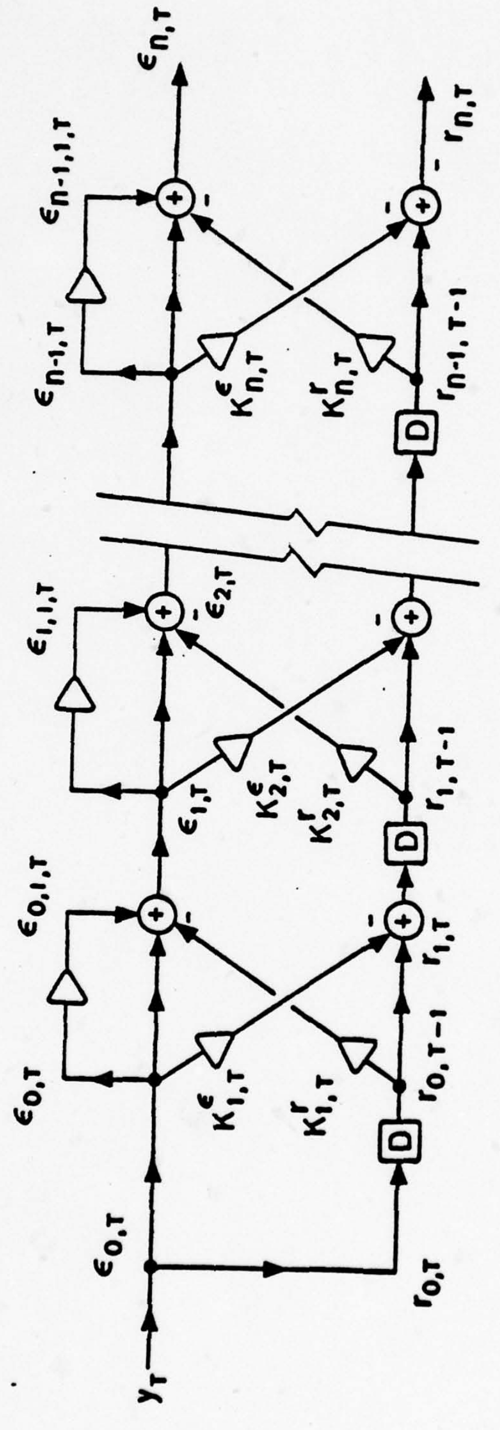


Figure 1. Ladder realization of exact one-step least-squares predictor.

2.4.6 Initial Conditions

In the covariance ladder realization, we may not start the recursion at $T = 0$ and just keep doing order and time updates. This is because for small values of T, $R_{p,0,T}$ is always singular. More specifically if $T < p(m+1) + m - 1$, where $m = dim\, y_t$, then $R_{p,0,T}$ will certainly be singular, for then $Y_{p,0,T}$ will be of low rank. To properly initialize the covariance ladder, we can wait until enough data is obtained, say $\{ y_0, \ldots, y_{T1} \}$, so that $R_{p,0,T1}$ is nonsingular for all orders p to be considered. Thus if $pmax$ is the maximum order of the ladder, we have only to check the nonsingularity of $R_{pmax,0,T1}$ since in this case $R_{p,0,T}$ will be nonsingular for $p \le pmax$, $T \ge T1$ as can be easily verified.

In practical situations (e.g. the scalar case), it is to be expected that $T1$ is close to $2pmax$. Now suppose we have determined $T1$ such that $R_{pmax,0,T1}$ is nonsingular. From $\{ y_0, \ldots, y_{T1} \}$ we recursively compute the last row of of $R_{p,0,T1}$ for $p = 0, \ldots, pmax$ and keep the first $pmax$ values $\{ y_0, \ldots, y_{pmax} \}$. Using the recursions given above, we can then compute $\Delta_{p,T1}$, $R^{\epsilon}{}_{p,0,T1}$, $R^{r}{}_{p,0,T1}$ and $\epsilon_{p,0,T1}(p)$ for $p = 0, \ldots, pmax$, which constitute the initial conditions for the time update of these quantities.

The initial conditions for order updating the ladder recursion are given by

$$\epsilon_{0,0,T} = \epsilon_{0,1,T} = r_{0,0,T} = y_T \tag{39}$$

$$f_{-1,0,T} = g_{-1,0,T} = h_{-1,0,T} = 0 \ . \tag{40}$$

There exist simpler alternatives for initializing the ladder form but they would require a more extensive explanation and justification; we therefore leave them for later publications. For example, we may initialize the various sections of the ladder in a recursive way, that is we initialize the first section at time $T1$ (in the scalar case $T1 = 0$), the second at time $T2$, and so on.

2.4.7 Invertibility of $R_{p,T}$

Here we establish a criterion to guarantee that $R_{p,T}$ is invertible. From relation ($R1$) we have

$$R_{p,T} = R_{p,T-1} + y\, y^{\top}$$
$$y^{\top} = [\, y_T^{\top}, \dots, y_{T-p}^{\top} \,] . \qquad (41)$$

So, using a well known matrix inversion lemma, we have

$$R^{-1}_{p,T} = R^{-1}_{p,T-1} - R^{-1}_{p,T-1}\, y\, [I + y^{\top} R^{-1}_{p,T-1}\, y\,]^{-1}\, y^{\top} R^{-1}_{p,T-1} . \qquad (42)$$

Pre- and post-multiplying this by $y^{\top}$ and y , we get

$$g_{p,T} = \alpha_{p,T} - \frac{\alpha^2_{p,T}}{1 + \alpha_{p,T}} , \qquad (43)$$

where

$$\alpha_{p,T} = y^{\top} R^{-1}_{p,T-1}\, y > 0 \qquad (44)$$

or

$$0 < g_{p,T} = \frac{\alpha_{p,T}}{1 + \alpha_{p,T}} < 1 . \qquad (45)$$

A similar argument shows that

$$0 < f_{p,T} < 1 . \qquad (46)$$

Thus we see that the invertibility is equivalent to the condition (45), (46) so that divisions by $1 - g_{p,T}$, and $1 - f_{p,T}$ can be carried out.

2.4.8 Summary of the Algorithm

```
initialization: ( See Text )
main loop
          for  $T \leftarrow T1$  upto  $Nsamples$  do
  begin  $\epsilon_{0,T} \leftarrow r_{0,T} \leftarrow y_T$ ;
         $f_{-1,T-1} \leftarrow g_{-1,T-1} \leftarrow h_{-1,T-1} \leftarrow 0$ ;
         $R^{\epsilon}_{0,T} \leftarrow R^{r}_{0,T} \leftarrow R^{\epsilon}_{0,T-1} + y_T y_T$;
         $R^{\epsilon}_{0,T} \leftarrow R^{r}_{0,T} \leftarrow R^{\epsilon}_{0,T-1} + y_T y_T$;
         $OrderUpdate$;
  end;
  Procedure  $OrderUpdate$;
    for $p \leftarrow 0$ upto $pmax$ do
    begin  $\epsilon_{p,T}(p) \leftarrow \epsilon_{p,T-1}(p) - \epsilon_{p,T} h_{p-1,T-1} / ( 1 - g_{p-1,T-1})$;
           $\epsilon_{p,1,T} \leftarrow \epsilon_{p,T} + \epsilon_{p,T}(p) h_{p-1,T-1} / ( 1 - f_{p-1,T-1})$;
           $\Delta_{p+1,T} \leftarrow \Delta_{p+1,T-1} + r_{p,T-1} \epsilon_{p,1,T} / ( 1 - g_{p-1,1,T-1} )$;
           $g_{p,1,T-1} \leftarrow g_{p,T-1} + h^2_{p,T-1} / ( 1 - f_{p,T-1} )$;
           $g_{p,T} \leftarrow g_{p-1,T-1} + \epsilon_{p,T} \epsilon_{p,T} / R^{\epsilon}_{p,T}$;
           $h_{p,T} \leftarrow h_{p-1,T-1} + \epsilon_{p,T}(p) \epsilon_{p,T} / R^{\epsilon}_{p,T}$;
           $f_{p,T} \leftarrow f_{p-1,T-1} + \epsilon_{p,T}(p) \epsilon_{p,T}(p) / R^{\epsilon}_{p,T}$;
           $R^{\epsilon}_{p,1,T} \leftarrow R^{\epsilon}_{p,T} - \epsilon_{p,T}(p)\epsilon_{p,T}(p) / ( 1 - f_{p-1,T-1})$;
           $K^{\epsilon}_{p+1,T} \leftarrow \Delta_{p+1,T} / R^{\epsilon}_{p,1,T}$;
           $K^{r}_{p+1,T} \leftarrow \Delta_{p+1,T} / R^{r}_{p,T-1}$;
           $\epsilon_{p+1,T} \leftarrow \epsilon_{p,1,T} - K^{r}_{p+1,T} r_{p,T-1}$;
           $r_{p+1,T} \leftarrow r_{p,T-1} - K^{\epsilon}_{p+1,T} \epsilon_{p,1,T}$;
           $R^{\epsilon}_{p+1,T} \leftarrow R^{\epsilon}_{p,1,T} - K^{r}_{p+1,T} \Delta_{p+1,T}$;
           $R^{r}_{p+1,T} \leftarrow R^{r}_{p,T-1} - K^{\epsilon}_{p+1,T} \Delta_{p+1,T}$;
    end;

      end of algorithm
```

2.5 Pole-Zero or Autoregressive Moving -Average(ARMA) Ladder Forms

In this section we present ladder forms for pole-zero or autoregressive moving-average (ARMA) models. Our approach here is to embed the underlying ARMA model into a two-channel AR model resulting in a joint innovations representation of the process. The two-channel predictor of this joint process may be implementated as a two-channel AR ladder.

2.5.1 Joint Innovations Representation of the ARMA Process

Given a pole-zero (ARMA) model of the following form

$$y_t + A_1 y_{t-1} + \ldots + A_N y_{t-N} = B_0 u_t + B_1 u_{t-1} + \ldots + B_N u_{t-N}, \tag{1}$$

where $\{ y_t, 0 \le t \le T \}$ are *m-vector* observations, $\{ u_t \}$ the input process which is assumed to be an uncorrelated sequence of *m-vector* random variables, and A_n and B_n, the m *by* m matrix coefficients of the model.

The model equation can be rewritten as

$$y_t + A_1 y_{t-1} + \ldots + A_N y_{t-N} - B_1 u_{t-1} - \ldots - B_N u_{t-N} = B_0 u_t, \tag{2}$$

and in matrix notation, we have

$$\mathbf{a}_N^T \ \mathbf{y}_t \ - \ \mathbf{b}^*{}_N^T \ \mathbf{u}_t \ = \ B_0 u_t \ , \tag{3}$$

with

$$\begin{aligned} \mathbf{a}_N^T &= [\, I_m, \ A_1, \ A_2, \ \ldots, \ A_N \,], \\ \mathbf{b}^*{}_N^T &= [\, O_m, \ B_1, \ B_2, \ \ldots, \ B_N \,], \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{y}_t^T &= [\, y_t^T, \ \ldots, \ y_{t-n}^T \,], \\ \mathbf{u}_t^T &= [\, u_t^T, \ \ldots, \ u_{t-n}^T \,], \end{aligned} \tag{5}$$

leading to the following augmented equation

$$\begin{bmatrix} \mathbf{a}_N^T & -\mathbf{b}^*{}_N^T \\ O & \delta_0^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_t \\ \mathbf{u}_t \end{bmatrix} = \begin{bmatrix} B_0 u_t \\ u_t \end{bmatrix}, \tag{6}$$

where δ_0 is the first m *by* m block unit vector.

This embedded model can be interpreted as a 2-channel AR model of the joint process $\{ y_t, u_t \}$. Here, the right hand side of the augmented equation is equal to the joint innovations of $\{ y_t, u_t \}$, since

$$\epsilon_t = \begin{bmatrix} \epsilon^y{}_t \\ \\ \epsilon^x{}_t \end{bmatrix} = \begin{bmatrix} y_t - \hat{y}_{t|t-1} \\ \\ u_t - \hat{u}_{t|t-1} \end{bmatrix} = \begin{bmatrix} B_0 u_t \\ \\ u_t \end{bmatrix} . \tag{7}$$

Indeed if we apply a simple interleaving permutation of (1, 3, 5, . . . , $2N+1$, 2, 4, 6, . . . $2N+2$) to the augmented equation, we have the familiar joint 2-channel one-step predictor of the form

$$A_N{}^{\mathsf{T}} z_t = \epsilon_t , \tag{8}$$

where

$$A_N = \left[\begin{bmatrix} I_m & O_m \\ \\ O_m & I_m \end{bmatrix}, \begin{bmatrix} A_1 & -B_1 \\ \\ O_m & O_m \end{bmatrix}, \ldots, \begin{bmatrix} A_N & -B_N \\ \\ O_m & O_m \end{bmatrix} \right] , \tag{9}$$

$$z_t{}^{\mathsf{T}} = [y_t{}^{\mathsf{T}}, u_t{}^{\mathsf{T}}, \ldots . y_{t-N}{}^{\mathsf{T}}, u_{t-N}{}^{\mathsf{T}}] . \tag{10}$$

In the stochastic case, the problem of finding the linear least-squares predictor is then reduced to solving the normal equation of the following form

$$R^{yu}{}_N A_N = [R^{\epsilon}{}_N, 0, 0, \ldots 0]^{\mathsf{T}} , \tag{11}$$

where

$$R^{yu}{}_N = E\, z_t\, z_t{}^{\mathsf{T}} = \begin{bmatrix} R^{yu}{}_0 & R^{yu}{}_1 & . & . & . & R^{yu}{}_N \\ R^{yu}{}_1{}^{\mathsf{T}} & R^{yu}{}_0 & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ R^{yu}{}_N{}^{\mathsf{T}} & . & . & . & . & R^{yu}{}_0 \end{bmatrix} , \tag{12}$$

$$R^{yu}{}_n = E \begin{bmatrix} y_t \\ u_t \end{bmatrix} [y_{t-n}{}^{\mathsf{T}} \; u_{t-n}{}^{\mathsf{T}}] , \quad R^{\epsilon}{}_N = E\, \epsilon_t\, \epsilon_t{}^{\mathsf{T}} . \tag{13}$$

$$R^{yu}{}_0 = \begin{bmatrix} R_y(0) & R_{yu}(0) \\ & \\ R_{yu}(0)^{\mathsf{T}} & I_m \end{bmatrix}, \quad R^{yu}{}_n = \begin{bmatrix} R_y(n) & R_{yu}(n) \\ & \\ O_m & O_m \end{bmatrix}. \tag{14}$$

Since $R^{yu}{}_N$ is a block Toeplitz matrix with blocks of size $2m$ *by* $2m$, the solution to this normal equation can be obtained via the Levinson algorithm [WR]. The parameters estimates which are embedded in the solution are recursively computed in order. (Note the sparseness of R^{yu} and A_N.)

2.5.2 Least-Squares Recursions for ARMA Models

In the deterministic least squares case, the joint forward and backward prediction errors of order n at time T is defined as

$$\epsilon_{n,T} = \begin{bmatrix} \epsilon^y{}_{n,T} \\ \\ \epsilon^u{}_{n,T} \end{bmatrix} = \begin{bmatrix} y_T - \hat{y}_{T|T-1,\dots,T-n} \\ \\ u_T - \hat{u}_{T|T-1,\dots,T-n} \end{bmatrix} = A_{n,T}{}^{\mathsf{T}}\, z[T:T-n]\,, \tag{15}$$

$$r_{n,T} = \begin{bmatrix} r^y{}_{n,T} \\ \\ r^u{}_{n,T} \end{bmatrix} = \begin{bmatrix} y_{T-n} - \hat{y}_{T-n|T-n+1,\dots,T} \\ \\ u_{T-n} - \hat{u}_{T-n|T-n+1,\dots,T} \end{bmatrix} = B_{n,T}{}^{\mathsf{T}}\, z[T:T-n]\ . \tag{16}$$

The covariance matrix for the joint process is thus similar to that given in Section II, and with its shifting properties, one can obtain recursions for the 2-channel ladder form from the recursions of an AR case, i.e.

$$\epsilon_{n+1,T} = \epsilon_{n,T} - D^{\mathsf{T}}{}_{n+1,T}\, R^{-r}{}_{n,T-1}\, r_{n,T-1} \tag{17}$$

$$r_{n+1,T} = r_{n,T-1} - D_{n+1,T}\, R^{-\epsilon}{}_{n+1,T}\, \epsilon_{n,T}\ . \tag{18}$$

In the two-channel case, $D_{n+1,T}$, the partial correlation matrices exhibit addition structures, due to the embedding of the white input process of the ARMA model into an AR model. Since all future inputs are assumed to be uncorrelated with the present and past outputs, half the elements $D_{n+1,T}$ matrix are approximated by zeros. Indeed the partial correlation matrix is given by

$$\Delta_{n+1,T} = \begin{bmatrix} \Delta^{y}_{n+1,T} & O_m \\ \\ \Delta^{u}_{n+1,T} & O_m \end{bmatrix} = \begin{bmatrix} \sum_{t=n+1}^{T} y_{t-n-1}\epsilon^{y}_{n,T}(t)^{\mathsf{T}} & O_m \\ \\ \sum_{t=n+1}^{T} u_{t-n+1}\epsilon^{y}_{n,T}(t)^{\mathsf{T}} & O_m \end{bmatrix} . \qquad (19)$$

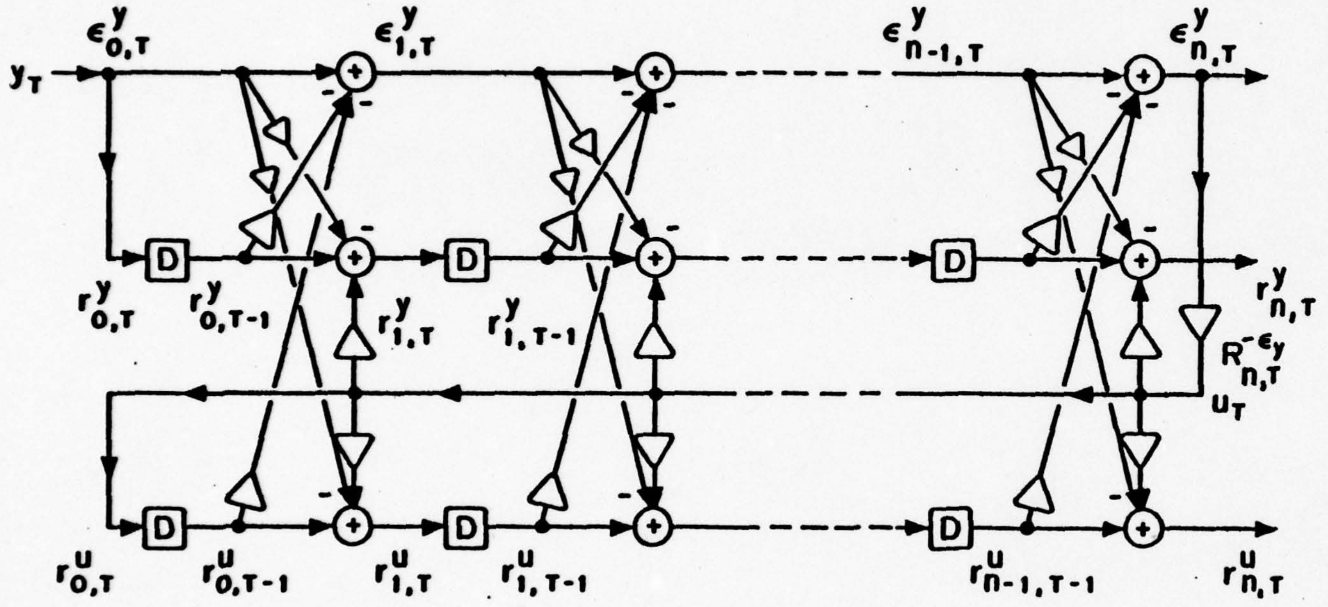
The time-updates for these partial correlations obtained in a similar manner as in Section II.

$$\Delta^{y}_{n+1,T+1} = \Delta^{y}_{n+1,T} + [\ \frac{r^{y}_{n,T}{}^{\mathsf{T}} \cdot \epsilon^{y}_{n,T+1}{}^{\mathsf{T}}}{1-\gamma_{n-1,T}}\ ] \qquad (20)$$

$$\Delta^{u}_{n+1,T+1} = \Delta^{u}_{n+1,T} + [\ \frac{r^{u}_{n,T}{}^{\mathsf{T}} \cdot \epsilon^{y}_{n,T+1}{}^{\mathsf{T}}}{1-\gamma_{n-1,T}}\ ] \qquad (21)$$

In order to obtain ladder recursions for the prediction errors, inversions of the joint $2m$ *by* $2m$ prediction error covariances, $R^{\epsilon}_{n,T}$ and $R^{r}_{n,T}$ are needed. These inversions are simplified by first decomposing them into upper-diagonal-lower form and then taking inverses. Examples of ladder forms obtained this way are presented in [Lee]. Figure 2.5.1 shows an example of one such structure.

Figure 2.5.1. An embedded ARMA Ladder Form.



Notice the feedback structure of the embedded ARMA ladder form. The output of the forward innovations section is fedback as input to the backward residual for $\{ u_t \}$. Assuming that $\{ u_t \}$ has unit variance, we have

$$u_T = R^{-\epsilon y/2}_{N,T} \, \epsilon^{y}_{N,T} \quad . \tag{22}$$

Instead of using one single feedback, as in (22), it is also possible to obtain u_T by distributed feedbacks. Ladder forms of such kinds are still under investigation and results of these studies will be available in later publications.

2.6 Computational Complexity

One of the attractive features of the class of algorithms described in the previous sections is their computational efficiency. The computational requirements of the fast algorithms are in fact proportional to the number of model parameters. This compares favorably with currently used methods as indicated in Table I . There are many ways of measuring computational complexity; the measure adopted here is the number of multiplications per N samples of speech (N may be the number of samples per frame) for a *p–th* order autoregressive model. All index calculations are ignored. For methods that store the full covariance matrix, two-dimensional array indexing has to be performed which may be costly on a machine having only integer arithmetic. Our fast algorithms compute, in addition to the model parameters and gains, numbers that allow us to efficiently compute the log likelihood-ratio used for pitch extraction, i.e. decomposition of the driving process into a jump process and a Gaussian process. Many LPC systems use more computations in the pitch detection section than for the actual modeling, this is not the case for our algorithms. For optimal vector quantization schemes the number of operations is also much larger than the number required for the model parameter computations.

Table I Comparisons of Operation Counts for Fast Algorithms

Estimated number of operations per frame of size N and model order p, multiplications denoted by *, divisions by /, square-roots were ignored. The number of additions is equal to the number of multiplications.

	Pre&Post-Wind.*	Pre-Windowing	Non-Windowing	MEM-type*
AR	$(Np+p^2)*$ + p/	N(4p* + 4/)	N(6p* + 6/)	Np(2* + 2/)
AR Ladder	$(Np+p^2)*$ + p/	Np(6* + 4/)	Np(11* + 10/)	Np(5* + 2/)
ARMA Ladder	$(Np+3p^2)*$ + p/	3Np(6* + 4/)	3Np(11* + 10/)	not available
References	[Lev], [Mo]	[MLNV]	[MVL]	[Burg]

Actual program complexity depends heavily on the computer language used. For example most of our algorithms require programs of a half a page to a page of clearly structured code of the ALGOL type language MAINSAIL. (A FORTRAN program would take about 5 times more!)

* The Levinson and the MEM - Maximum Entropy Methods are not recursive in time, i.e. they would have to be recomputed for every change in frame size or place.

2.7 Implementation Issues

Generally speaking, the implementation of the exact recursive algorithms is fairly straight-forward, if at least some care is taken that is advisable in implementing any algorithm on a computer. Since we implicitly solve linear (normal) equations, the usual precautions for solving linear equations apply also to our case, such as conditioning (ratio of largest to smallest eigenvalue), consistency, etc. I.e. if fewer data samples than parameters are available, the set of equations is over-determined and the covariance matrix or prediction errors are singular, a situation that can arise during startup of a (recursive) algorithm. Some of these properties can be observed in the spectrum of the (speech) signal used. For example, a sum of a finite number of sine-waves without noise leads to a covariance matrix of finite rank (twice this number), i.e. if the number of parameters is more than twice the number of frequencies, the covariance matrix is singular. Numerically it is very possible that the high order prediction errors can become negative and consequently the prediction filter leads to unstable models. It is on the other hand possible for recursive algorithms to have a "self-healing property", that is they can converge back to a stable solution. For a time-varying model this is permissible since the output power can still be bounded; however if the model parameters are sampled, for instance at the end of a transmission frame, the sample could lie in an unstable region that would lead to unstable reconstructions.

Another area of concern during the implementation process is the implicit (statistical) assumptions one makes by choosing particular initial states (say zero) in recursive algorithms. These choices correspond to particular assumptions about the a priori distributions of parameters, which in turn might not be consistent with the actual information available. For instance assuming zeroes for data before the start of the first data sample is equivalent to assuming the data is a white noise (the predictor is initialized as a feed-through); however we are working here with speech, i.e. we have at least an idea what an average spectrum looks like (say from 300 to 3kHz). In order to achieve top performance, many of these issues have to be considered, i.e. if the performance is judged unacceptable it is most of the time due to some underlying assumptions that are inconsistent.

To demonstrate this we discuss two practical problems which appeared during the implementation of these algorithms. The first one is concerned with the accurate tracking of model parameters when strong transients, i.e. nongaussian driving process (e.g. pitch pulses or plosives), are present in the initial data samples. The second problem deals with the capability of fast resets of the recursive algorthms when the underlying model parameters suddenly decrease to zero, i.e. accurate tracking of zero-valued parameters (e.g. no speech).

2.7.1 Non-Gaussian Driving Process

Non-Gaussian signals can have two effects: either they can help to track parameters extremely fast, i.e. within a few samples, or they can "kick the parameter estimates way off", which leads to very slow convergence to the true value of the parameter. Our algorithms produce a likelihood type quantity that can detect the presence of nongaussian components with extreme sensitivity. The second condition can be remedied or prevented as follows. A non-zero (small) initial prior convariance estimate of the input process should be used and a linearly time varying weighting factor (*Tau*) up to the maximum window length (constant *Tau*) is applied to the errors in the least-squares criterion. In the program this just implies that the variable *Tau* is initialized to some initial weighting (say *Tau*(0) = 10*order) and linearly increased proportional to the running time *st* up to a fixed window time constant (say *Taumax*) and then held constant (i.e. *Tau* = (*st* – 10*order*) *min* (*Taumax*)). This is consistent with the knowledge that *speech signals* are expected and not just nongaussian inputs or even out-lyers such as switching noises. The problem really is that the condition number (extreme eigenvalue ratio) of the covariance matrix can get very large if the first sample is unexpectedly large. For example, if the a priori estimate of the power is small, say 1/10,000, and the first sample is a modest 10, its square is 100 and the condition number is equal to the ratio $10^2/(1/10,000)$ or one million. For 6 significant figures in integer arithmetic this is already at the end of the range, i.e. the covariance would be numerically singular. Another good numerical alternative is the square-root form of the normal equation or of our ladder-forms. The solution using a nonzero a priori covariance estimate is simpler, but ultimately we would prefer the square-root versions since they can gain a factor of two in effective wordsize.

2.7.2 Tracking Zero-Valued Parameters

Another problem is the tracking of zero-valued parameters. When the underlying parameters suddenly decrease to zero they can reduce the data samples to very small values. conversely, if the signals get very small, the parameter estimates are not much excited. Special cases of this are when the correlation between two signals goes to zero or the model order of a time-varying model decreases. In all these cases the tracking of the zero-valued parameters becomes very slow, e.g. exponentially in the case where an exponential window is used on the error. This is a basic problem in parameter estimation. Briefly, the reason is that the time-update is obtained by minimizing a squared-errors criterion. When inputs are zeros, the time-update for the *pre-windowed* unweighted case just take the previous estimate as the optimal estimate, because the error criterion "doesn't care" what the non-excited parts of the system does! The existence of an exponential weighting function in the error criterion, the estimates only converge to zero at the rate of the associated time constant, which is undesirable if fast tracking of zero-valued parameters is important for reconstruction of speech. The solution may be to make use of the likelihood ratio parameters to detect the condition for reseting the appropriate parameters. This is a fairly new problem in system identification and not much theory is yet available.

3. The Speech Transmission System

The algorithms described in Section 2 are the basic building blocks of a complete speech compression system. The structure of the system which was developed and tested during our investigation is described in Section 3.1. The rest of the section presents in greater detail the various components of this system.

3.1 General System Description

The basic structure of the speech transmission system used to demonstrate the performance of the new speech modeling algorithms is given in figure 3.1. The sampled speech is the input data to one of the ladder-form algorithms described in Section 2. The ladder form is a recursive type algorithm and produces a new set of outputs at each sample time. The model parameters (reflection coefficients) are sampled at each transmission frame time and they constitute the main part of the per frame speech parameter vector. The other components of that vector are pitch information and residual energy (or gain). The pitch information is obtained from one of the coefficients computed by the ladder form algorithm, as will be described in more detail in 3.2. The energy parameter is a measure of the total energy in the speech frame being transmitted and serves to control the synthesis algorithm driving signal.

The speech parameter vector is used to reconstruct or synthesize the speech from which it was generated. First, however, it is necessary to transmit this information from the analyzer to the synthesizer. Furthermore, the objective of our work is to be able to do this transmission at a low data rate. Thus, the parameter vector has to be quantized and encoded in an efficient manner (e.g. with few bits, but minimum distortion). This function is performed by the encoder/decoder which is discussed in Section 3.3. The coder/decoder is capable of reconstructing a good approximation to the speech parameter vector at the speech synthesizer input.

The speech synthesizer uses the model parameters provided by the encoder to set up a speech model which in some sense reflects the shape of the vocal tract. The pitch and enery information from the parameter vector is used to generate a driving process which is then used to drive the speech model. The details of the synthesis algorithm are presented in Section 3.4.

A basic quantity of the speech transmission system design is the frame size. The frame is a segment of the speech which is chosen as the basic unit of the transmission system. During each frame time a single block of quantized speech parameters is transmitted. To see the relationships between the various quantities let

fs = Sampling rate [Hz]

Ns = Number of raw speech samples in a single frame [samples/frame]

B = Transmission rate [bits/second]

Nb = Number of bits per coded speech parameter vector [bits/frame]

In order to get real time transmission we must have $(fs/Ns) \leq (B/Nb)$ or

$$(\text{Block Transmission Time}) \leq (\text{Frame Time}) \tag{1}$$

In words, the transmission of the parameters for a speech frame must be completed in less than the frame time. In most cases a strict equality will be chosen in equation (1), thus

$$Nb\,/\,B = Ns\,/\,fs \quad \text{bps} \tag{2}$$

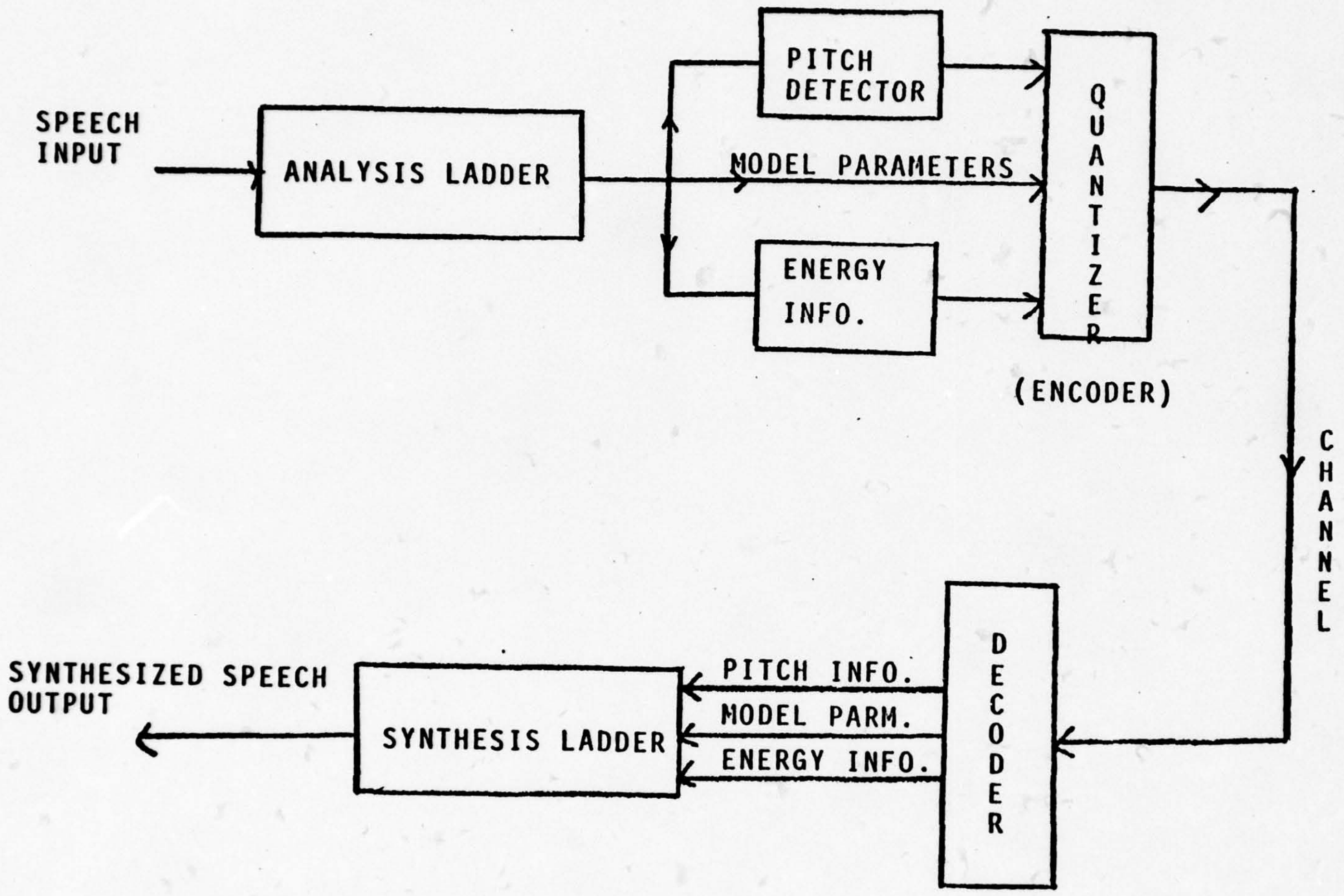
$$\text{i.e.} \qquad B = Nb\,fs\,/\,Ns \tag{3}$$

Some typical values used in our simulations are

$$fs = 8000 \ , \quad Ns = 60 \ , \quad Nb = 18 \ , \tag{4}$$

which means that a transmission rate of B = 2.4 kbps is required and that the frame duration is 7.5 msec, i.e. within a commonly accepted limit on the rate of change of speech parameters.

Fig. 3.1 The speech transmission system



SPEECH TRANSMISSION SYSTEM

3.2 Speech Analysis and Parameter Transmission

The computation of the reflection coefficients can be carried out by any one of the algorithms described earlier. For the actual simulations the pre-windowed ladder form was chosen as being the most cost-effective candidate for this purpose (see Section 5). Outputs of the algorithm feed a transmission module which selects the contents of the per-frame parameter vector.

Because of the recursive nature of our algorithms, they can be run continuously and do not have to be synchronized to the frame period. Once in every frame period the best estimates of the reflection coefficients are chosen for transmission. Similarly pitch and energy information is passed on to the transmission encoder.

The analysis algorithm decomposes the original speech signal into a model, to be represented by the reflection coefficients, and a driving process, to be represented by pitch and energy. Since the analyzer operates "continuously", its outputs track the original speech. This permits us to logically separate the speech modelling from the data transmission. Provided we transmit the model parameters "often enough", we are at liberty to choose the frame size to optimize transmission efficiency. There is a tradeoff to be made. We can choose a short frame and send parameter updates more often, but at the expense of having fewer bits available to do it with. Alternatively, we can choose a long frame, providing many bits to quantize the parameters, but at the expense of sending fewer parameter updates. The tracking property of the analysis algorithm allows this tradeoff to be made independently of the algorithm design.

3.2.1 Model Parameters

At each frame time, we choose a best estimate of the model parameters for the frame. Sampling the model parameters current at the frame boundary is appealing, but we have adopted a slightly modified strategy. Simply stated, at each frame boundary, we transmit the most recent "reasonable" model parameters. Since the algorithm tracks so fast, if a pitch pulse or algorithm reset occurs very near the end of a frame, a pure sampling strategy would select parameters not representative of the entire frame. The transmission module maintains a short history of the model parameters; if a pitch pulse or reset occurs in approximately the last third of the frame, then we transmit parameters current slightly before the disturbance.

This rules were motivated by our finding that in the voiced case the reflection coefficients are piece-wise constant. From a rate distortion point-of-view we would have to test the time-varying model against a set of prototype models (that could be constant within a frame) and send the index of the "nearest" model in a distortion measure.

First we used a uniform quantization scheme to test out our programs and the limits of a simple quantization scheme, acceptable for 9600baud transmission rates. In order to compare our models against other modeling schemes we selected for one set of experiments up to 10 reflection coefficients using the optimal quantizer published in [MG], and bit rates of 3050 and 6100 just for the reflection coefficients were observed. As a next test our own pitch detection algorithm was used to test out a complete transmission system. This is unfortunately a difficult step since a separate evaluation of the modeling part of the system can only be done with a perfect pitch detector which was unavailable to us at the time of evaluation. (This is somewhat similar to trying to test race-tires on a passenger car!)

3.2.2 Pitch Information

Pitch detection is a crucial and difficult part of most speech compression techniques. Many kinds of pitch detection techniques have been used (see e.g. the survey by Rabiner [Rab]).

The ladder-form algorithms developed in this investigation provide a novel integral pitch detection method which seems to be very promising. The speech driving process consists of a gaussian part (unvoiced), and a jump part (voiced). The separation of such a mixed process may be described as a Doob-Meyer decomposition into predictable and Martingale difference components. Such decompositions have recently been receiving increased attention in the non-linear estimation literature but almost no practical results are available.

A parallel algorithm might be designed to obtain the pulse positions from the original time series, but the non-whiteness of the gaussian component will act to cloud the position of the impulses. In our ladder-form algorithm, the time update γ (gamma parameter) is a liklihood ratio directly useful in separating the mixed driving process. The normalized log-likelihood function for a process parametrized by its innovations representation using ladderforms can be obtained relatively easily using the formulas appearing in the ladder form equations given in previous section. Given a (zero mean) scalar gaussian process with covariance matrix R, the determinant $|R| = R^{\epsilon}_{0} \cdot \ldots \cdot R^{\epsilon}_{p}$, where the $\{ R^{\epsilon}_{i} \}$ are the prediction error covariances related to the reflection coefficients K_i in the stationary case via $R^{\epsilon}_{i+1} = R^{\epsilon}_{i} (1 - K_i^2)$. Now the log-likelihood function ll can be obtained using (29') of Section 2 and the standard formula

$$ll = ln|R| + \|y\|^2_{R^{-1}} = \sum_{i=0}^{p} ln\, R^{\epsilon}_{i} + \gamma_p \quad . \qquad (5)$$

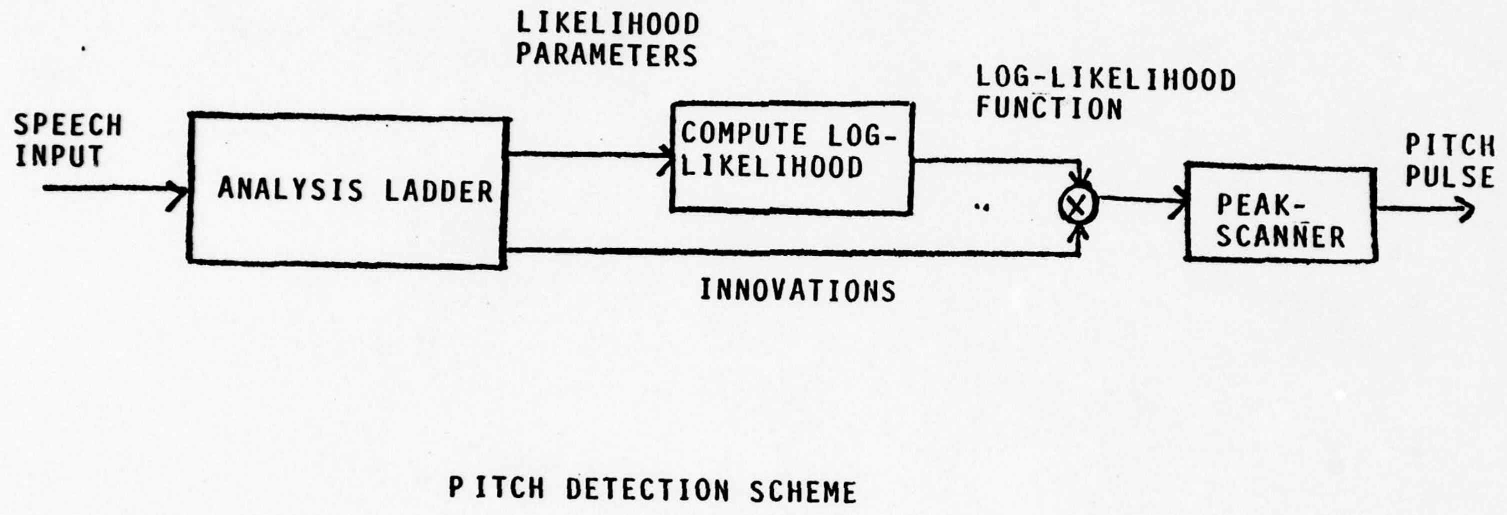
We applied several versions of this formula to the pitch detection problem. The increase in ll per sample, i.e. the time differenced form $\delta\, ll$ is a very sensitive measure of the "unexpectedness" of a time sample, i.e. a measure of deviation from non-Gaussianness. We have Used a simple local maximum algorithm either on $\delta\, \gamma_{pt}$ or $\delta\, ll$ combined with an exponential threshold detector to locate the position of driving impulses. As the figures in Section 5 show, the result is quite remarkable. The large jumps in ll or γ itself can clearly be related to the start of a new word or even phoneme. Eventhough this research was not directed towards pitch detection, we discovered that a high quality pitch detection scheme was required in order to test our modeling methods. The likelihood approach is extremely well matched to our modeling approach, so some effort was directed towards the development of this method. However we consider this as only a first attempt in the direction of finding methods to decompose (discrete) mixed processes. Such results would hopefully avoid the use of any more or less ad-hoc schemes containing internal "tweeking" factors of heuristic nature, a characteristic of most present day pitch detectors.

The end result of all this is that the speech analysis algorithm includes an integral pitch detector which provides a pitch-pulse-present signal at every sample time. In our transmission module, we have adopted the strategy of transmitting the time indices of these pulses. In a loose sense, we are using a run-length coding scheme to transmit the sparse one-bit-per-sample sequence which represents the pulse locations. An important result of this technique is that there is no need to make the usual voiced/unvoiced decisions.

3.2.3 Energy

The ladder-form residuals, or innovations, contain the energy of the driving process. Our transmission module computes the frame average driving source power by summing the energy of the innovations of the original signal over each frame.

Figure 3.2 Illustration of Log-Likelihood Computations for Pitch Detection



3.3 Quantization Issues

For comparison reasons we used the uniform single symbol quantization scheme given in [MG]. At 80 samples per frame and 10th order models the 61bits per frame lead to a transmission rate of 6100bits per second for the K's only. The speech degradation using this quantization was not perceptible in our experiment. A few trials were made with rate-distortion quantizers at 8 bits per frame for the K's, i.e. about a reduction of a factor of 8 in transmission rate for the K's alone. The prototype models needed in the rate-distortion have to be trained to the particular method and a collection of representative speakers. Nevertheless, in one of our experiments we encoded our model parameters obtained without pre-emphasis with respect to a set of prototypes trained on a different speaker set using a standard LPC system with pre-emphasis, see [MG]. Using also a subgrade pitch detector, the result was surprisingly good considering the fact that only 8 bits (i.e. 256 prototypes) were used. We do not consider this a conclusive experiment, but rather a strong indication that these methods are very promising and have the potential to achieve the desired low transmission rates. Pitch information and energy parametrization can be handled similarly, if our modeling techniques can be extended to produce somewhat more accurate pitch period estimates than this first attempt. The raw pitch period estimates obtained now would have to be smoothed over several pitch periods and rate-distortion encoded. Energy or gain can be handled relatively simply via rate-distortion encoding, i.e. similarly to the K's.

3.4 Speech Synthesis

The speech synthesis uses the frame synchronous parameter vector to set the model parameters of a ladder form filter and to set up the appropriate driving process. The model creation is straightforward. There is no need to modify the reflection coefficients to create the appropriate inverse filter as a ladder form may be inverted by reversing its signal flow graph and running it backwards.

The synthesis module creates a driving process based on the per-frame pulse position and energy information in the parameter vector. If no pulses are present in a frame, the source energy is put into a gaussian noise component - i.e. an unvoiced decision is made. If pulses are found the source energy is distributed equally among them. Ultimately the driving process could be a mixture of a gaussian and jump component, each with its own gain (and even its own linear filter.)

Many future practical problems remain, such as inter- and intra-frame smoothing and dividing up the energy between gaussian and jump components. Again, ideally a time-varying rate distortion encoder could take care of this.

4. Software Tools

The similarity between the various speech processing algorithms suggested that they be written as simple modules which "plug" into a framework that supplies I/O facilities, graphics, and debugging tools. The resulting algorithm module expresses the essence of the algorithm, and removes the extraneous code required to interface the algorithm to some particular machine environment. We have also attempted to write the framework as well as the modules in a machine-independent fashion. This necessitates a machine-independent language, and we have chosen MAINSAIL [Wil], a dialect of Algol patterned after SAIL [Reis].

4.1 A Machine-Independent Language

Unlike Fortran, which is not actually portable, MAINSAIL (MS for short) was expressly designed to allow program portability to any reasonable machine with a word length of 16 bits or larger. The compiler itself is written in MS, and currently generates code for PDP-10s and PDP-11s. Support for Data General machines, IBM 370s, Interdatas, and other machines is planned. We expect that the module sources will be very portable as a result. The frame and existing modules are written in MS, but are now compiled with SAIL because the MS compiler is not well supported at SUAI yet. The PDP-11 UNIX version of MS is expected soon. A library of SAIL macros and procedures is currently used to simulate MS syntax and its runtime environment. Preliminary experience indicates that about 5 minutes of editing is required to convert one of these modules to the real MS syntax. Programs which use more extensive syntax require more effort.

The MS language supports integers and long integers with guaranteed minimum sizes of 16 and 32 bits, respectively. The long integer construct is essential here for sample numbers, etc. MS supports reals and long reals with minimum sizes of 6 and 11 decimal digits, respectively, with base 10 exponent ranges of plus or minus 38. There are garbage-collected strings *ala* SAIL and PL/1, and garbage-collected records and pointer variables. An important point is that I/O is carefully defined so that both text and data files can be randomly accessed in a way which is relatively independent of the machine and operating system. The data files themselves are not directly portable, but programs which read and write data files are functionally equivalent on different machines.

4.2 Frame and Module Structure

Each algorithm module is separately compiled and dynamically linked with the framework at run-time by the MS run-time system. This costs very little, since each MS module is position-independent, and accesses data and procedure fields in other modules via a pointer/field addressing scheme. It also means that very large programs can be supported in a small memory space by breaking them into modules which are demand swapped into memory by the run-time system. This requires that the machine architecture support position-independent code so that modules may be loaded anywhere without relocation. MS code modules are garbage-collected just like records when space is needed.

Each algorithm module consists of 5 procedures and some internal data fields. The procedures are named defineExp, initialize, analyze, quantize, and synthesize. The frame calls these when appropriate; its main processing loop has the following skeleton structure:

```
initialize;                         * set up initial state
for block ← 0 upto Nblocks-1 do
  begin
      read input data block;        * input speech signal
      analyze;                      * linear prediction
      quantize;                     * parameter transmission
      synthesize;                   * reconstruct speech
      write output data block;      * output speech & diagnostics
  end;
```

A block would be just a single sample for an on-line method. This structure represents a sub-experiment; it is enclosed in a loop which allows the initialize procedure to change parameters which must remain fixed while reading the input speech file, such as block size. The same speech file is re-read for each sub-experiment, and the output speech file is a concatenation of the results of each sub-experiment. The defineExp procedure is called at the very beginning of the total experiment to set initial parameter values.

The SU-AI SAIL system does not support the MS module concept, so a similar structure was worked out that is compatible with SAIL and upgradable to MAINSAIL. Essentially each module is a separate program, e.g. *analyze*, *quantize*, and *synthesize*. The communication between the "modules" is via data files. A number of other programs have been written to manipulate these data files, including an interactive graphing program. The internal structure of each program is still clean, however, since the I/O required is simple (it emulates the MS I/O system). Also, the modules are broken into several procedures for clarity, and structured coding practices were used to maximize the readability.

The flow of control in each module can be better ascertained from the MAINSAIL (MS) source than from a flowchart-type description, which does not convey the hierarchical structure. The flowchart is an attempt to document the flow of control in a program which has poor structure. With languages like Fortran, where undisciplined use of GOTOs and three-way IF statements obscure the control-flow, the flowchart may provide some information. But with block-structured languages like MS and SAIL, the control flow is most naturally described using the language features provided for structured control, namely WHILE, FOR, CASE, and IF-THEN-ELSE. In addition, the variable names have more meaning because both upper and lower case are allowed, and may be any length. Flowcharts were not made for the programs included in the appendix because such a flowchart is more difficult to understand than the actual sources.

5. Simulation Results

In this section we present results of the simulations carried out to test and verify our algorithms.

5.1 Constant parameter AR models

We first present simulation results of the analysis of data generated by time-invariant models using the *pre-windowed* ladder form and the *Levinson algorithm*. Both algorithms were applied to autoregressive (AR) systems of various orders. Reflection coefficients were computed on-line in the pre-windowed ladder form. In the Levinson algorithm, reflection coefficients were computed with data blocks of increasing blocksize, a excellent method for testing block methods for consistency.

Data generated from two AR models, 4*th* and 8*th* order, were tested. Zero-mean unit variance white noise input was used. The parameters for the 8*th* order model were taken from [MG,p.128].

The model parameters (A coefficients) and their corresponding reflection coefficients for the two models are shown in *Table 4.1* and output data are shown in *Figures 4.1 a, b* .

5.1.1 Simulation Results

A large number of simulations were performed on the most promising algorithms and a few characteristic examples are presented here. For each algorithm, 1000 samples of data were analyzed and the estimates of the reflection coefficients were plotted as a function of number of samples. All figures have time or number of samples as abscissa, where "1" corresponds to 1000 samples. The ordinate is the scale for the values of the various estimates of the reflection coefficients, using an automatic scaling of maximal and minimal value corresponding to the top and bottom of the graph. A horizontal line indicates the true value of the estimated reflection coefficients; hence the asymptotic behavior or convergence of the estimates as a function of time can easily be inspected.

In order to eliminate the irregularities in the estimates caused by block boundaries, some windowing was applied to the data within each block.

A *trapezoidal* window of the following form

$$\begin{aligned} w(n) &= n / 10 , & &\text{for } 1 \le n \le 9 \\ &= (T - n + 1) / 10, & &\text{for } (T - 9) \le n \le T \\ &= 1 , & &\text{otherwise} \end{aligned}$$

and a *Hamming* window of the following form

$$w(n) = 0.54 - 0.46 \cos (2 \pi \frac{n-1}{T-1})$$

were used.

Estimated Parameters of 4th Order Model

In order to illustrate and compare the convergence properties of each algorithm, the estimated reflection coefficients from each algorithm were collected and displayed at each prediction order. The results are shown in *Figures 4.2 - 4.5.* In each figure, the displays are arranged in the following order

(a) pre-windowed ladder form $K^{\epsilon}{}_{p,T}$,

(b) pre-windowed ladder form $K^{r}{}_{p,T}$,

(c) Levinson $K_{p,T}$ without windowing,

(d) Levinson $K_{p,T}$ with *trapezoidal* window, and

(e) Levinson $K_{p,T}$ with *Hamming* window.

The algorithms were also extended to estimate beyond *4th* order upto *8th* order, and the results are shown in *Figures 4.6 - 4.9.*

Estimated Parameters of 8th Order Model

The results for the *8th* order model were arranged in the same fashion as for the *4th* order model, and they are shown in *Figures 4.10 - 4.17*

5.1.2 Discussion of Results

A comparison of the various simulations shows the following characteristics.

1. Comparing the estimates of the reflection coefficients from the pre-windowed ladder form and those from the Levinson algorithm, it can be readily seen that that the latter estimates have a higher variance, i.e. they are very "noisy". Closer inspection reveals that the pre-windowed estimates match one of the "envelopes" of the noisy estimates computed via the Levinson recursions; furthermore it is the envelope closer to the true estimates. Hence the Levinson estimates appear to be uniformly worse than the pre-windowed estimates in terms of mean bias.

2. The convergence of both pre-windowed and Levinson recursions seem to depend somewhat on the index of the reflection coefficients. The pre-windowed estimates converge in general to within 10% on less than 250 samples, (i.e. a typical blocksize for speech analysis). However the Levinson recursions produce estimates that are so "noisy" that only non-linear schemes could improve them because of the asymmetry of the irregularities. (The pictures suggest the idea of "weeds" growing away from the best estimates represented by the envelope closer to the true parameter value).

3. In order to reduce the high variance of the estimates obtained from the Levinson recursions, windowing has traditionally been used. Our simulations using a Hamming window show unfortunately that windowing can quite drastically alter the dynamics of the estimates. The initial transients are completely different even with a trapezoidal window which affects only peripheral points ($n = 10$). Even more drastic differences are produced by the use of the Hamming window. It appears that the transients are now a complicated convolution of the original response and the window function. Smoothing is achieved for some parameters but at the cost of new transients lasting over 500 samples, or more than twice the commonly used blocksize. This would lead to biases in the estimates of the reflection coefficients and would ultimately result in "rough" speech.

4. In light of these results, several alternatives will be explored. The pre-windowed method can be easily modified to use an exponential weighting or

scaling so that the most recent data contributes most to the estimates. The convergence variance and bias of the estimates can be studied via the forward and backward prediction errors. The Levinson recursion without windowing can serve as a standard to compare other forms like pre-windowed, "covariance" and Burg type estimates. The somewhat *ad hoc* windowing commonly used can lead to drastic changes in the transient behavior of these algorithms, and should therefore be used with great caution! Methods which don't require windowing, such as the ladder - forms, avoid this problem altogether.

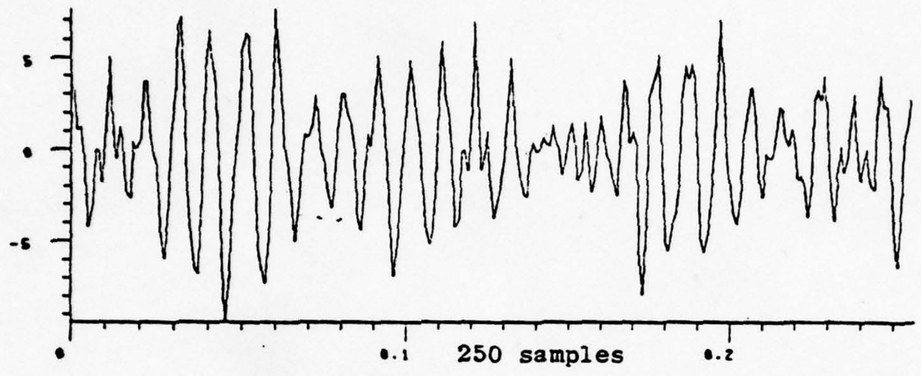
4th order AR process

i	$A[i]$	$K[i]$
0	1.000000	.0000000
1	-1.313600	-.7424092
2	1.440100	.8082622
3	-1.091900	.01756615
4	.8352700	.8352700

8th order AR process

i	$A[i]$	$K[i]$
0	1.000000	.0000000
1	-2.346440	-.9421574
2	1.656970	.9238739
3	-.005990000	-.5619754
4	.3230500	-.09454902
5	-1.482130	.2021682
6	1.154630	.5359436
7	-.1896600	-.3292221
8	-.05899000	-.05899000

Table 4.1 Model parameter and reflection coefficients of AR models.



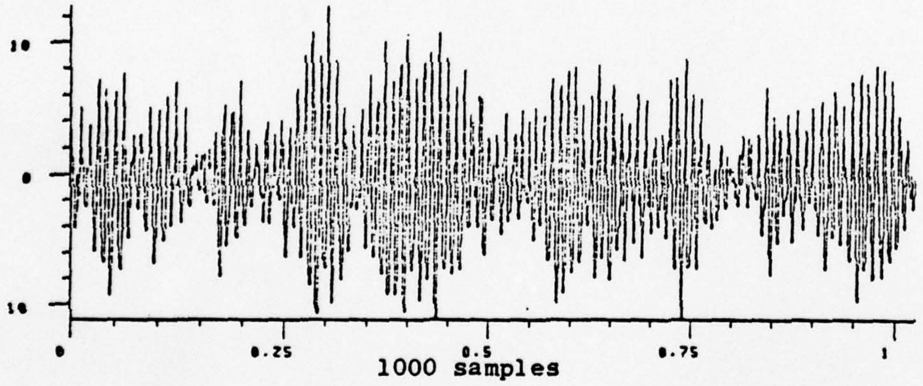
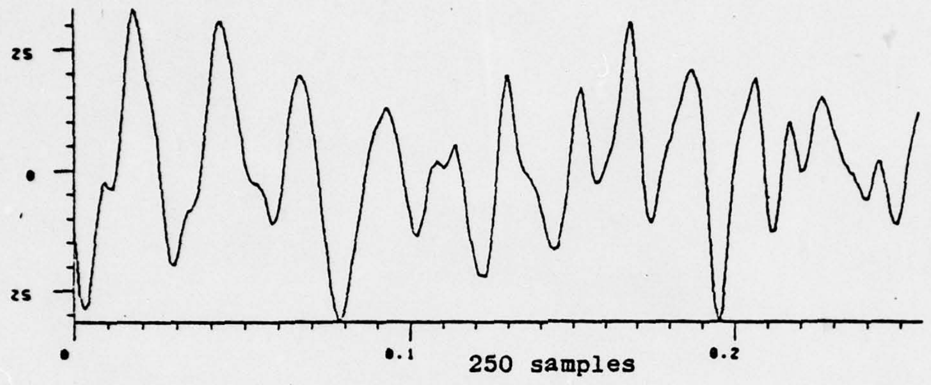


Figure 4.1(a) Data generated from 4th order AR model.



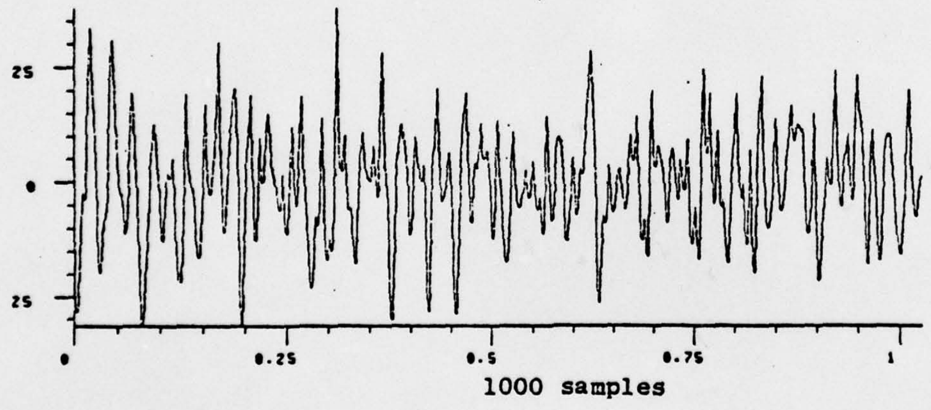
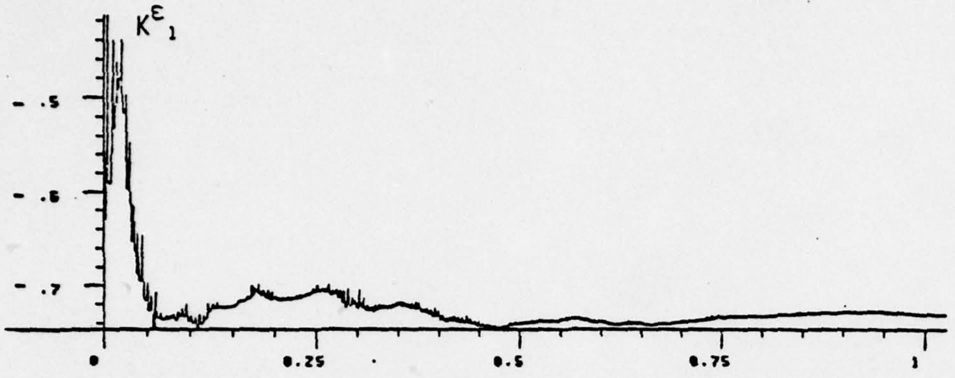
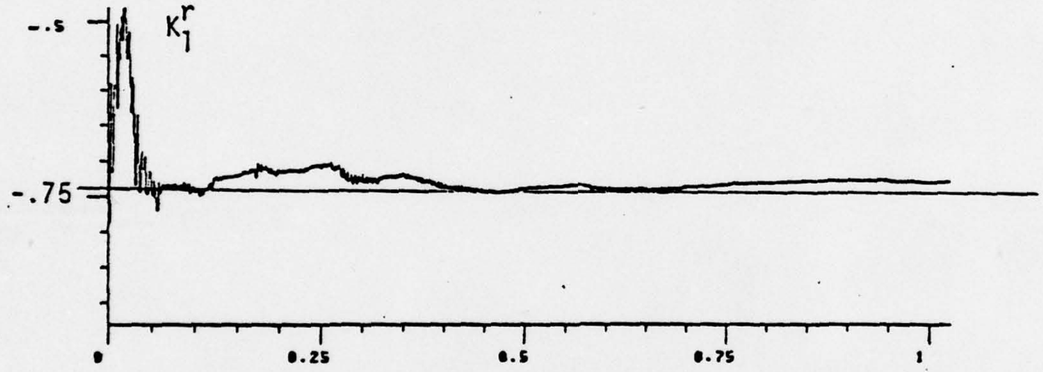


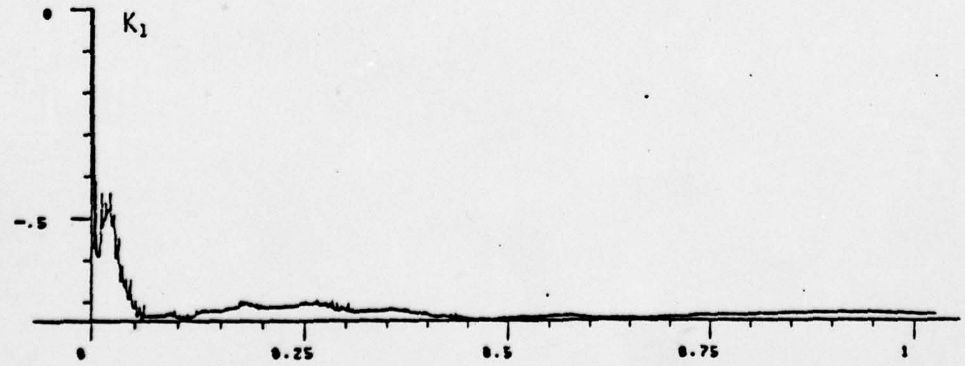
Figure 4.2(b) Data generated from 8th order AR model.



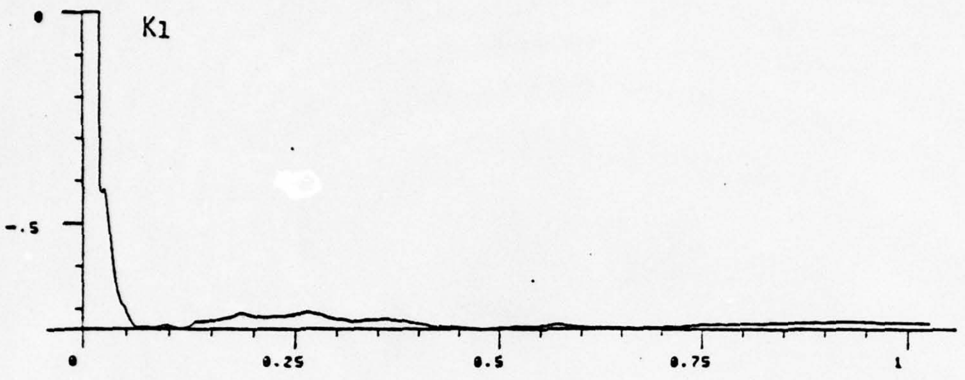
(a) Forward Reflection Coefficient , Pre-windowed Ladder Form



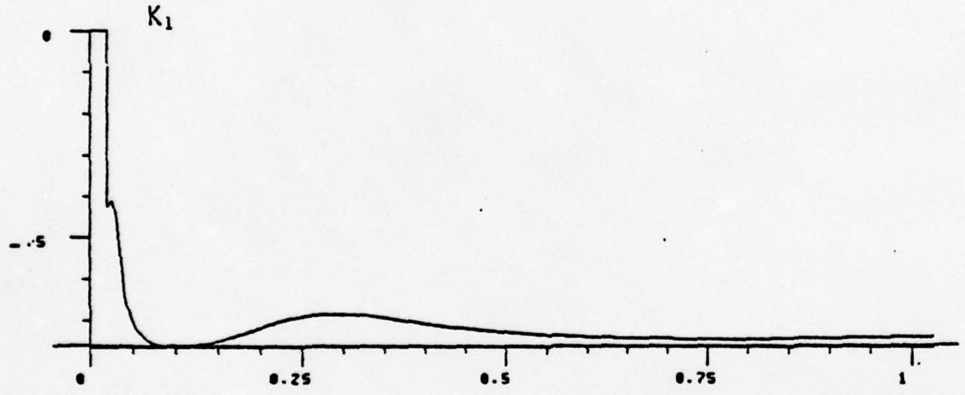
(b) Backward Reflection Coeffieient,Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

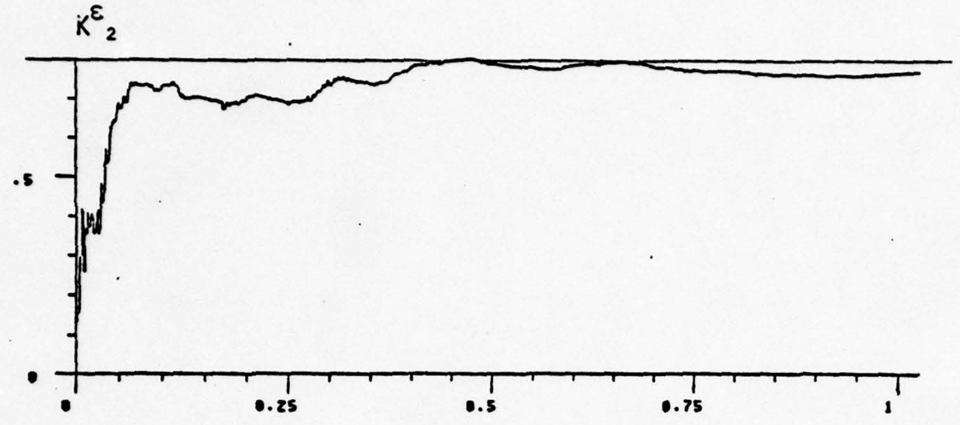


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

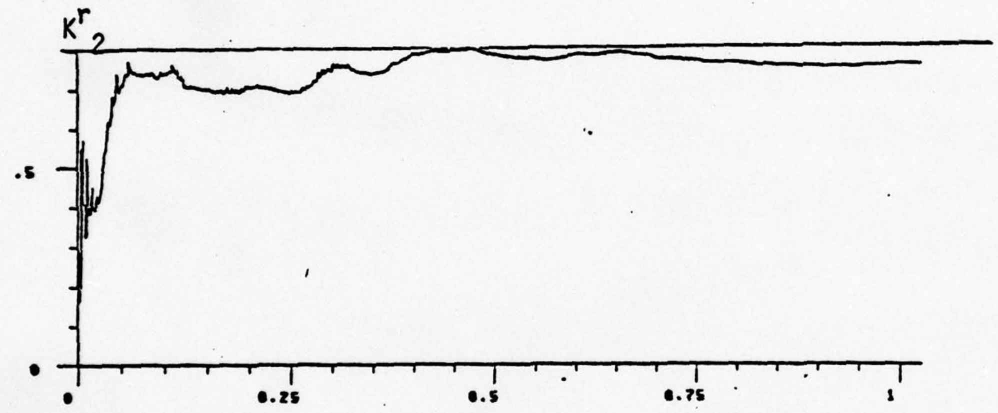


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

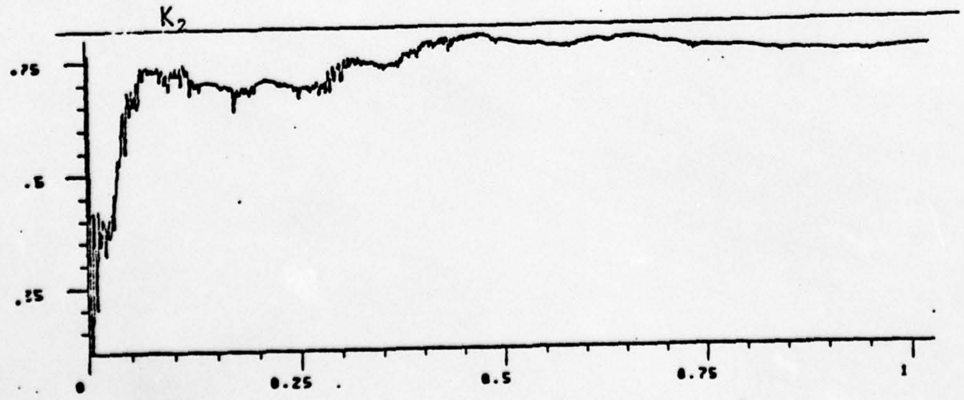
Figure 4.2 First order reflection coefficient of 4th order AR process. All algorithms give reflection coefficients that converge within 100 samples. Notice the smoothing effect of the two windowed estimates.



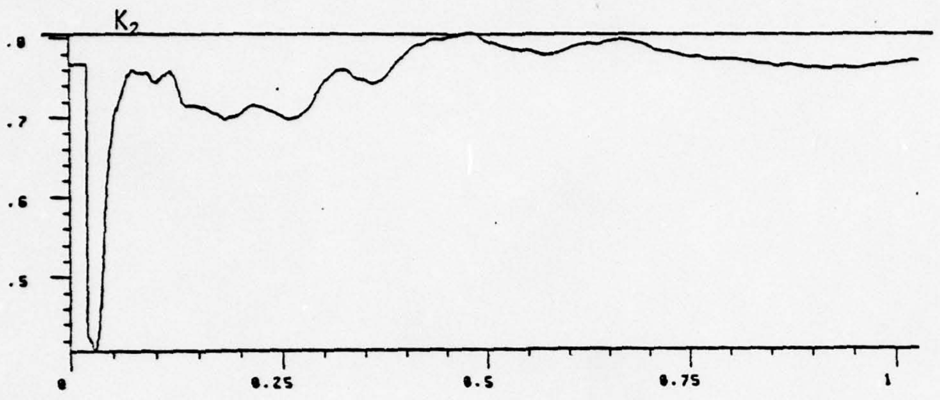
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



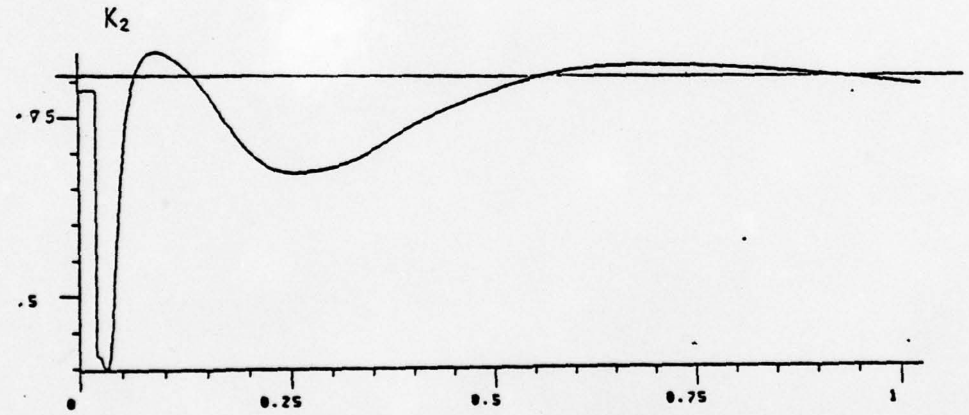
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing



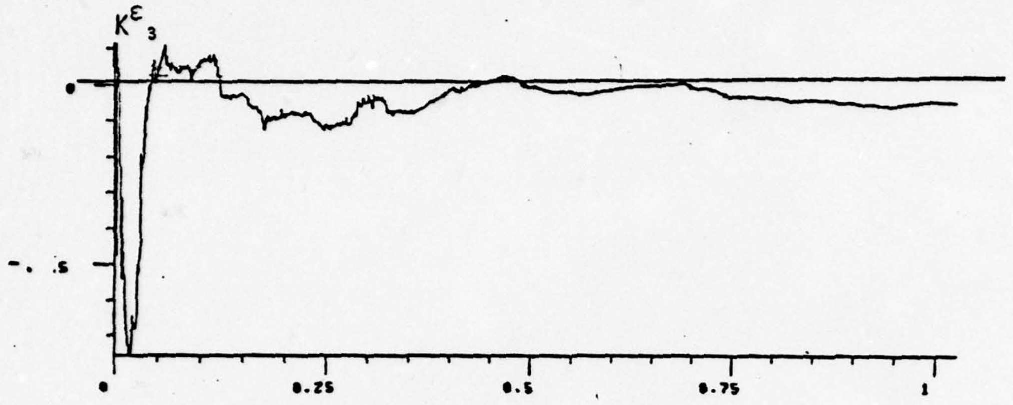
(d) Reflection Coefficient,Levinson Algorithm with Trapezoidal Window



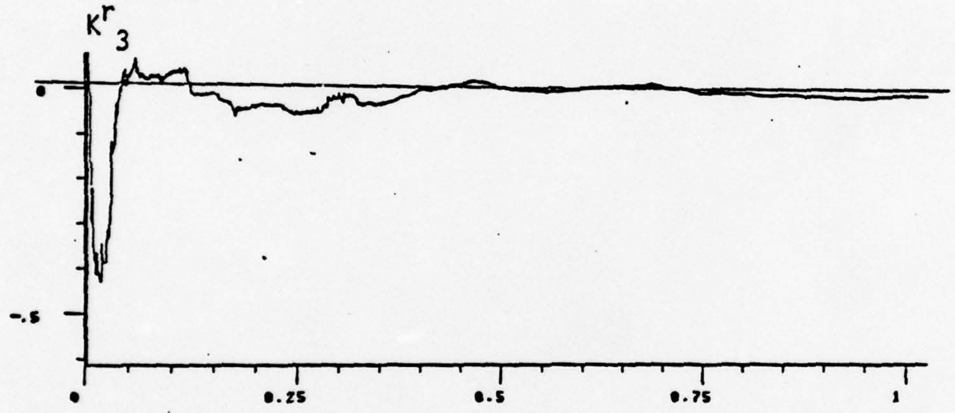
(e) Reflection Coefficient, Levinson Algorithm with Hamming window

Figure 4.3 Second order reflection coefficient of 4th order process.

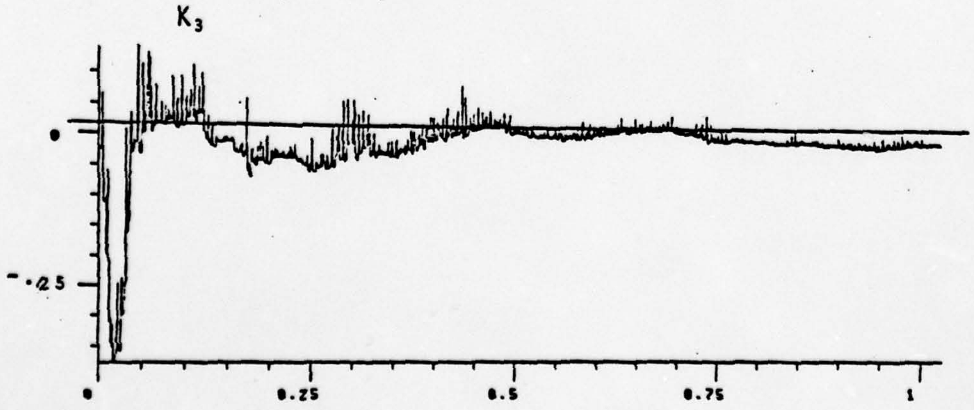
Both reflection coefficients from Pre-windowed Ladder Form converge to the true value in 250 samples. Also, estimate from Levinson Algorithm without windowing converges in the same fashion as the Pre-windowed Ladder Form . Both of the windowed estimates exhibit smoothing effects and in particular, the Hamming window introduces biased behaviour at around 250 samples region.



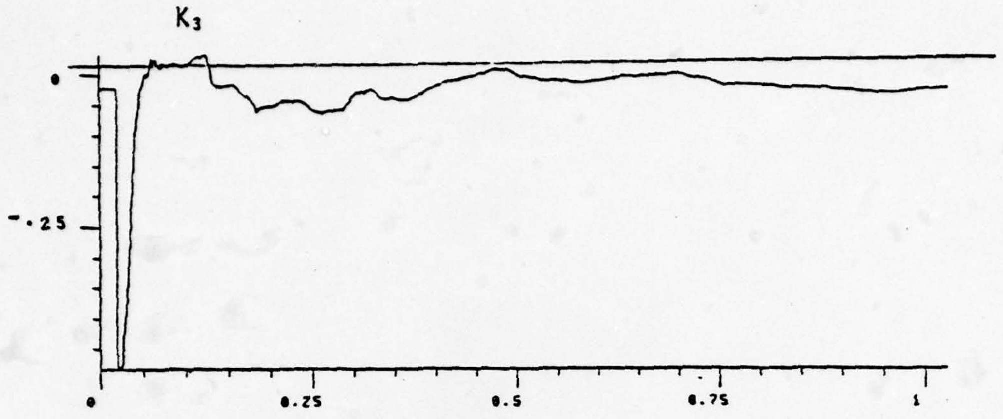
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



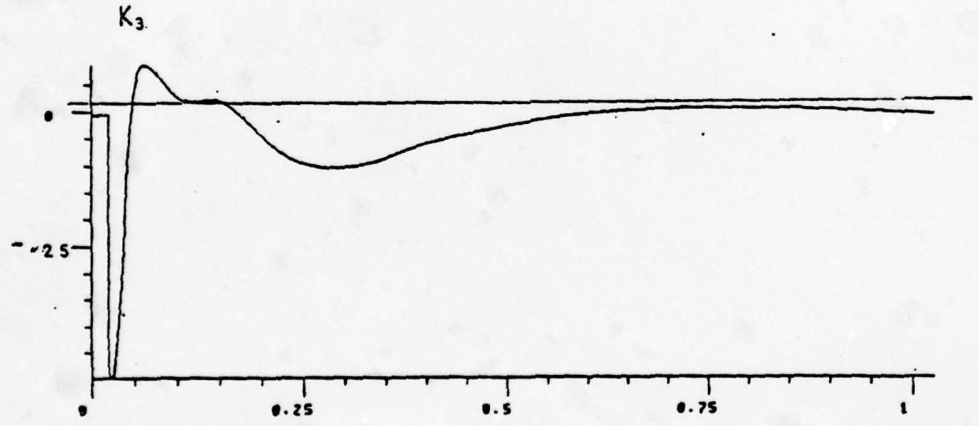
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

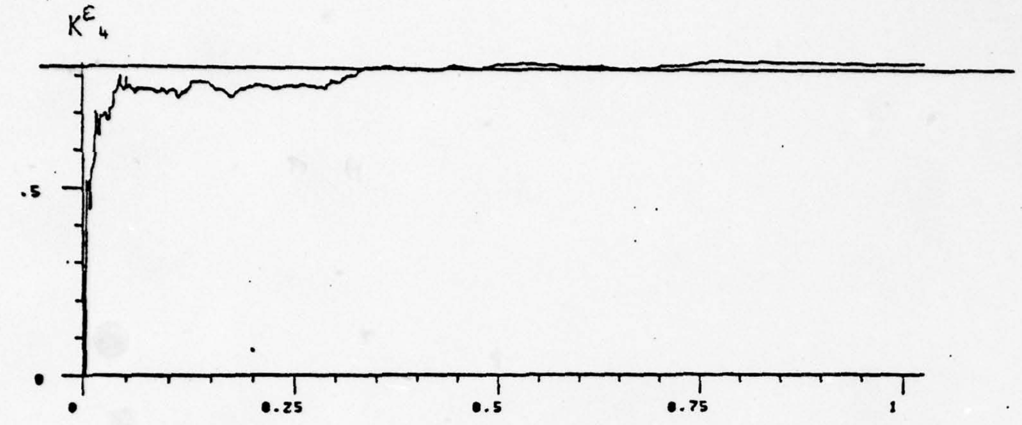


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

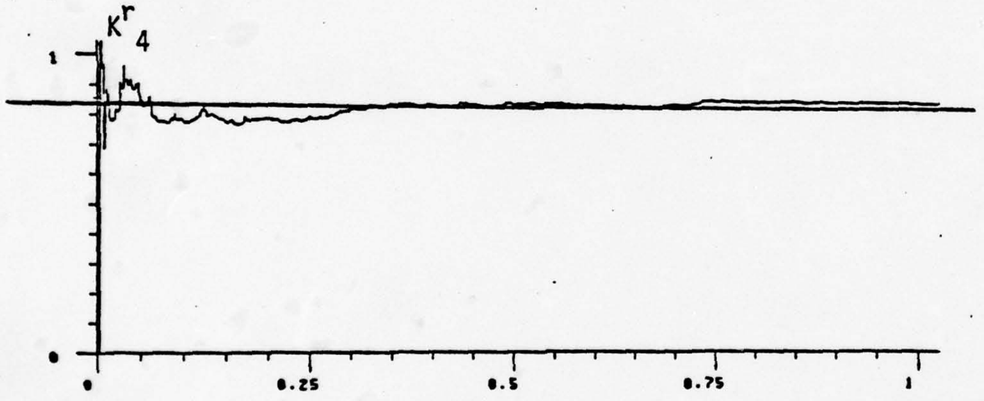


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

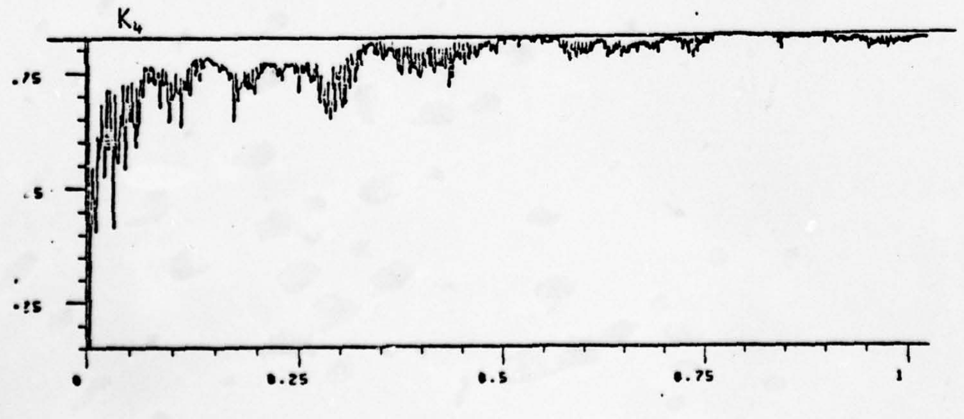
Figure 4.4 Third order reflection coefficient of 4th order AR process. Notice the noise-like spikes appearing in the estimates obtained from Levinson Algorithm without windowing. The picture suggests the idea of "weeds" growing away from the best estimates represented by the envelope closer to the true parameter value.



(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



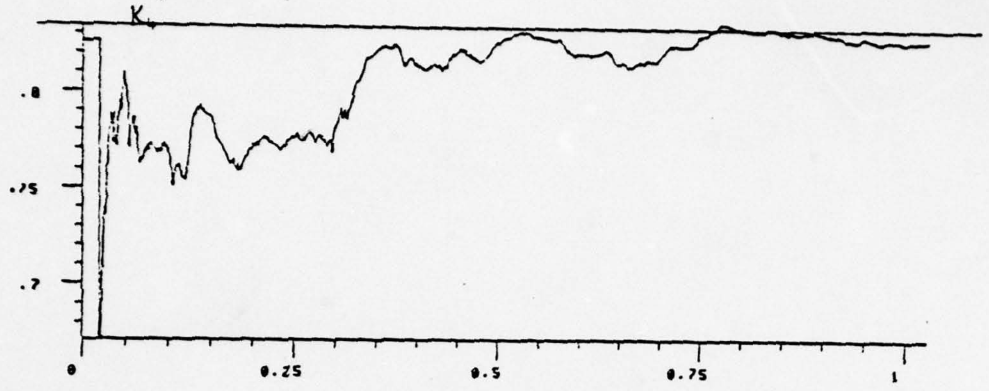
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



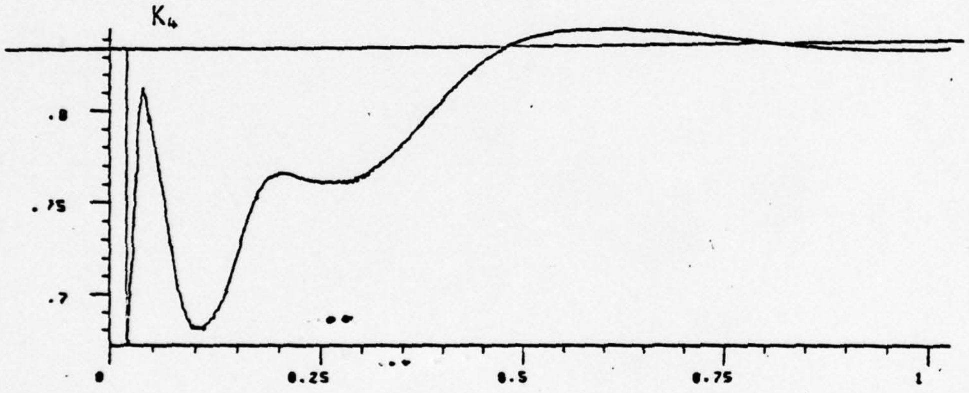
(c) Reflection Coefficient, Levinson Algorithm without windowing

AD-A070 050 STANFORD UNIV CALIF INFORMATION SYSTEMS LAB F/G 17/2
FAST ALGORITHMS FOR SPEECH MODELING.(U)
DEC 78 M MORF, D T LEE DCA100-77-C-0005
UNCLASSIFIED NL

2 OF 3
AD
A070050

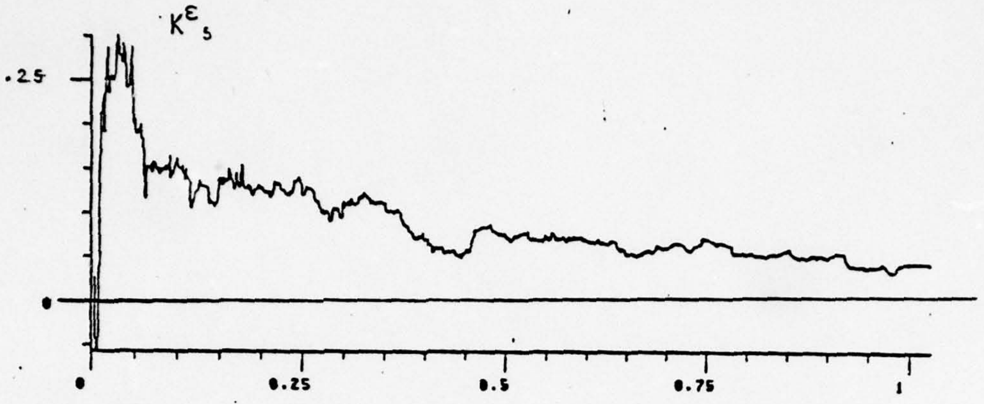


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

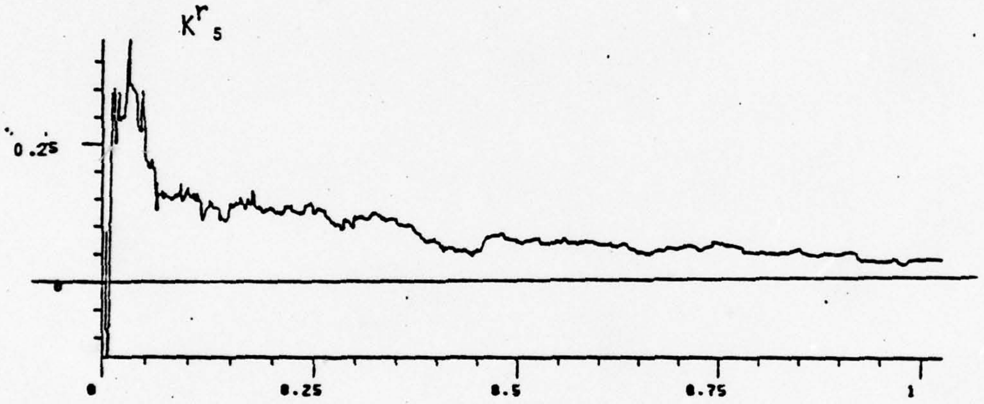


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

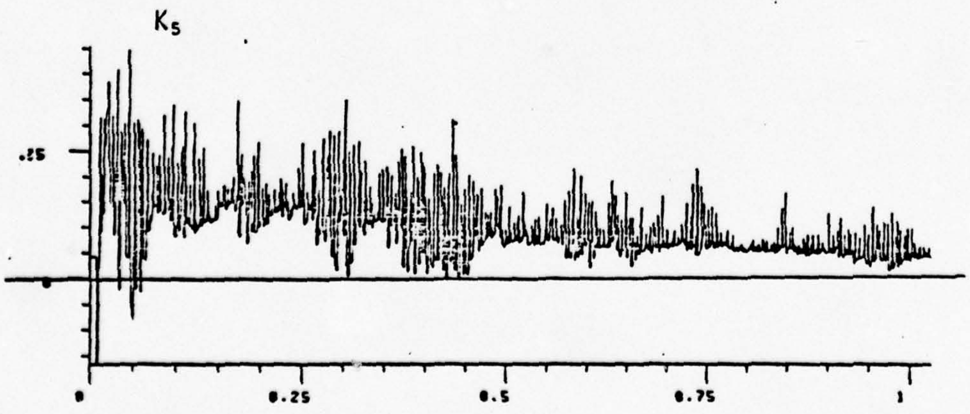
Figure 4.5 Fourth order reflection coefficient of 4th order AR process. Notice how the Hamming window alters the transient behaviour during the first 250 samples.



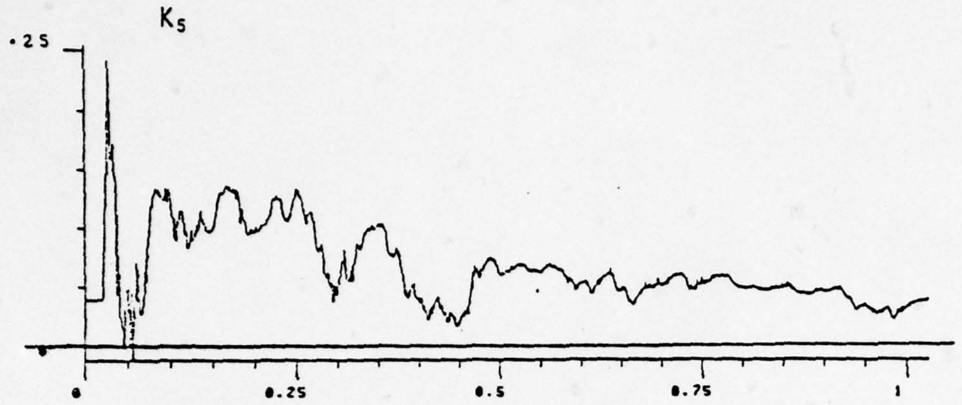
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



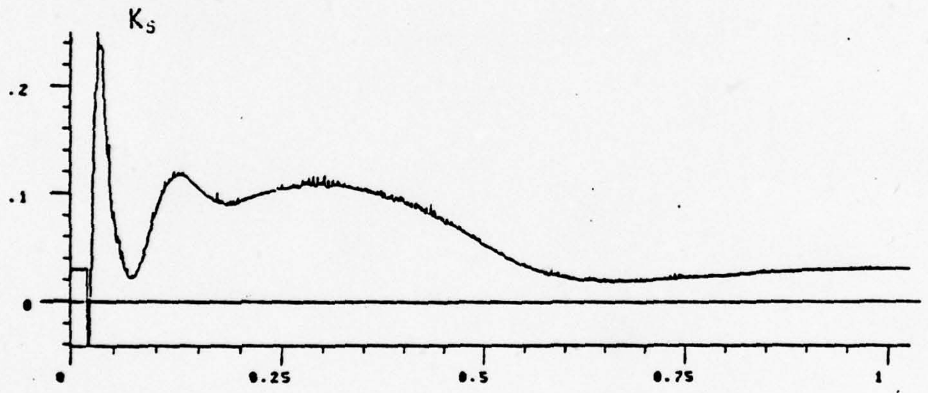
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Coefficient without windowing

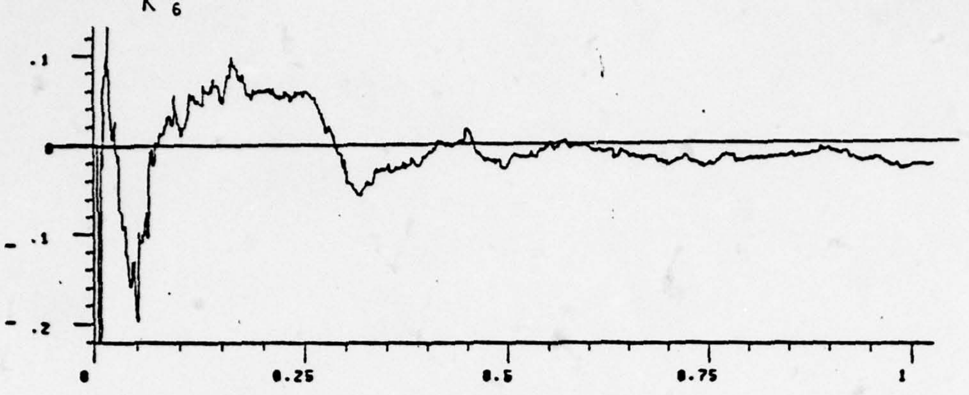


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

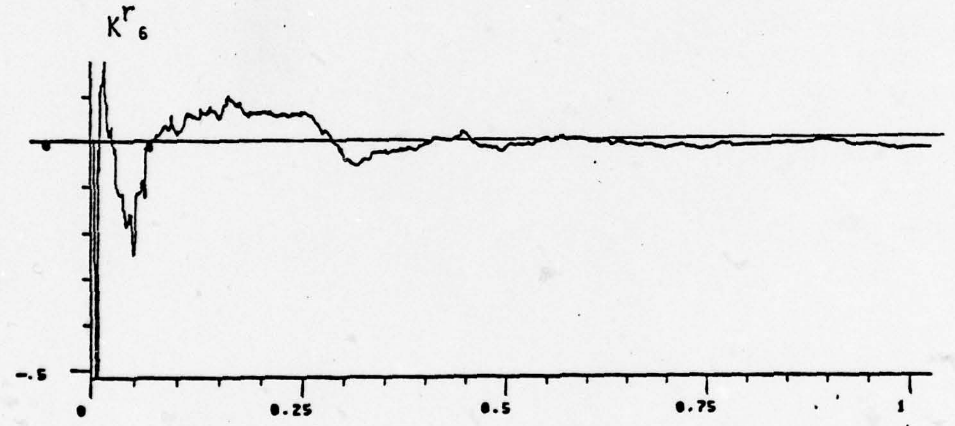


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

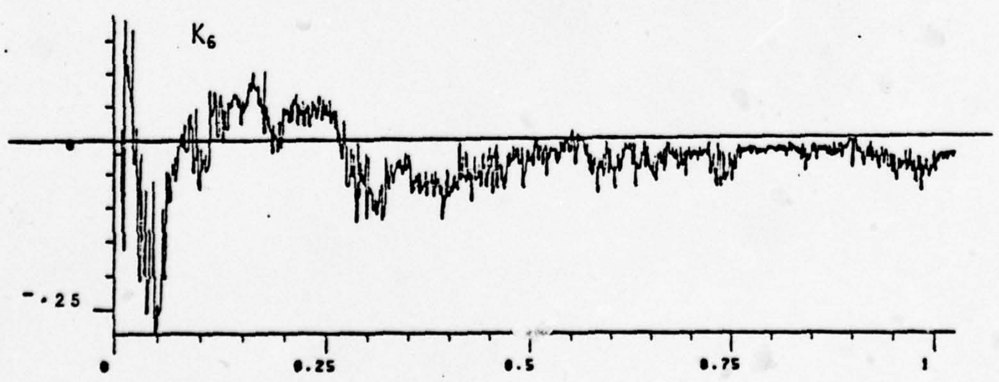
Figure 4.6 Fifth reflection coefficient of 4th order AR process. Bias was observed on all algorithms. Notice the drastic "weeds" appearing in the estimates from Levinson Algorithm without windowing and the smoothing effects of the two windows.



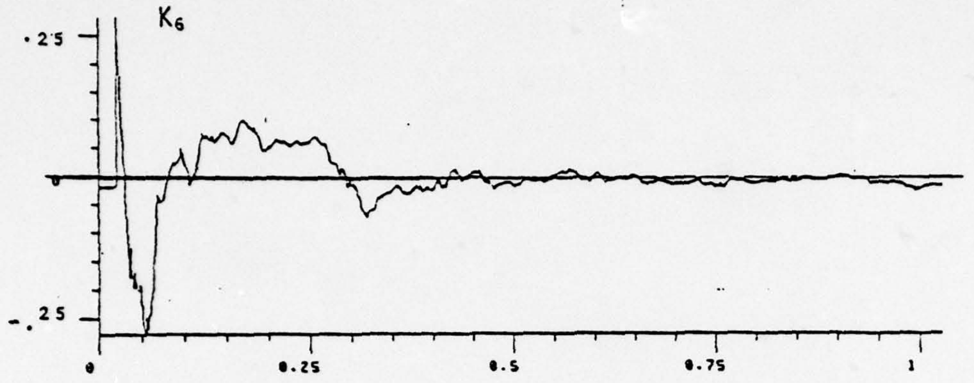
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



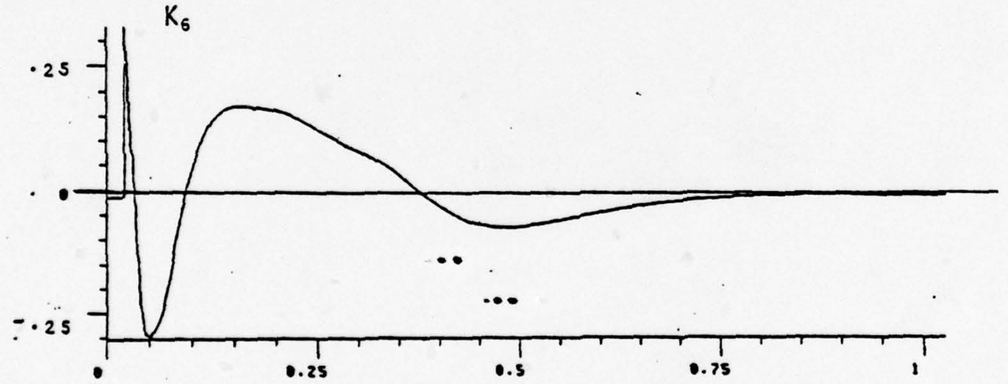
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

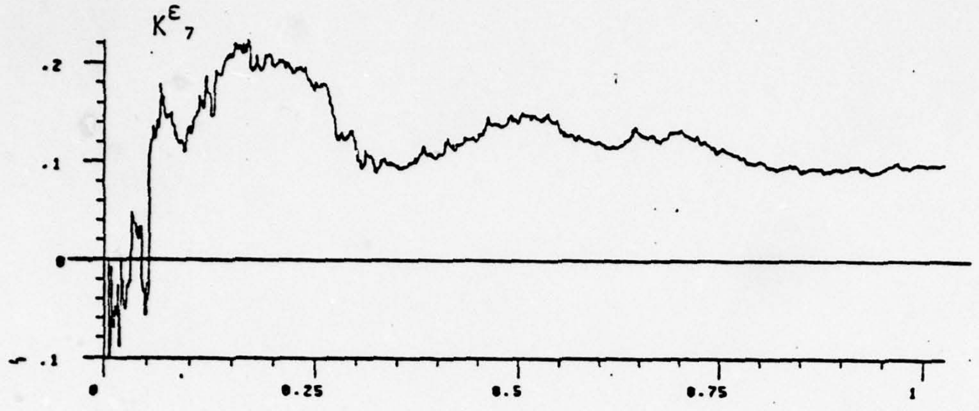


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

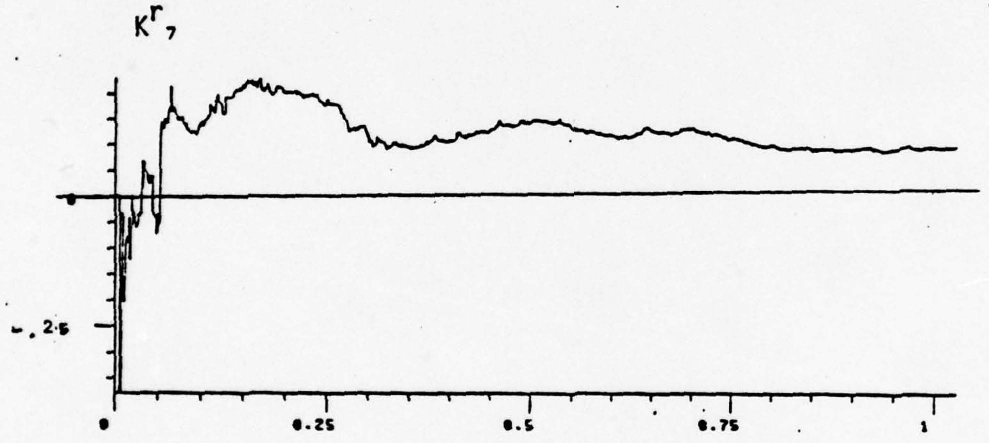


(e) Reflection Coefficient, Levinson Algorithm with Hamming window
Figure 4.7 Sixth order reflection coefficient of 4th order AR process.

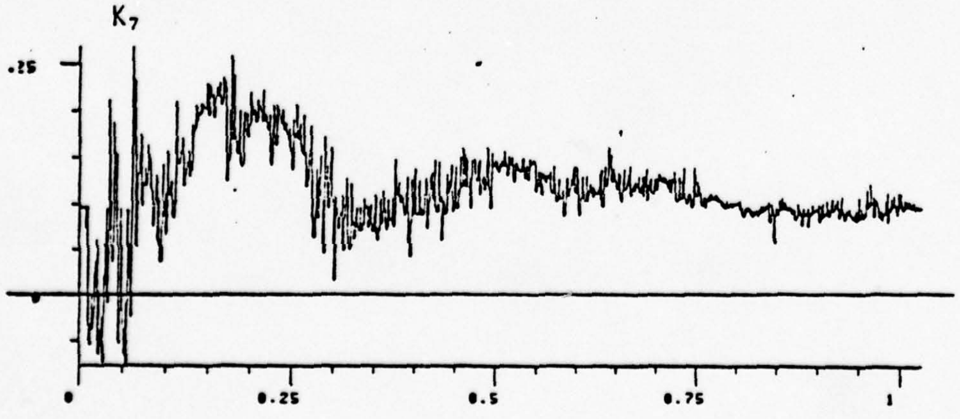
All algorithmsgive estimates that converge to zero, the true value in 100 samples.



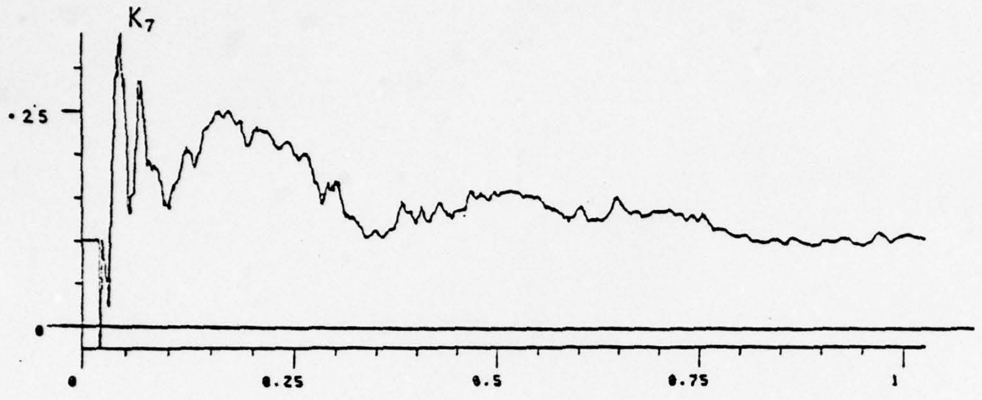
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



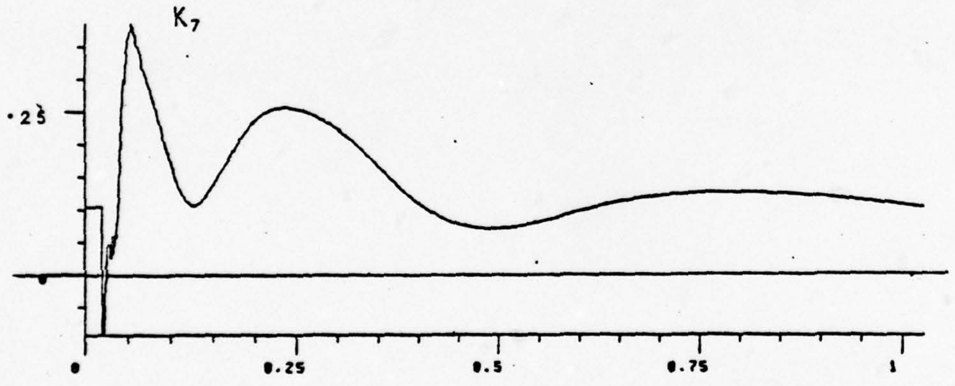
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

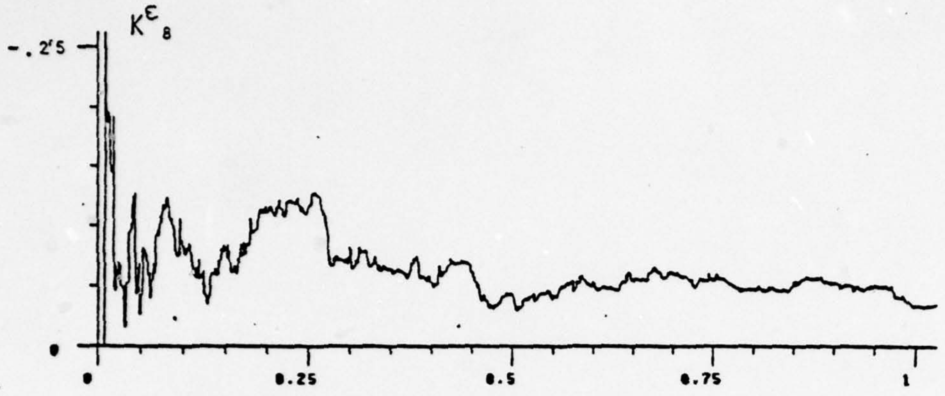


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

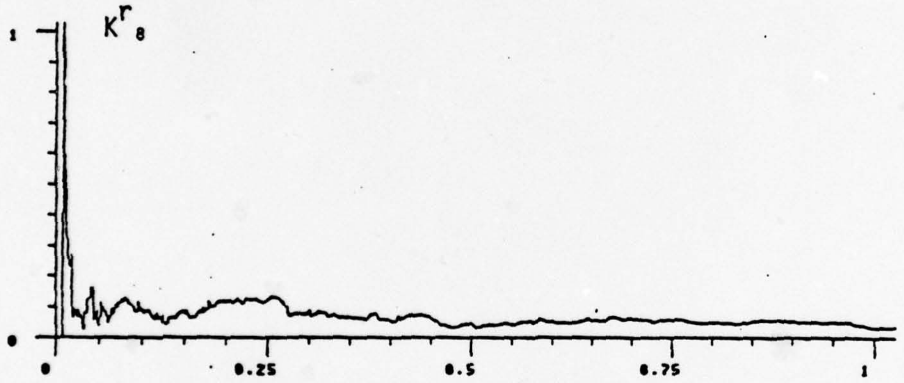


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

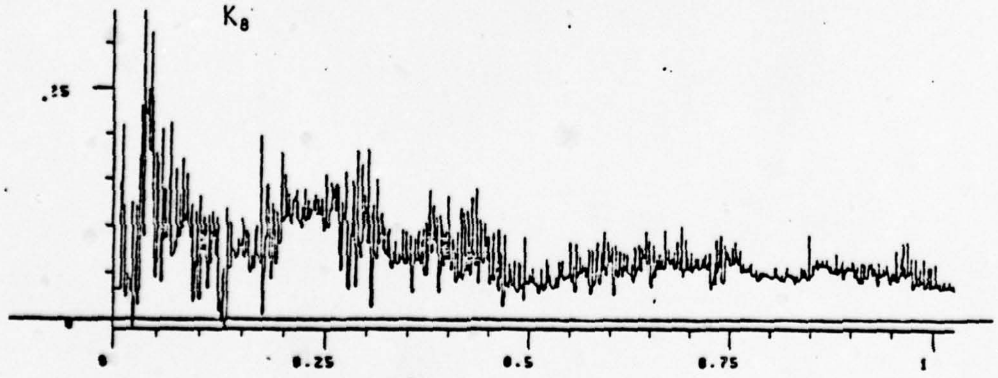
Figure 4.8 Seventh order reflection coefficient of 4th order AR process



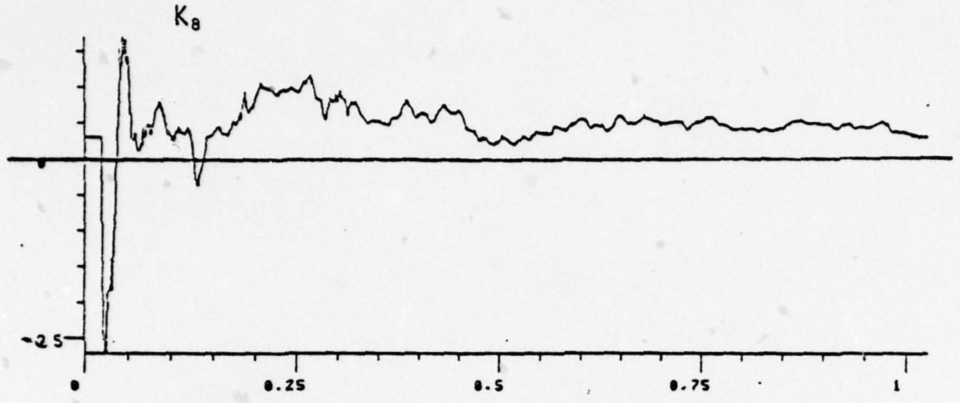
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



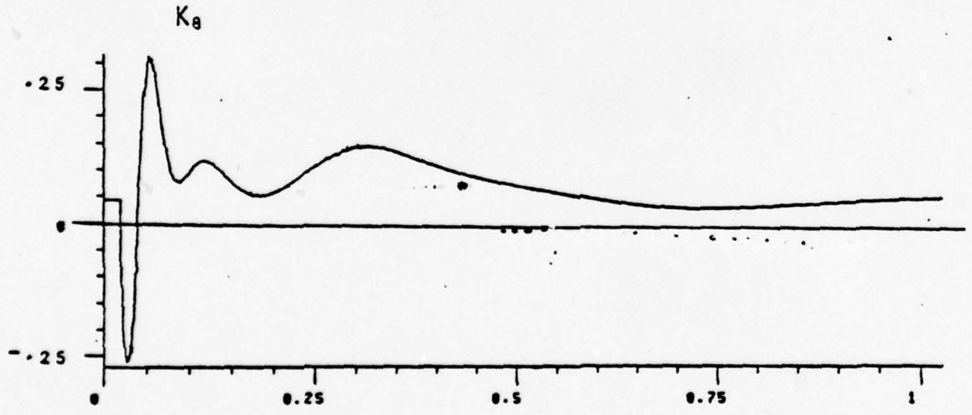
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

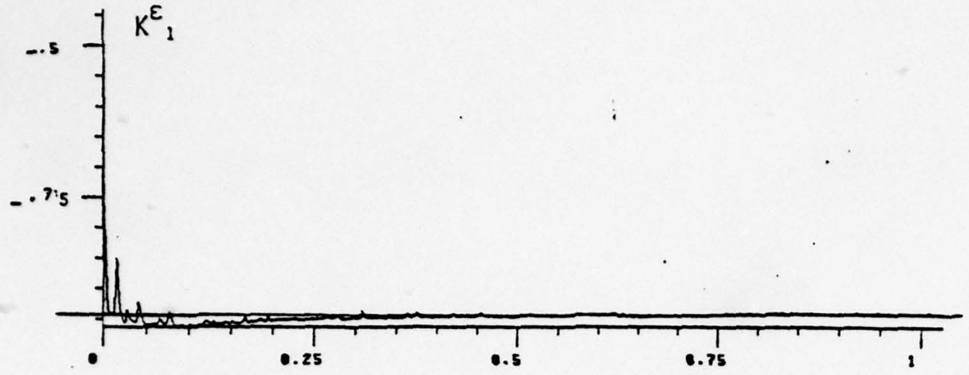


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

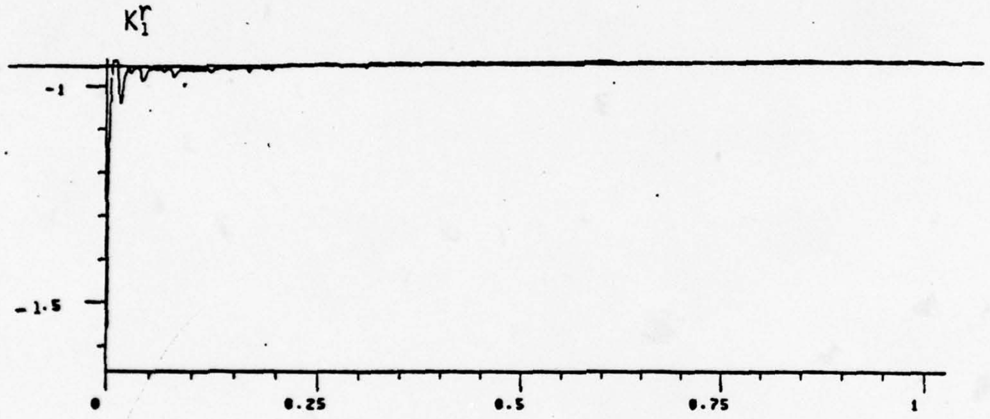


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

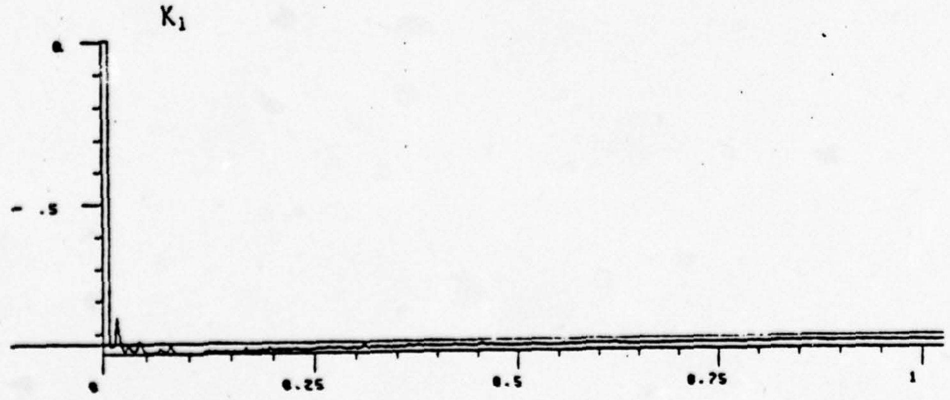
Figure 4.9 Eighth order reflection coefficient of 4th order AR process.



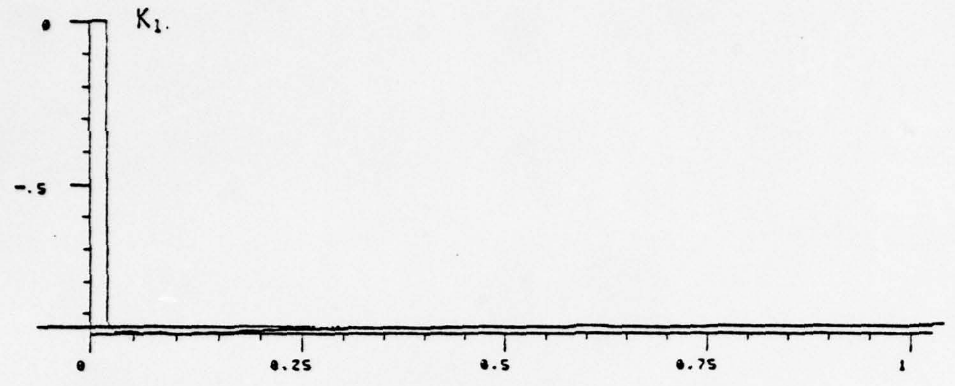
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



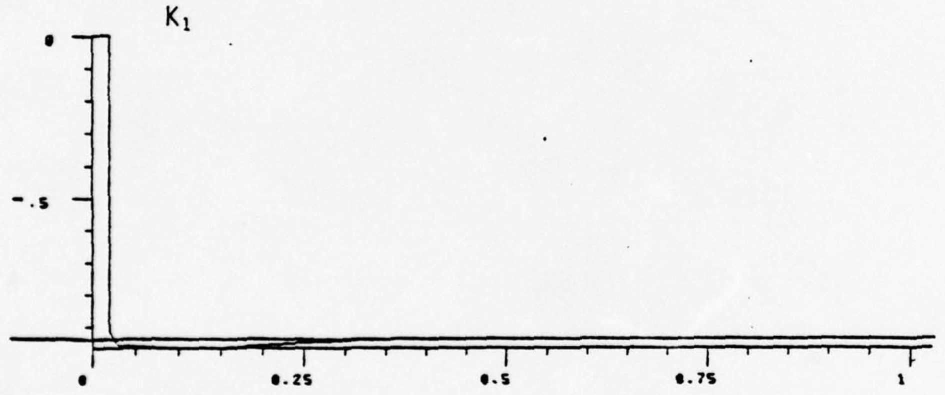
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing



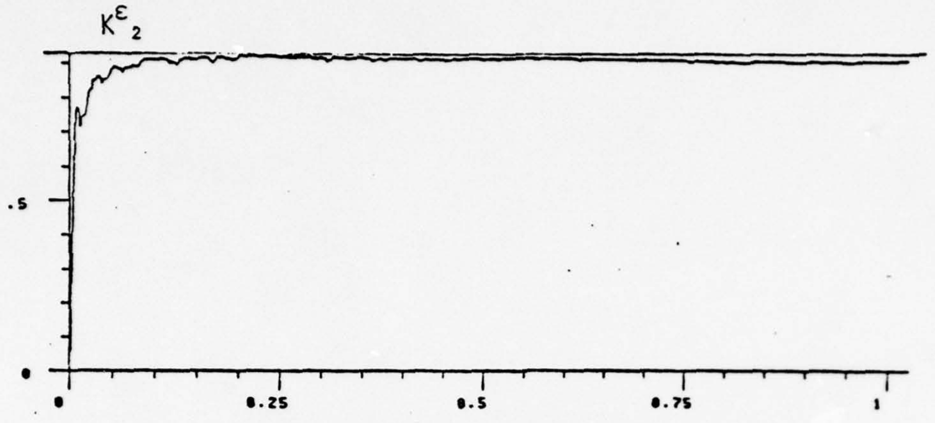
(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window



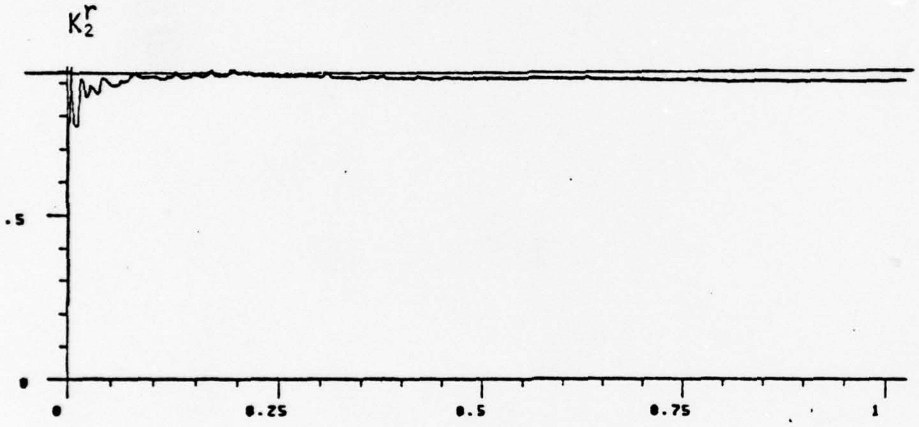
(e) Reflection Coefficient, Levinson Algorithm with Hamming window

Figure 4.10 First order reflection coefficient of 8th order AR process.

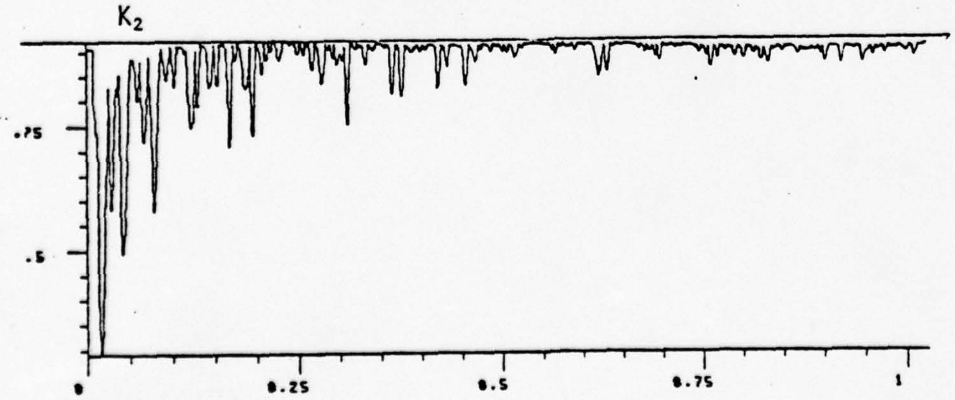
All algorithms give estimates that converge to the true value in 50 samples.



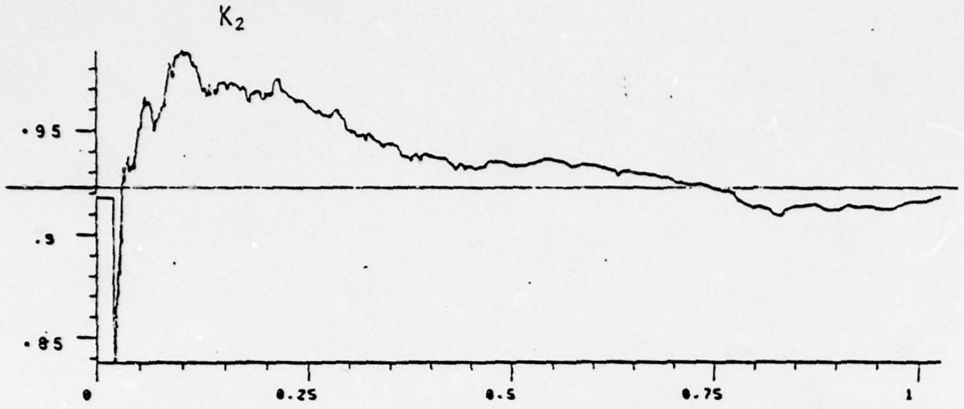
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



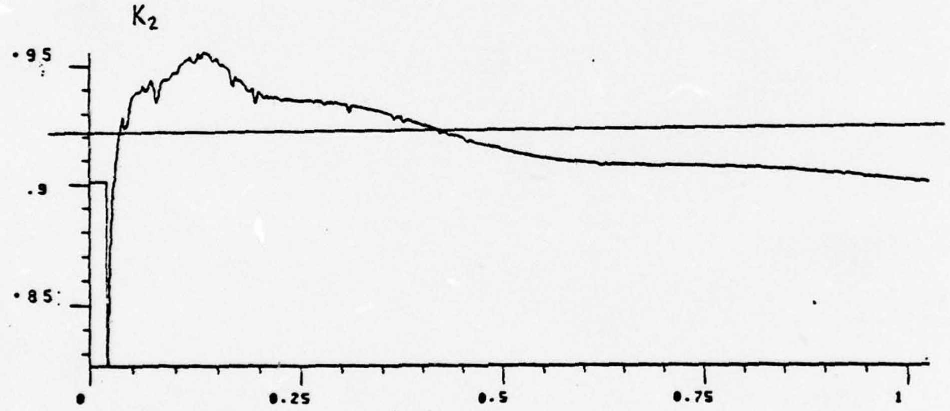
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

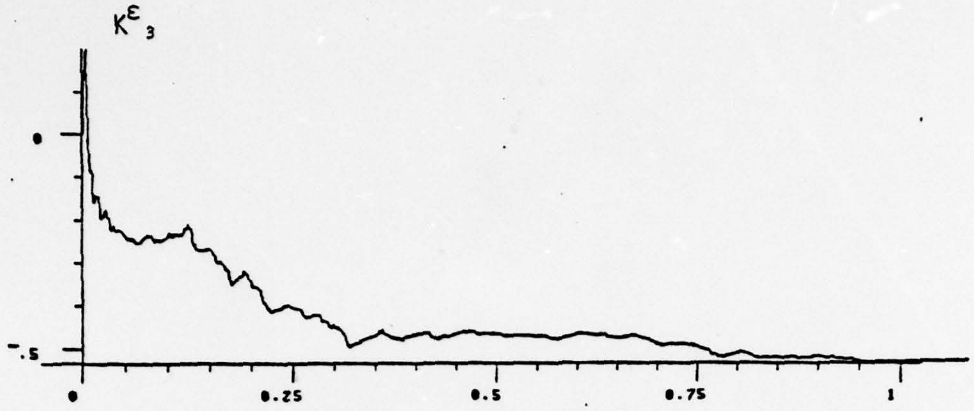


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

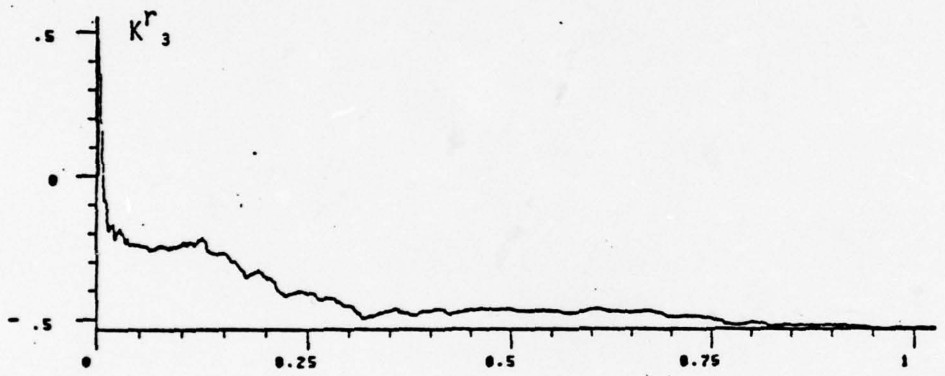


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

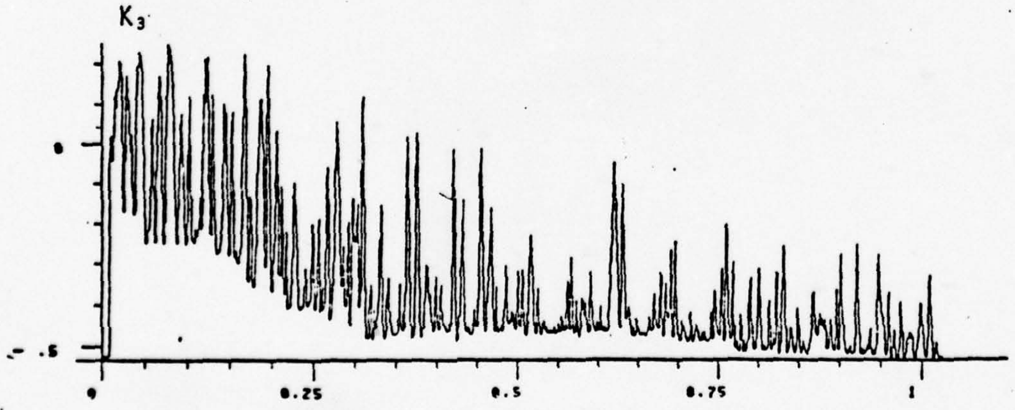
Figure 4.11 Second order reflecton coefficient of 8th order AR process. Both reflection coefficients of the Pre-windowed Ladder Form converge to the true value with 100 samples. In the Levinson algorithm without windowing, the envelope also converges in the same rate. In both of the windowed cases, the initial transients were altered. In the Hamming window case, bias in the estimates are apparent.



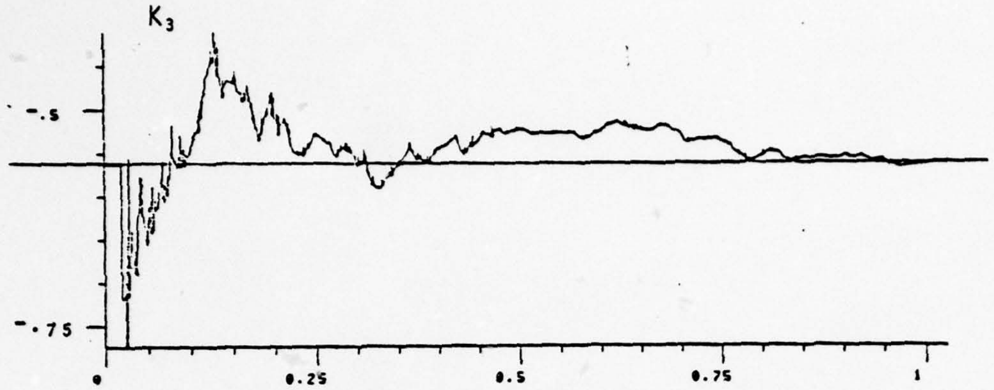
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



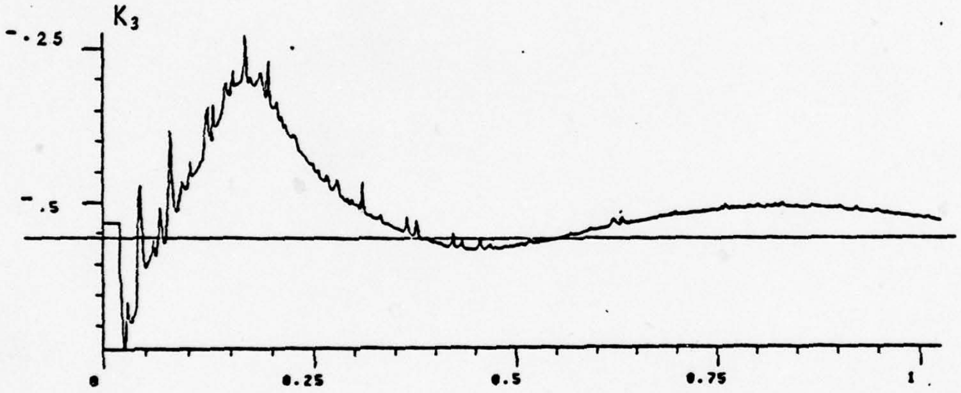
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

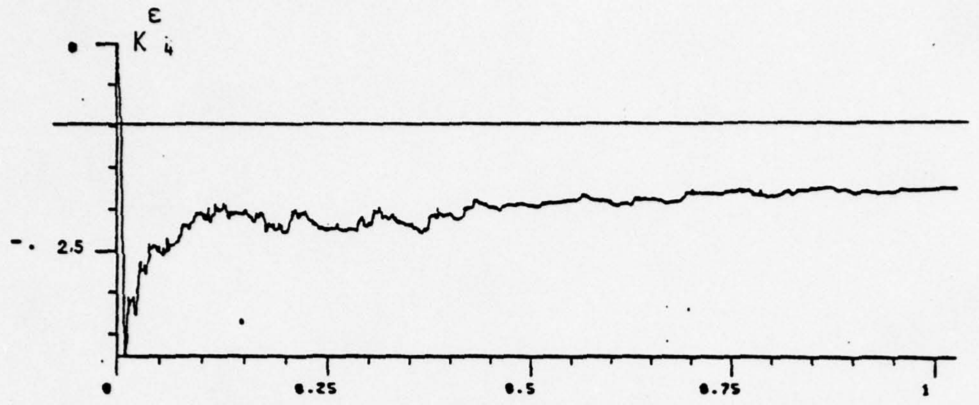


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

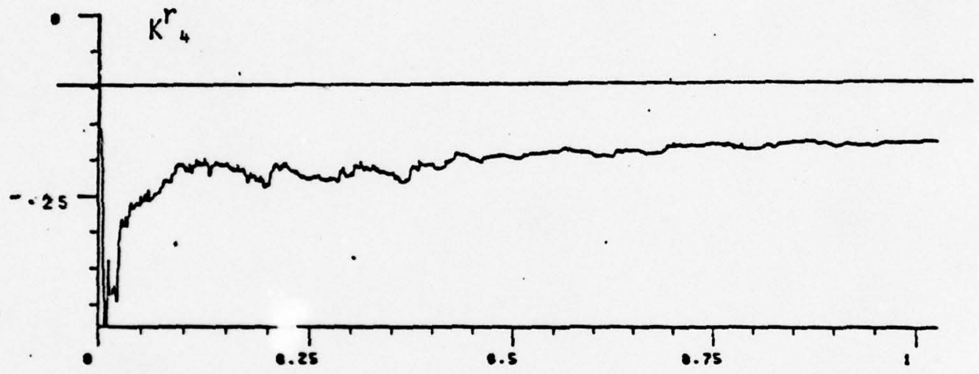


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

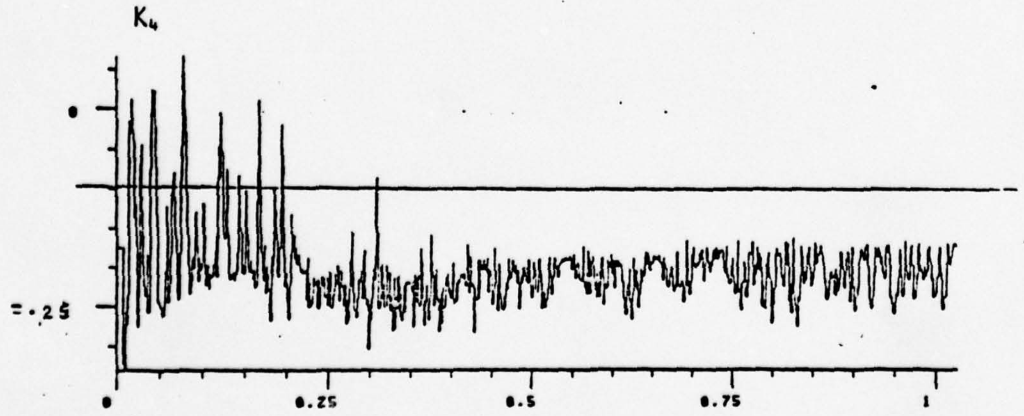
Figure 4. 12 Third order reflection coefficient of 8th order AR process. Notice the drastic effects of "weeds" in the estimates of Levinson Algorithm with windowing.



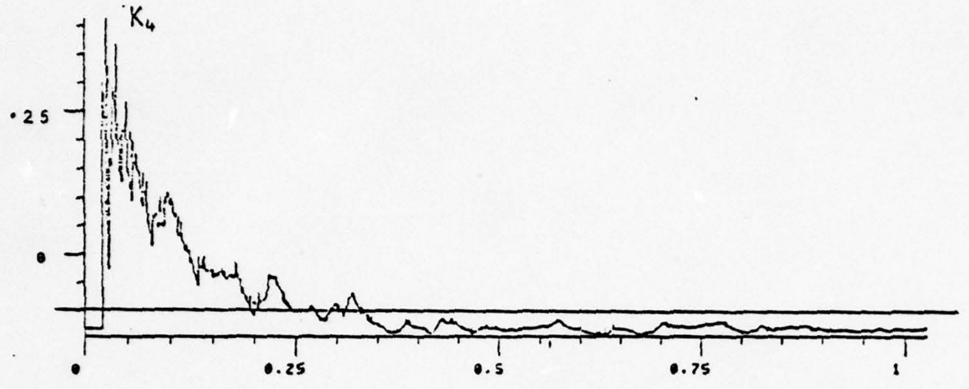
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



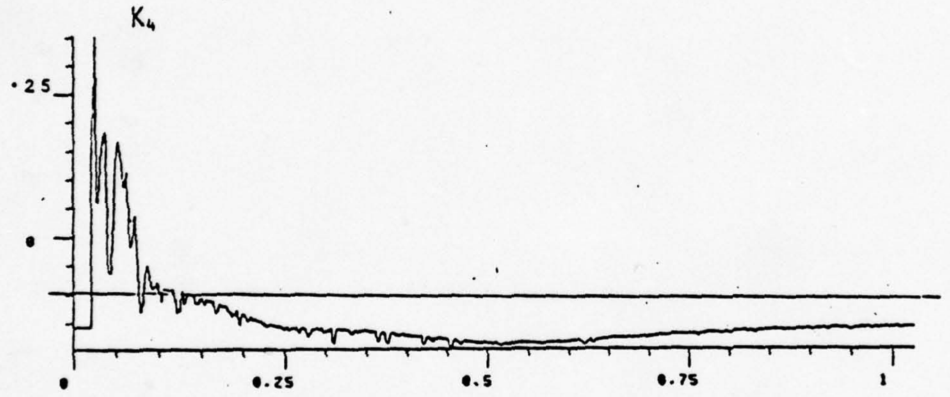
(b) Backward Reflecton Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without Windowing

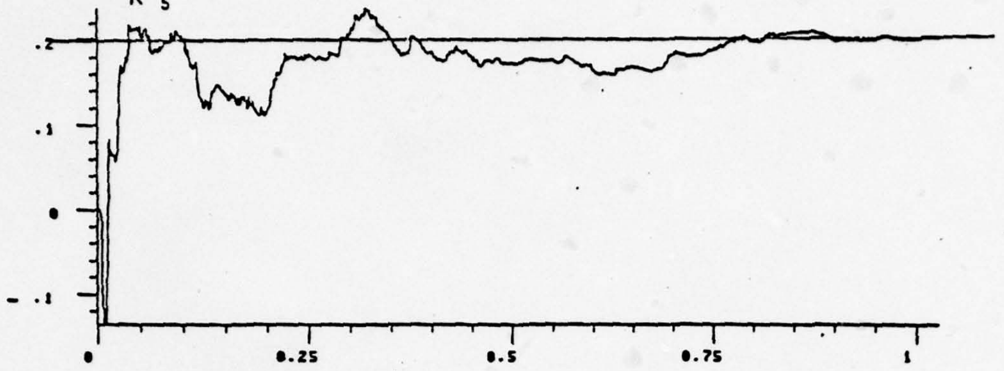


(d) Reflection Coefficients, Levinson Algorithm with Trapezoidal window

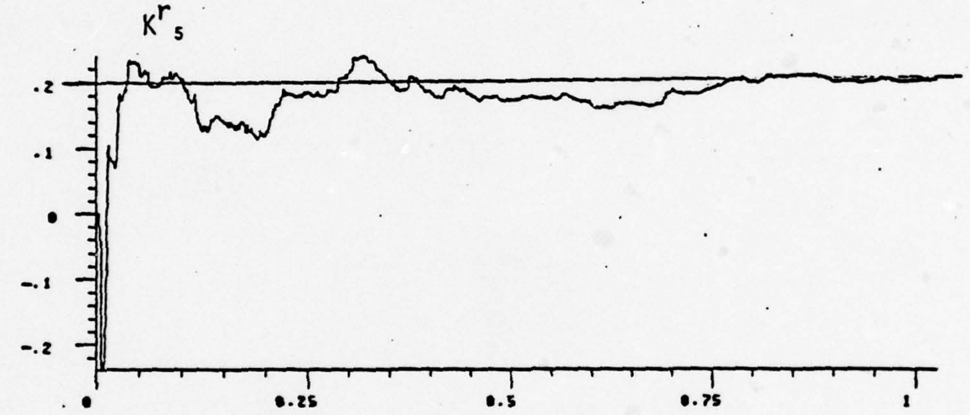


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

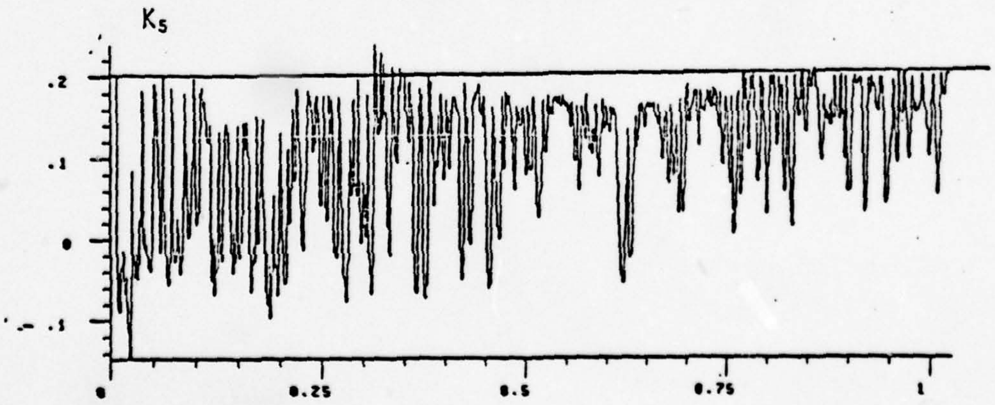
Figure 4.13 Fourth order reflection coefficient of 8th order AR process Bias is observed for all estimates.



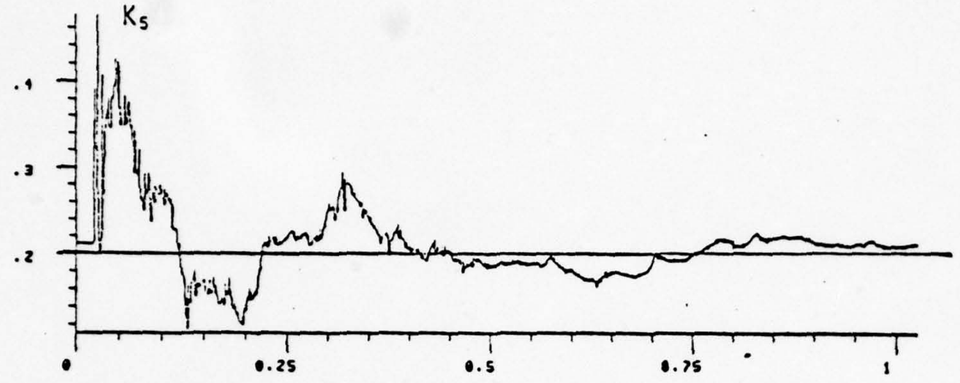
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



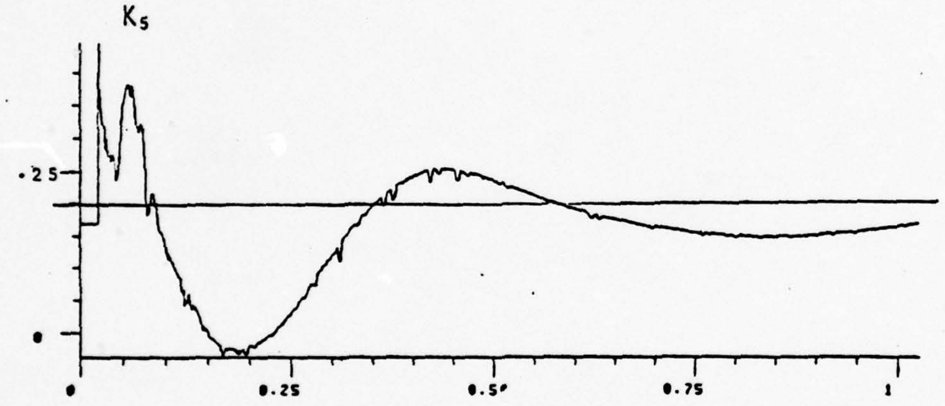
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

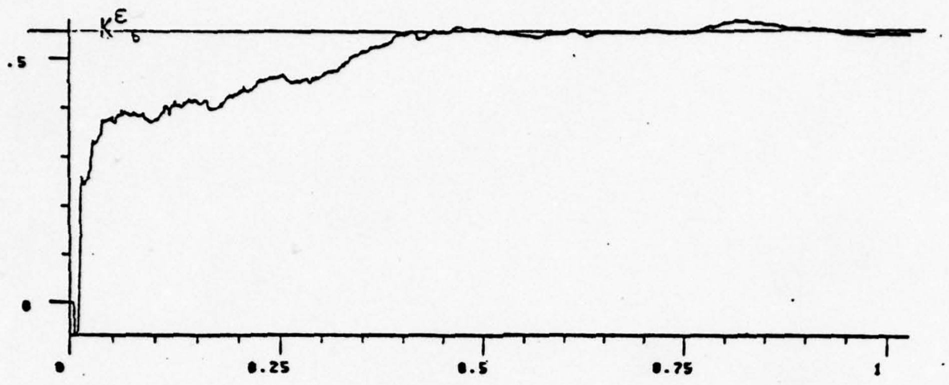


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

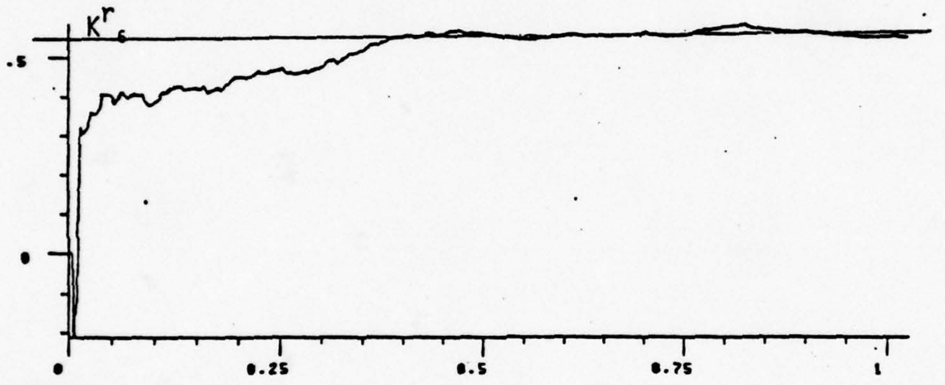


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

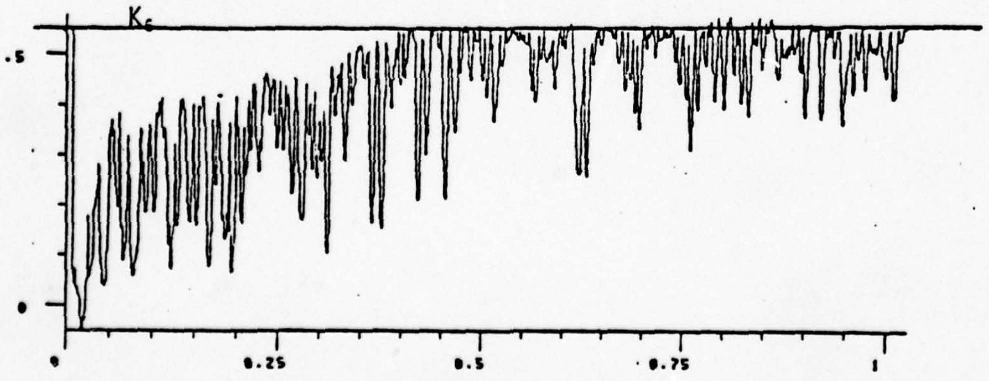
Figure 4.14 Fifth order reflection coefficient of 8th order AR process. Notice the bias in the windowed estimates.



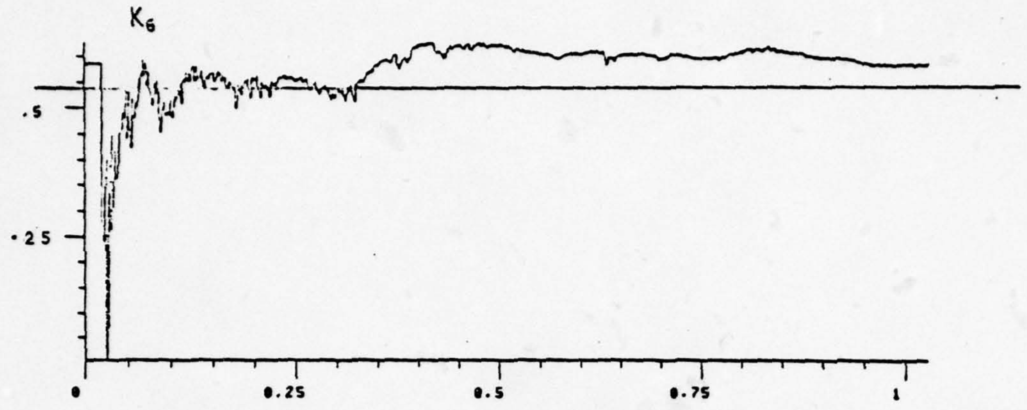
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



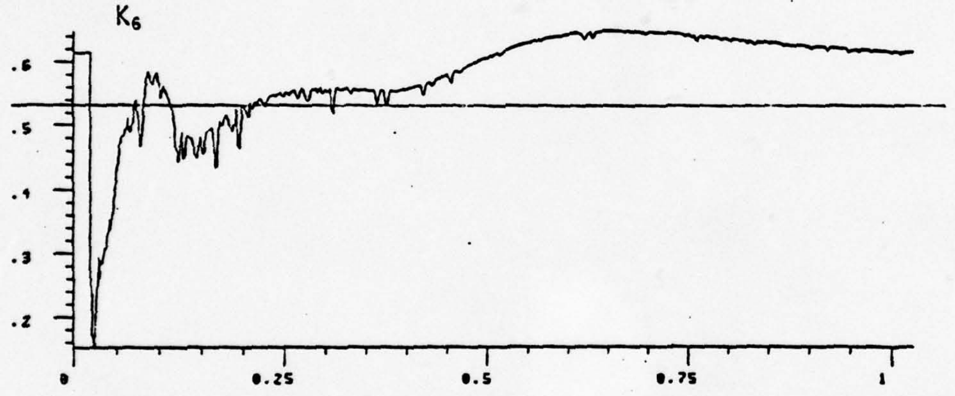
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

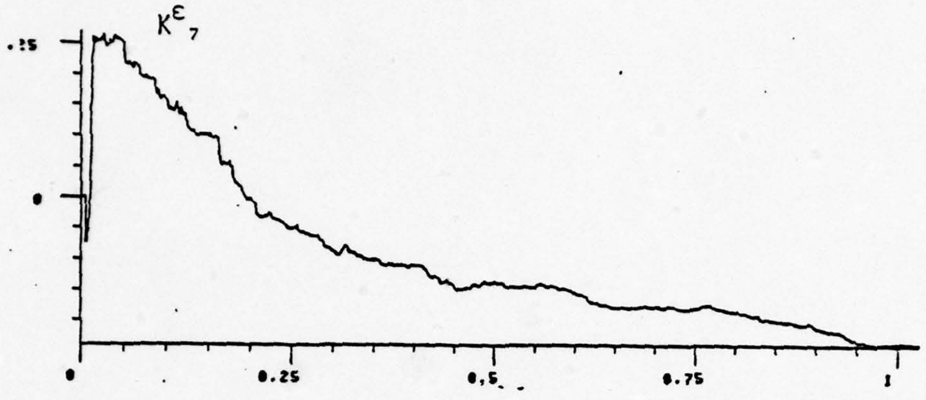


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

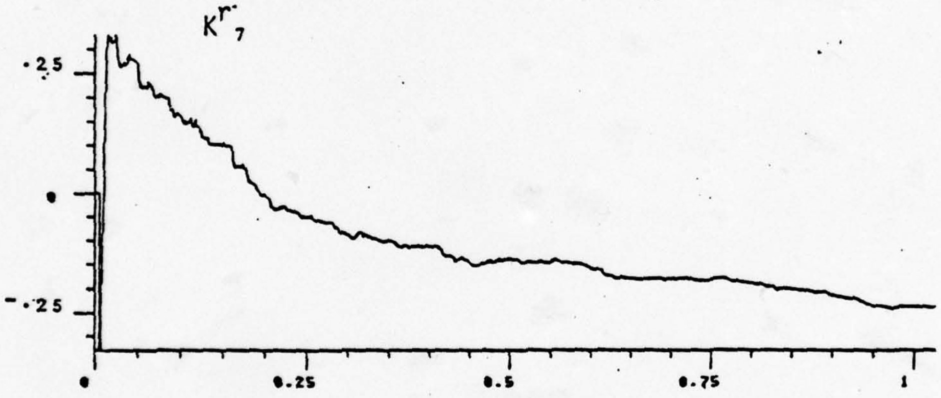


(e) Reflect on Coefficient, Levinson Algorithm with Hamming window

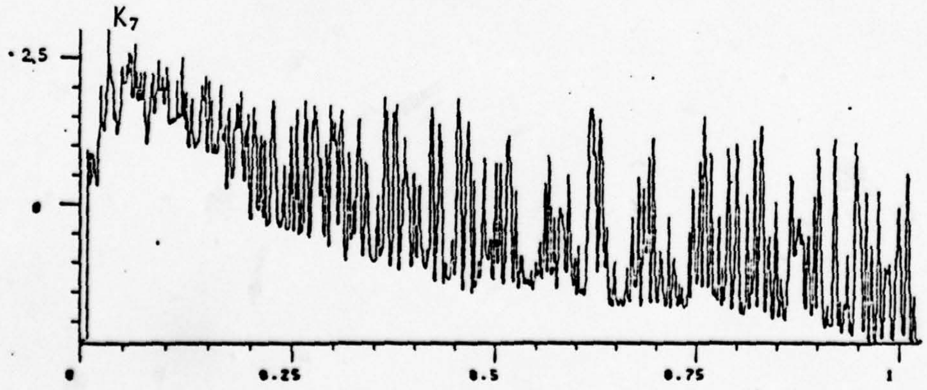
Figure 4.15 Sixth order reflection coefficient of 8th order AR process. Notice the bias introduced into the estimates in the windowed cases.



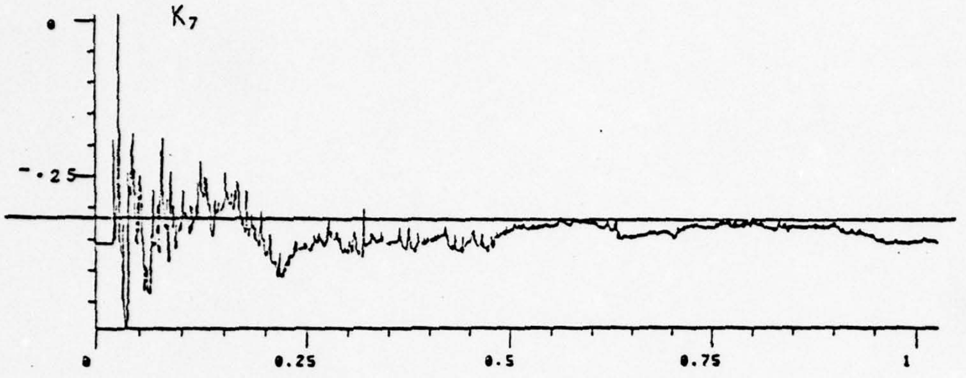
(a) Forward Reflecton Coefficient, Pre-windowed Ladder Form



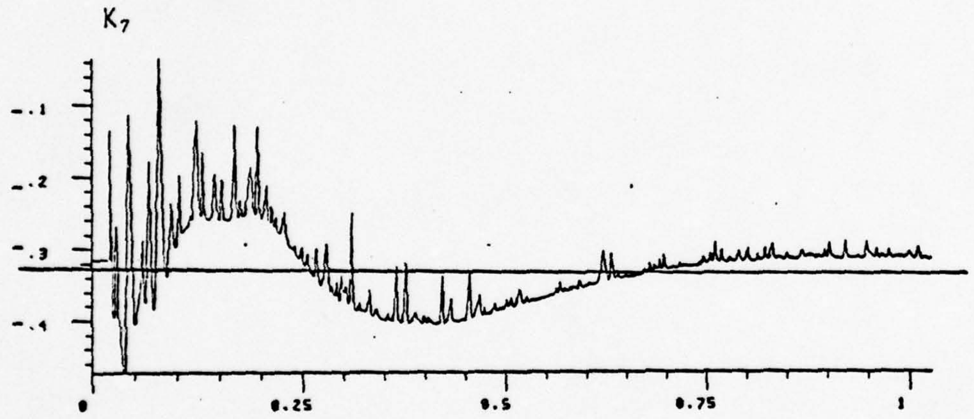
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing

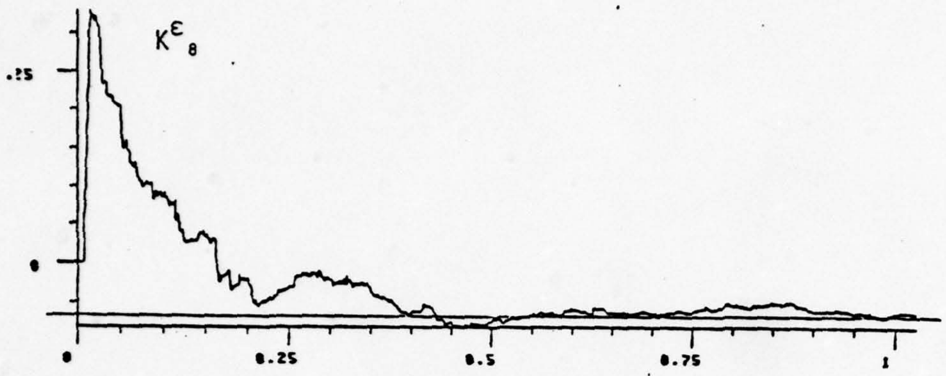


(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window

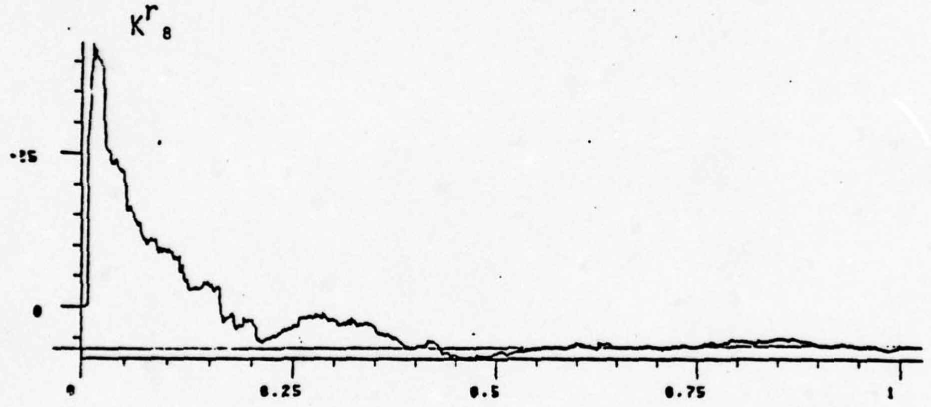


(e) Reflection Coefficient, Levinson Algorithm with Hamming window

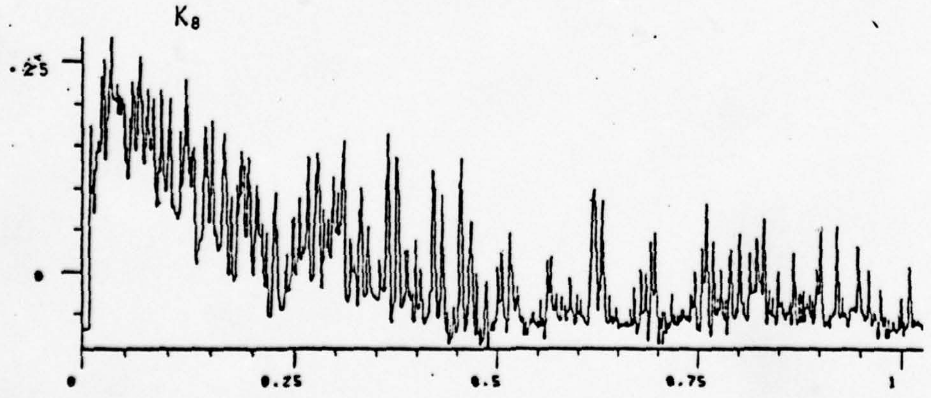
Figure 4.16 Seventh Order reflection coefficient of 8th order AR process. Notice that even in the windowed cases, "weeds" are still present in the estimates.



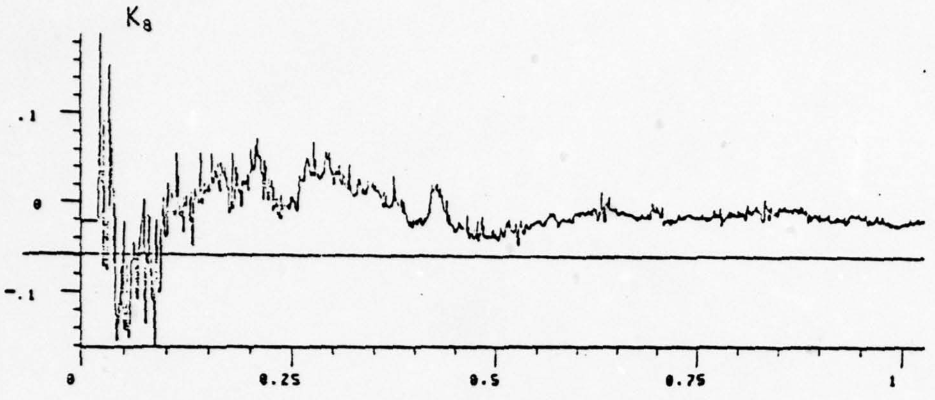
(a) Forward Reflection Coefficient, Pre-windowed Ladder Form



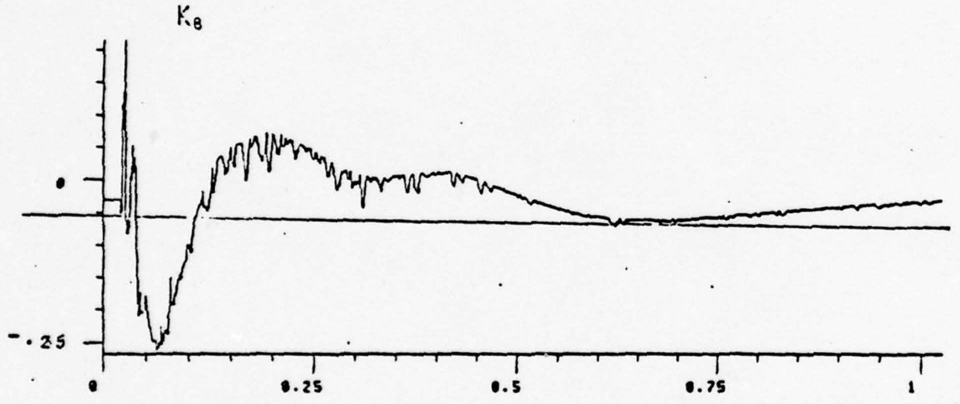
(b) Backward Reflection Coefficient, Pre-windowed Ladder Form



(c) Reflection Coefficient, Levinson Algorithm without windowing.



(d) Reflection Coefficient, Levinson Algorithm with Trapezoidal window



(e) Reflection Coefficient, Levinson Algorithm with Hamming window

Figure 4.17 Eighth order reflection coefficient of 8th order Ar process.

5.2 Time-Varying AR models

5.2.1 Results on Simulated Vocal Tract Behavior

This set of tests is to demonstrate the tracking ability of the PW ladder form. Two sets of data were generated from an 8th order AR model with time-varying (piecewise constant) reflection coefficients that converge exponentially to zero. The first set of data was obtained by driving the input with Gaussian white noise only. The second was obtained by driving the input with Gaussian white noise plus a pulse train. The pulses occur at the step changes of the reflection coefficients, and thus closely simulate actual vocal tract and speech behaviour. Figure 1(a) and 1(b) show the actual data. Figures 2 - 9 show the convergence of all eight of the reflection coefficients. Figure 10 shows the behavior of the likelihood quantity gamma7, suggesting its possible role in a pitch detection scheme.

The data sets are each 2048 samples long. The step changes occur at every 128 samples. The time-constant of the tracking rate was set at 100, i.e. at about the rate of the changes.

Observations

Observe that for the unvoiced data, the estimated reflection coefficients all tend to fluctuate when they are close to zero. While for voiced data, the estimated reflection coefficients converge uniformly to the true piecewise constant behavior. A carefull study of the log-likelihood function as described in the context of pitch detection and the conditioning of the underlying covariance matrix explains these phenomena. The log-likelihood function is proportional to $ln(1 - K_i^2)$, i.e. if the magnitude of the K_i's are close to one, the log-likelihood is a very sensitive function or conversely if the K's are small they are not very important, therefore they are harder to estimate with a least-squares criterion. An other observation is that if the very first sample is a pulse, this implies that the problem is ill-conditioned. Simulations show that if the first few samples are gaussian type signals, the convergence is much improved. This typically is the the case after a "numerical reset", i.e. ladder recursions recover rapidly after a reset.

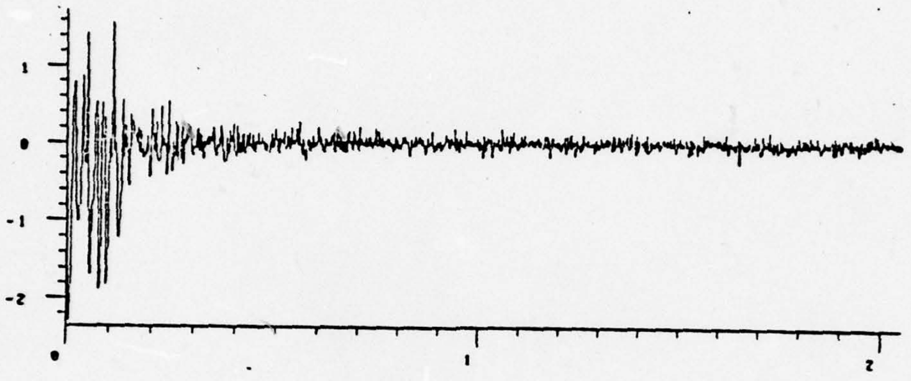


Fig. 1(a) An 8th order AR process with time-varying (piecewise constant) reflection coefficients that converge exponentially to zero. Process was driven by gaussian white noise only.

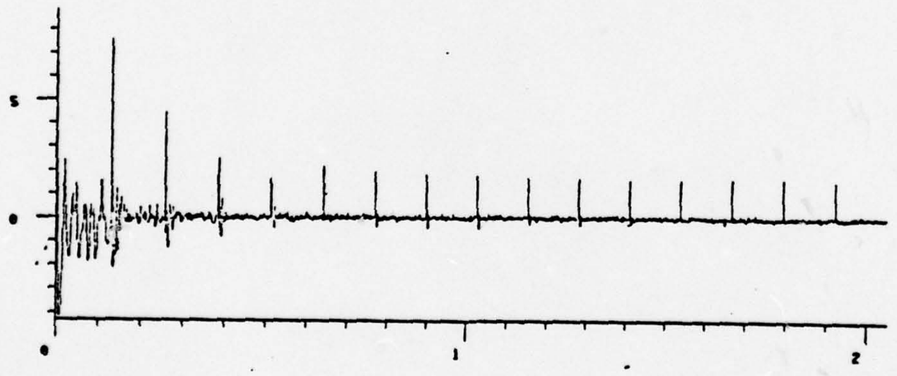


Fig. 1(b) Same 8th order AR process but driven by sum of gaussian white noise plus an impulse train.

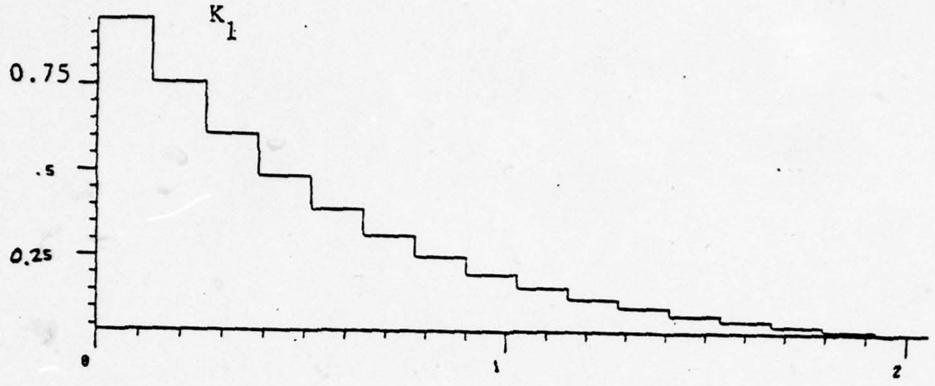


Fig.2(a) First reflection coefficient, K_1, of underlying model.

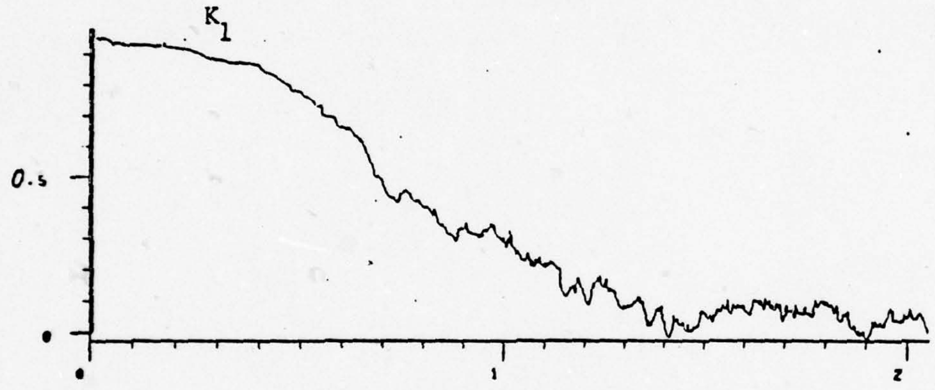


Fig.2(b) Estimated K_1 from artificial unvoiced data.

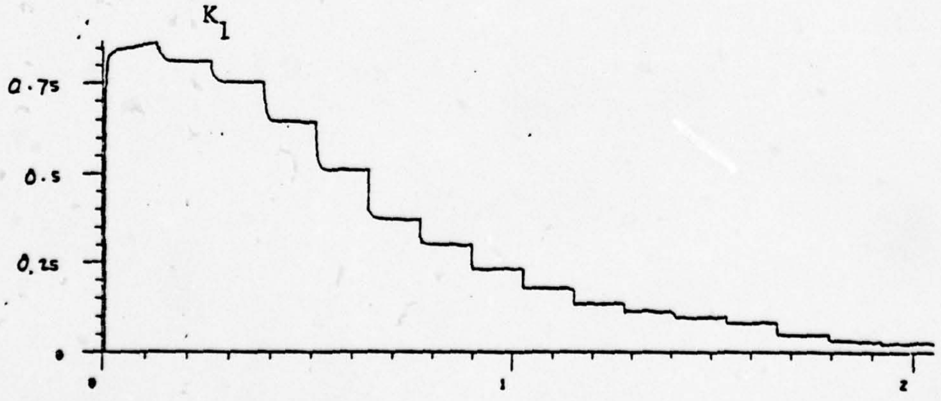


Fig.2(c) Estimated K_1 from artificial voiced speech data.

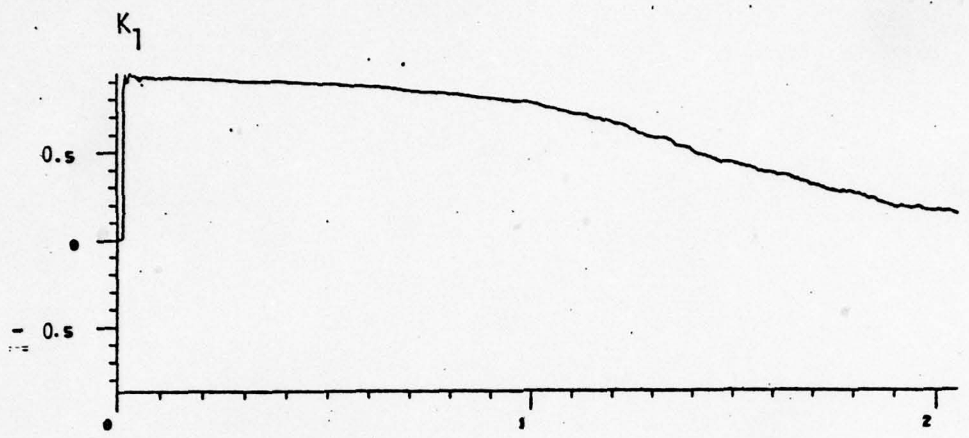
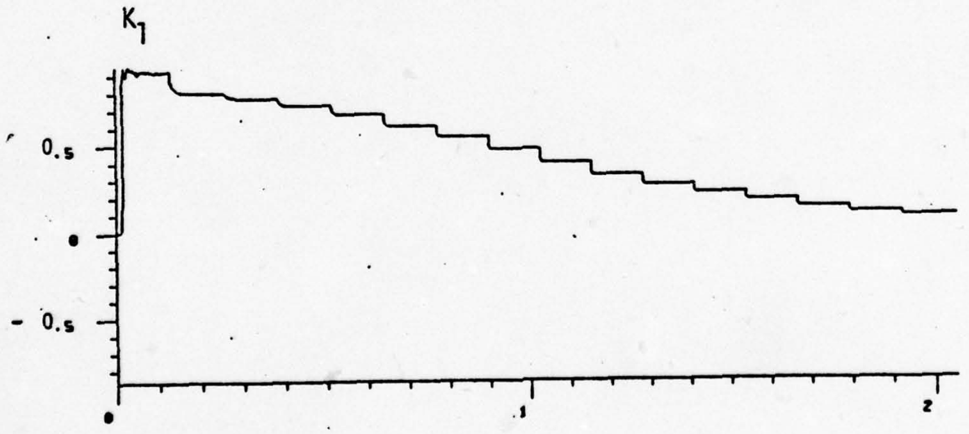


Fig. 2(d). Estimate of K_1 using time-varying weighting factor on artificial unvioced data.



Fif. 2(e). Estimate of K_1 using time-varying weighting factor on artificial voiced data.

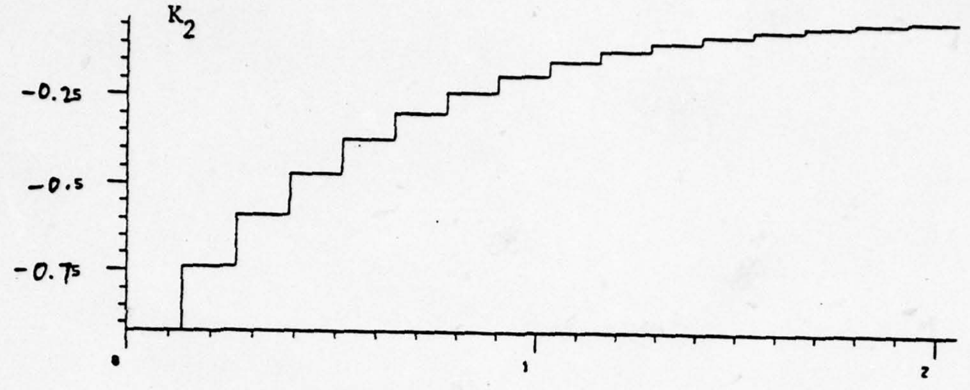


Fig.3(a) Second reflection coefficient, K_2, of underlying model.

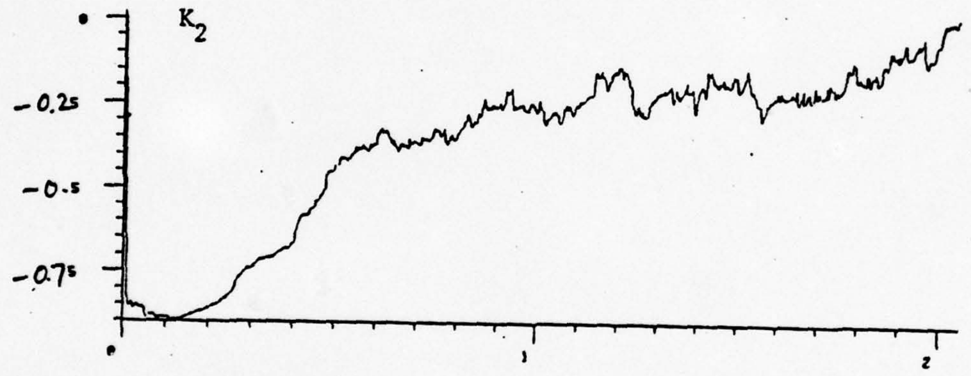


Fig.3(b) Estimated K_2 from driving function of white gaussian noise only.

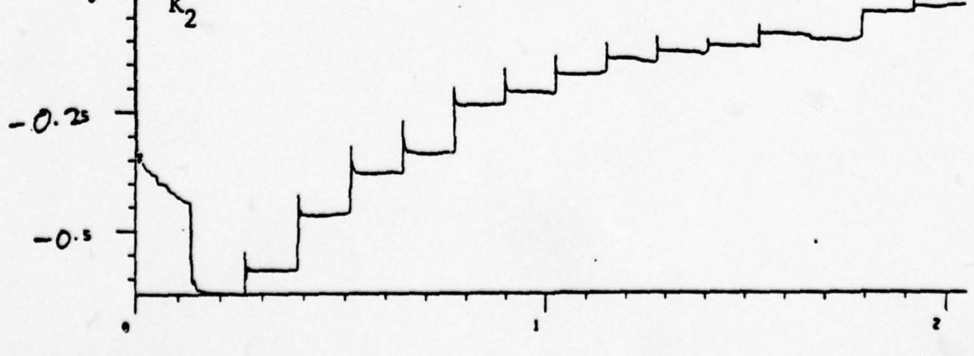


Fig.3(c) Estimated K_2 from impulse plus white gaussian noise input.

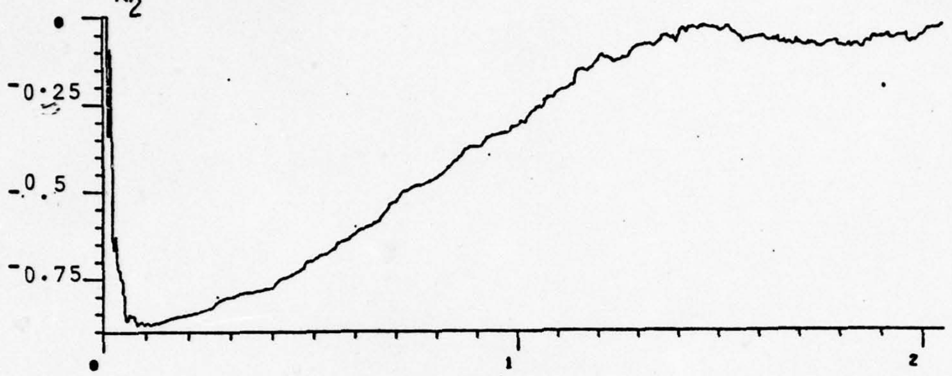


Fig. 3(d). Estimated K_2 of gaussian input onlu, using time-varying weighting factor.

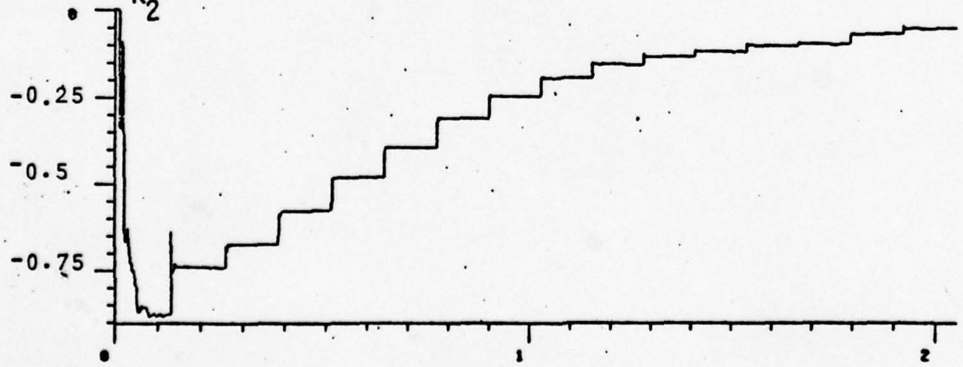


Fig. 3(e) Estimated K_2 of non-gaussian input, using time-varying weighting factor.

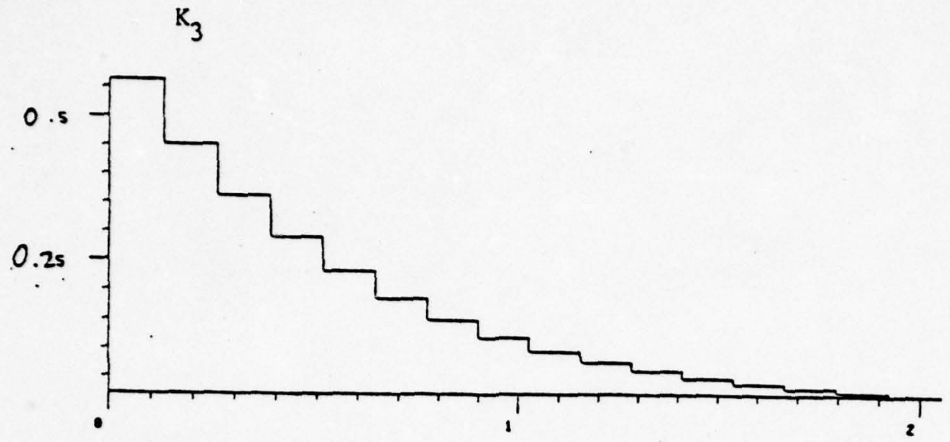


Fig.4(a) Third reflection coefficient, K_3, of underlying model.

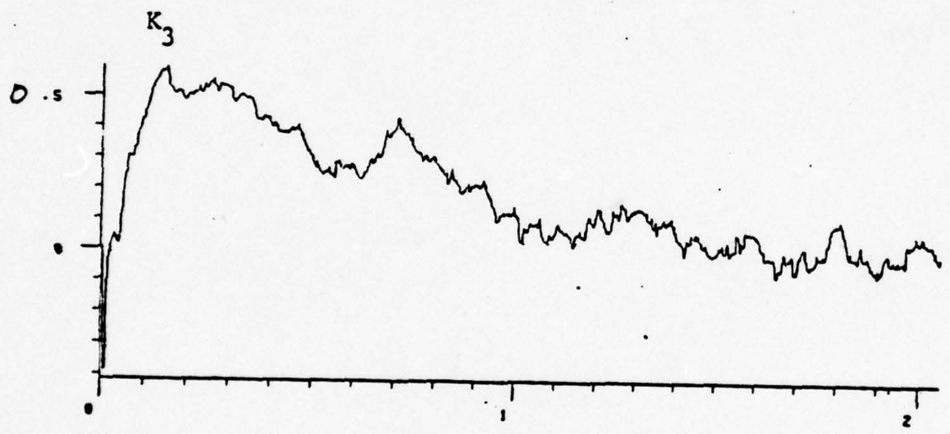


Fig.4(b) Estimated K_3 of unvoiced data (generated by white noise only).

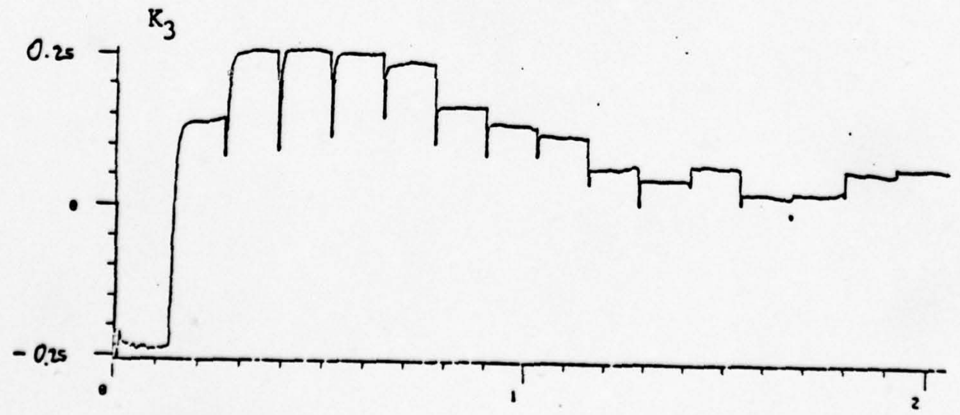


Fig.4(c) Estimated K_3 from voiced data (generated by impulse train plus white noise).

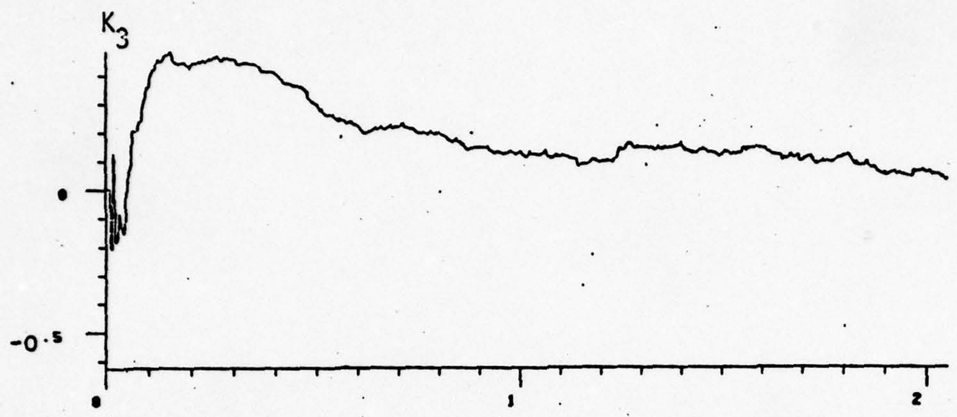


Fig. 4(d). Estimated K_3 of gaussian input only using time-varying weighting factor.

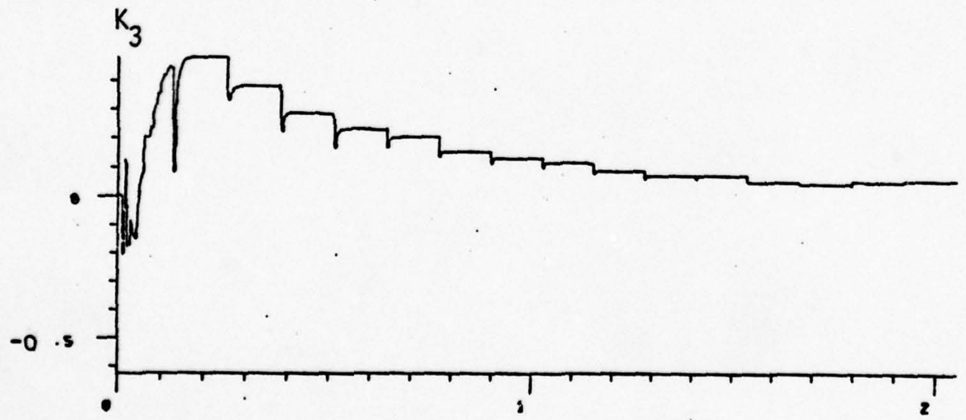


Fig. 4(e). Estimated K_3 of non-gaussian input, using time-varying weighting factor.

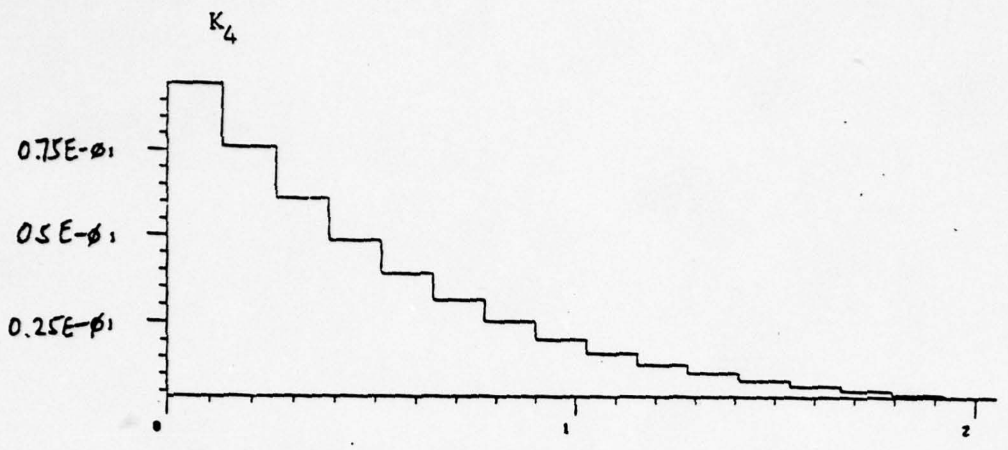


Fig.5(a) Fourth reflection coefficient, K_4, of underlying model.

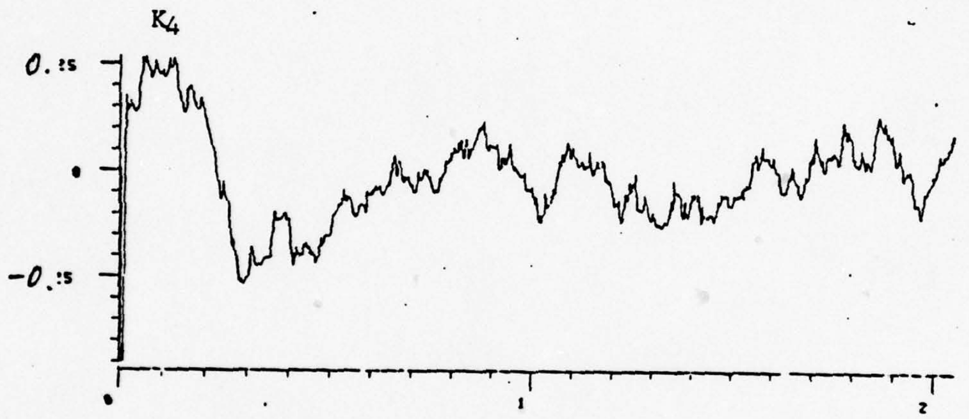


Fig.5(b) Estimated K_4 of unvoiced data (generated by white noise only).

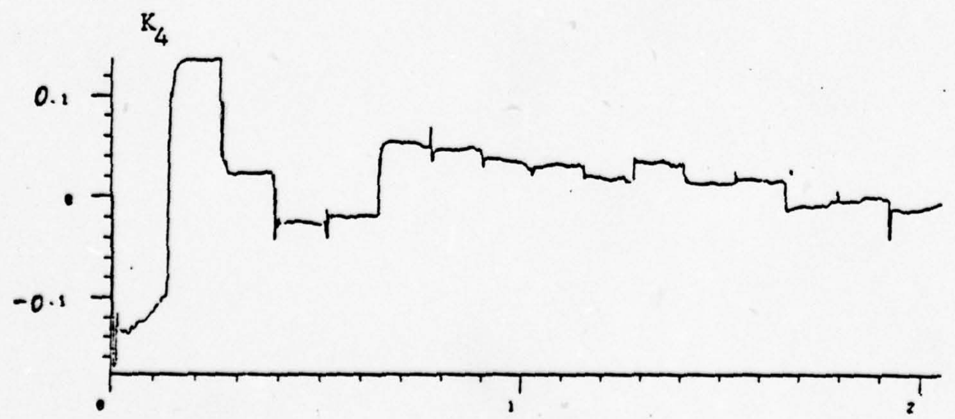


Fig.5(c) Estimated K_4 of voiced data (generated by impulse train plus white noise).

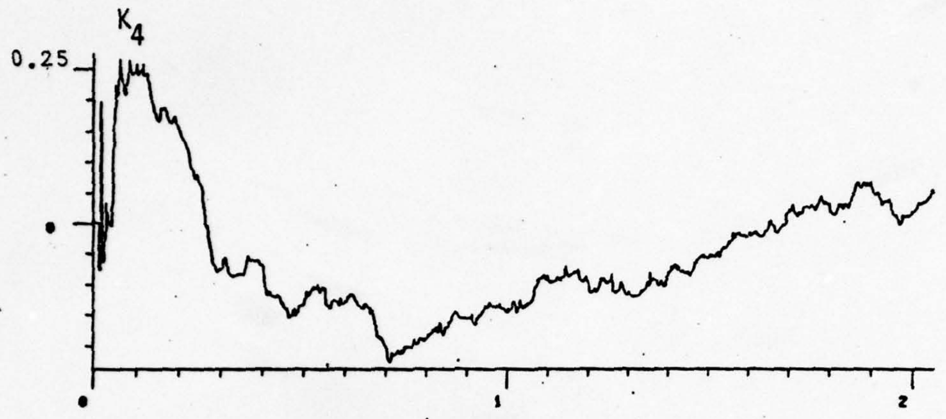


Fig. 5(d). Estimated K_4 of gaussian input only using time-varying weighting factor.

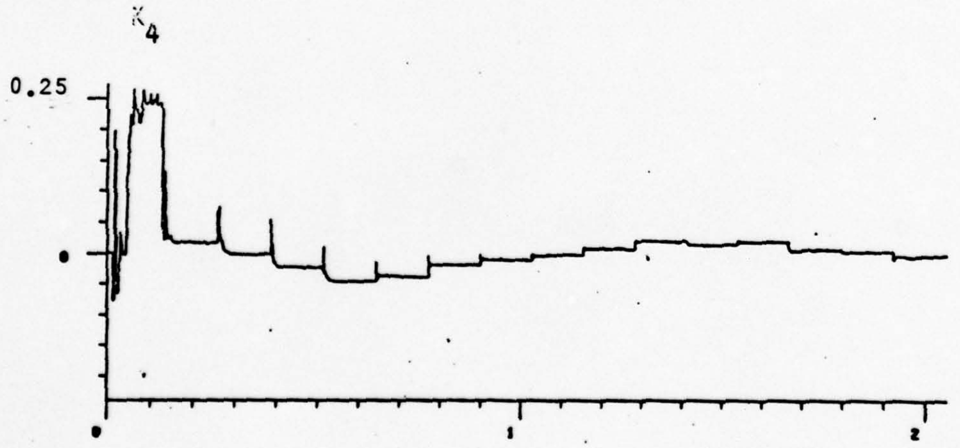


Fig. 5(e). Estimated K_4 of non-gaussian input using time-varying weighting factor.

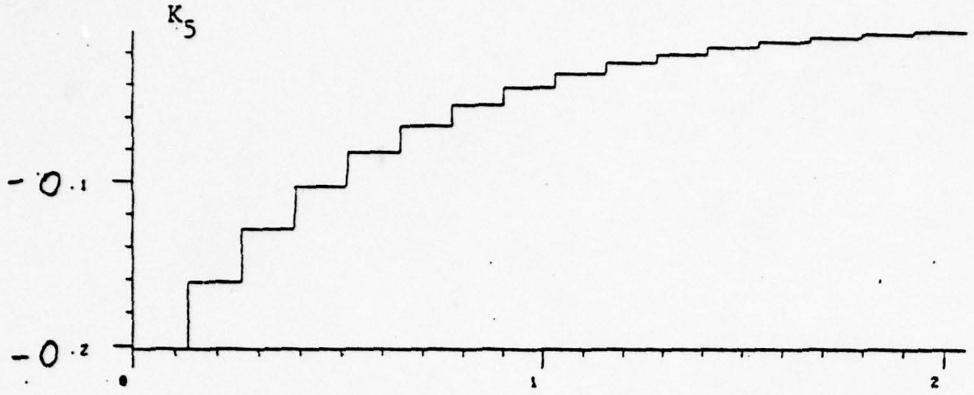


Fig.6(a) Fifth reflection coefficient, K_5, of underlying model.

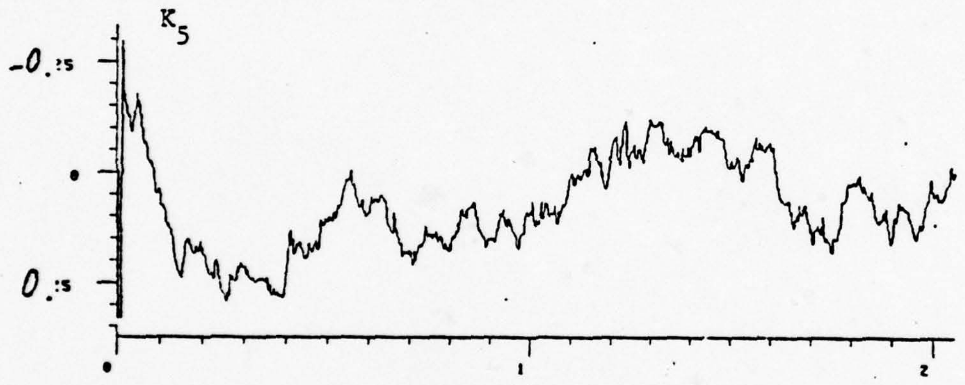


Fig.6(b) Estimated K_5 of unvoiced data (generated by white noise only).

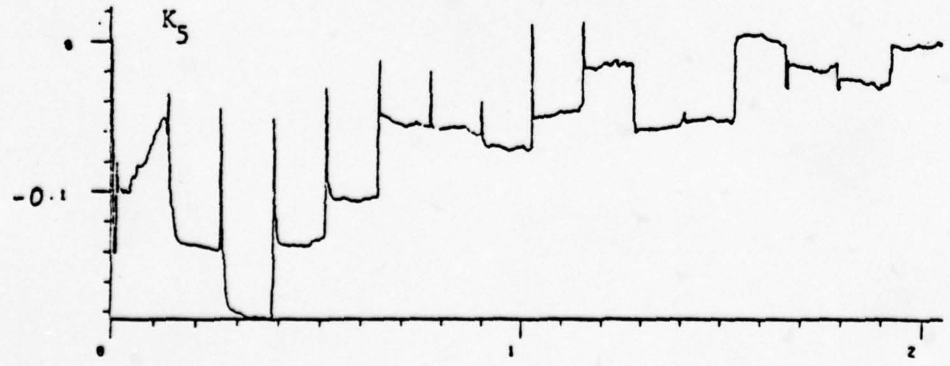


Fig.6(c). Estimated K_5 of voiced data (generated by Impulse train plus white noise).

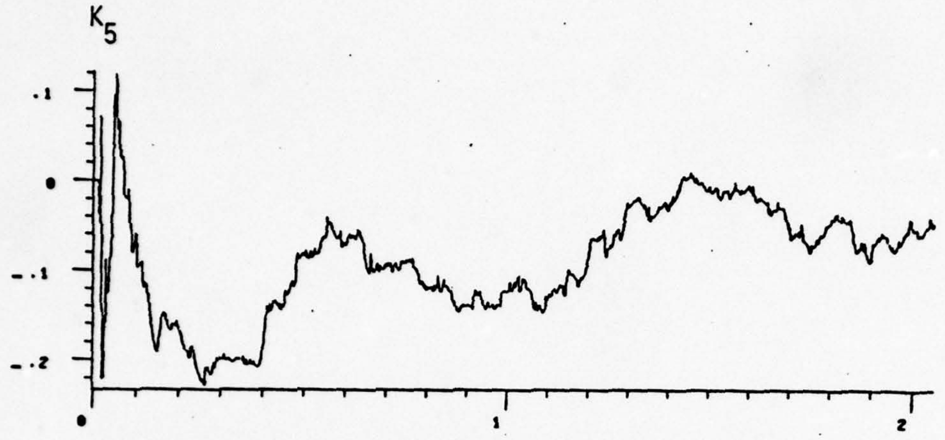


Fig. 6(d). Estimated K_5 of gaussian input only using time-varying weighting factor.

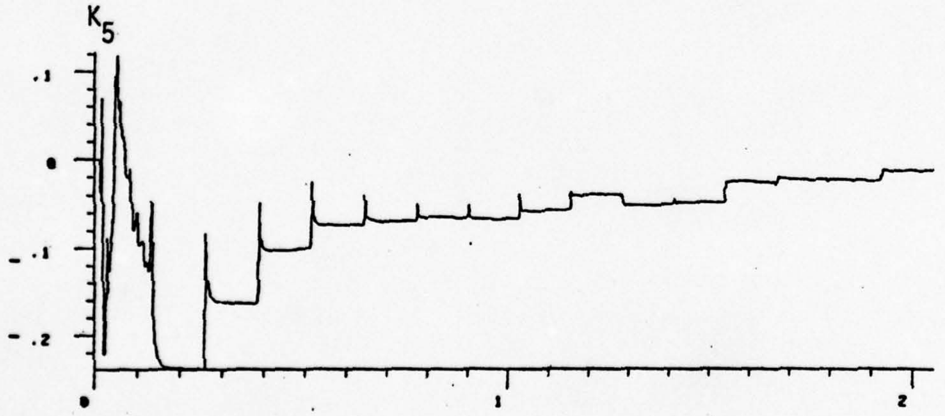


Fig. 6(e). Estimated K_6 of non-gaussian input using time-varying weighitng factor.

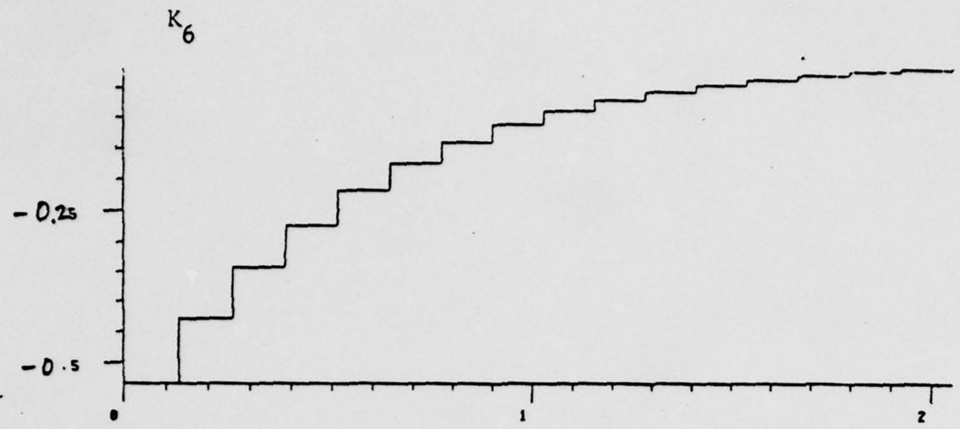


Fig.7(a) Sixth reflection coefficient, K_6, of underlying model.

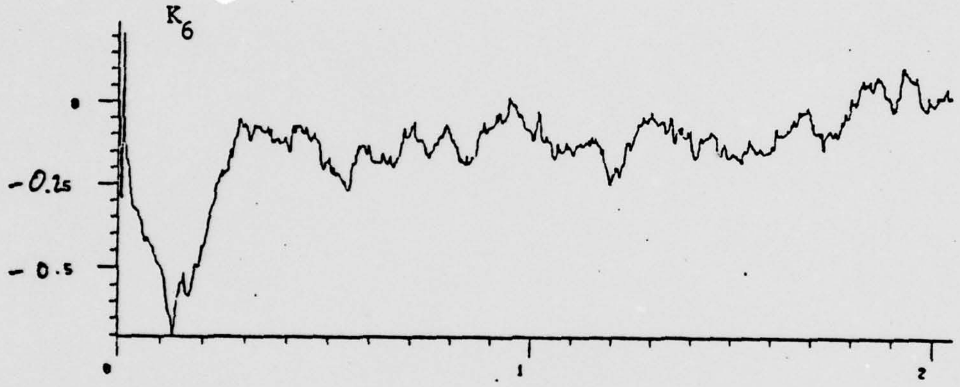
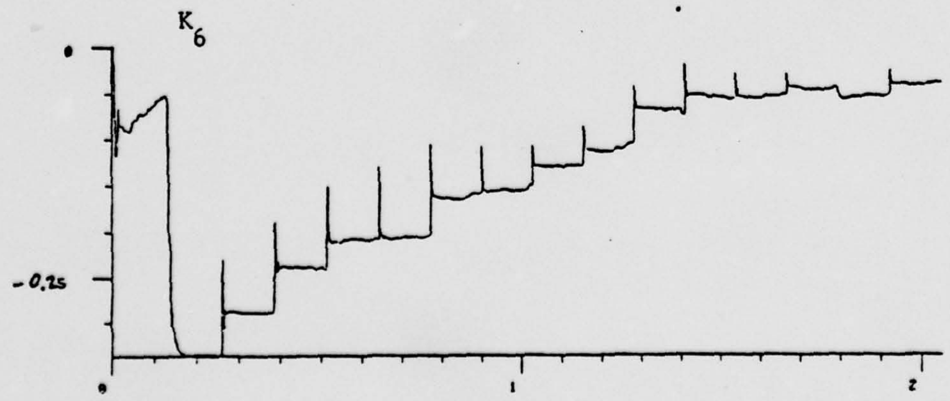


Fig.7(b) Estimated K_6 of unvoiced data (generated by white noise only).



Fig,7(c) Estimated K_6 of unvoiced data (generated by impulse train plus white noise).

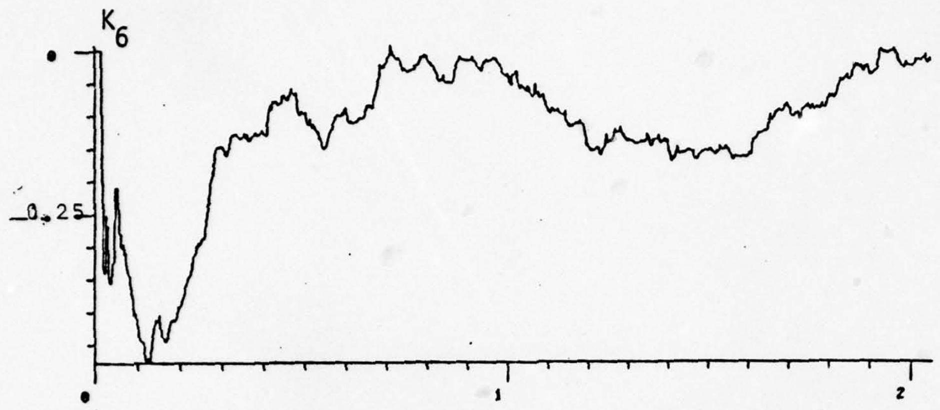


Fig. 7(d). Estimated K_6 of gaussian input only using time-varying weighting factor.

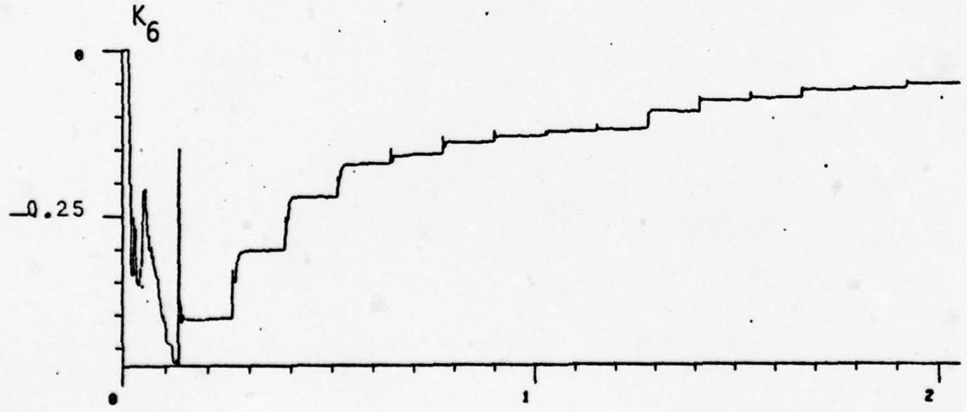


Fig. 7(e). Estimated K_6 of non-gaussian input, using time-varying weighting factor.

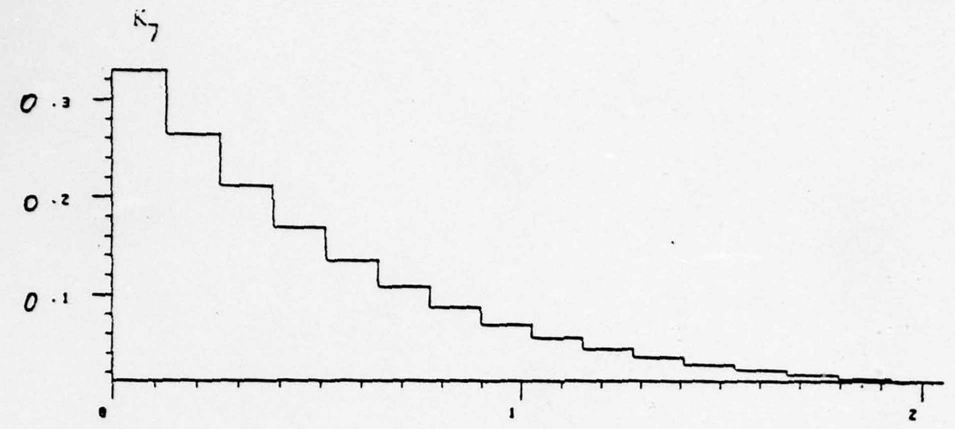


Fig.8(a) Seventh reflection coefficient, K_7, of underlying model.

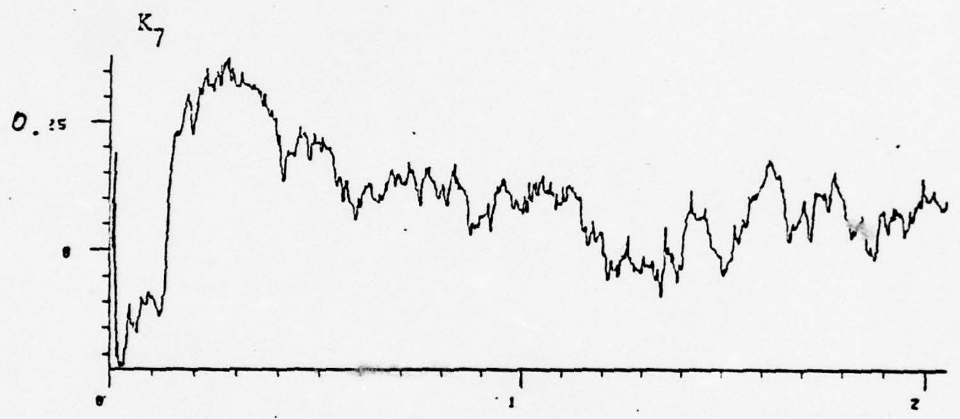


Fig.8(b) Estimated K_7 of unvoiced data (generated by white noise only).

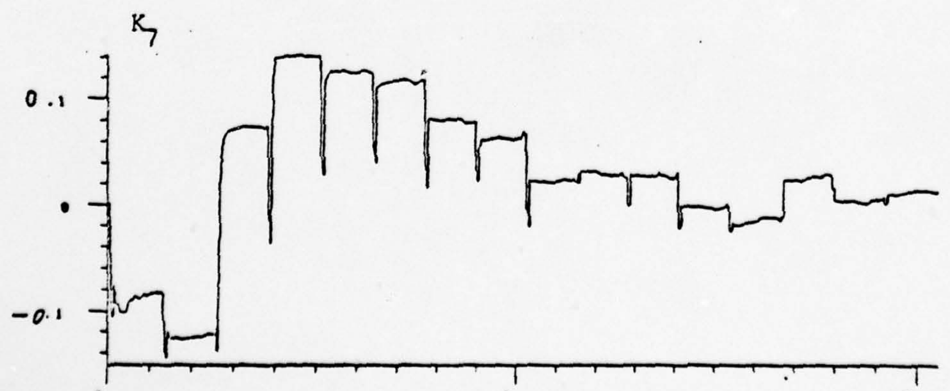


Figure 8(c) Estimated K_7 of voiced data (generated by impulse train plus white noise).

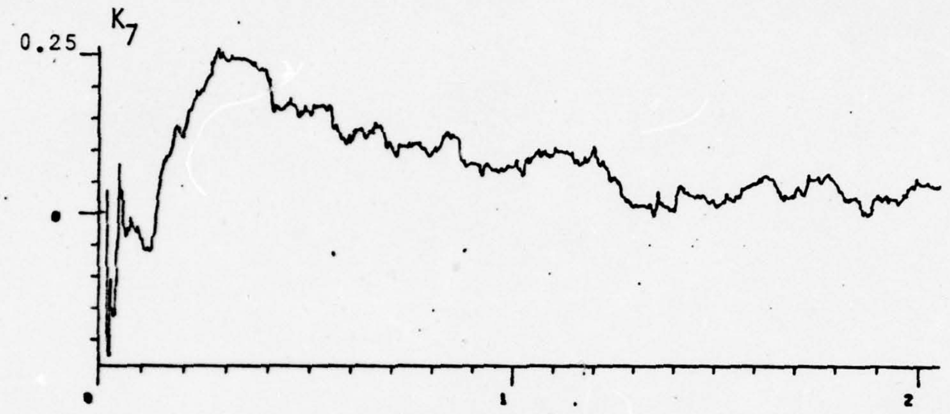


Fig. 8(d). Estimated K_7 of gaussian input only, using time-varying weighting factor.

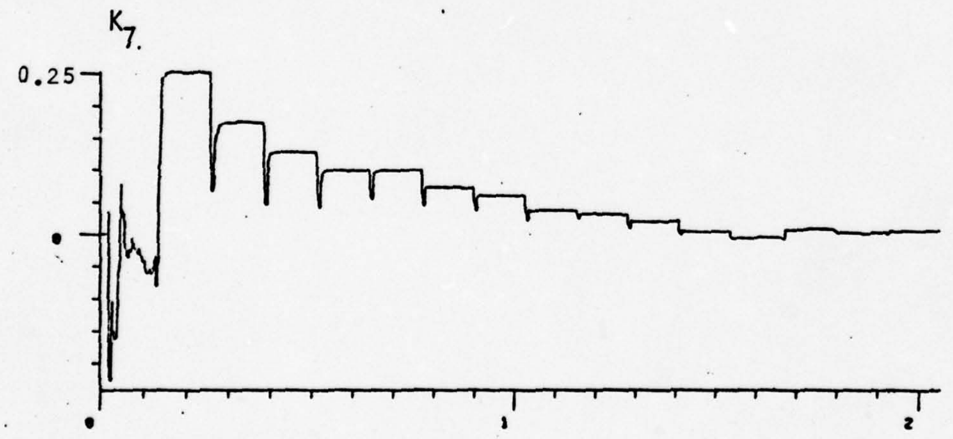


Fig. 8(e). Estimated K_7 of non-gaussian input, using time-varying weighting factor.

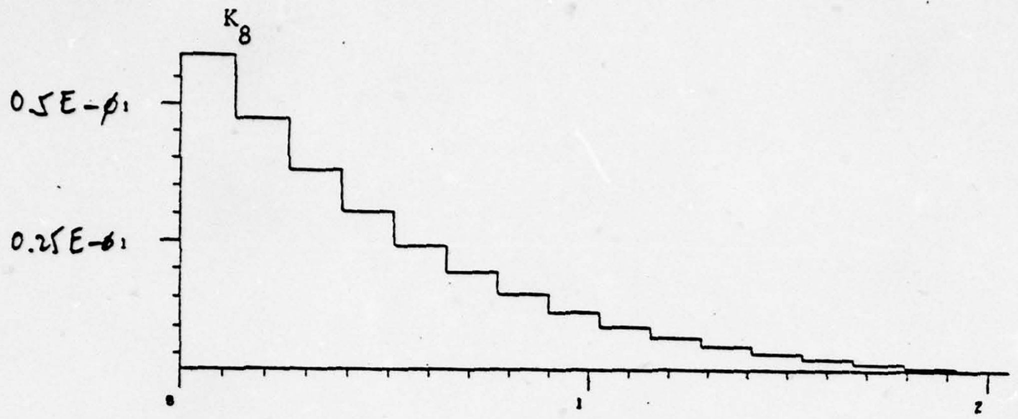


Fig.9(a) Eighth reflection coefficient, K_8, of underlying model.

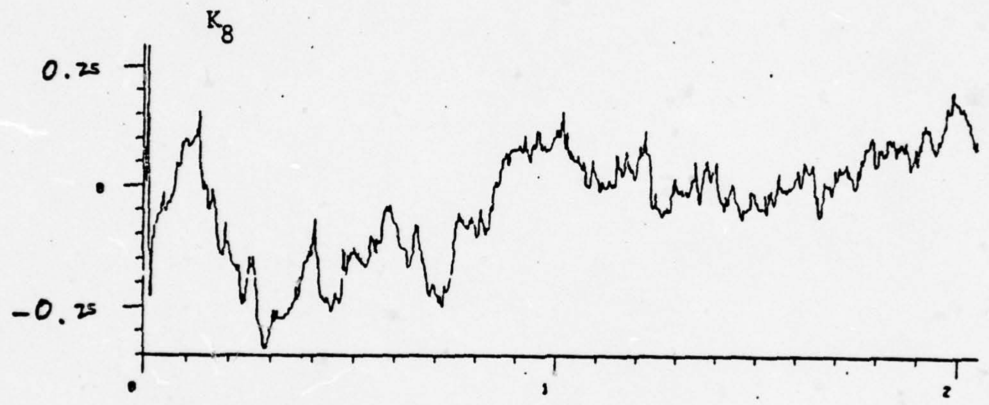


Fig.9(b) Estimated K_8 of unvoiced data (generated by white noise only).

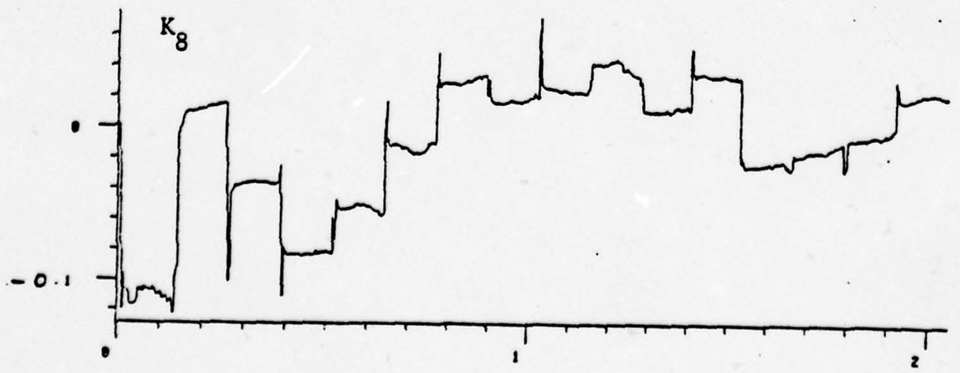


Fig9(c) Estimated K_8 of voiced data (generated by impulse train plus white noise).

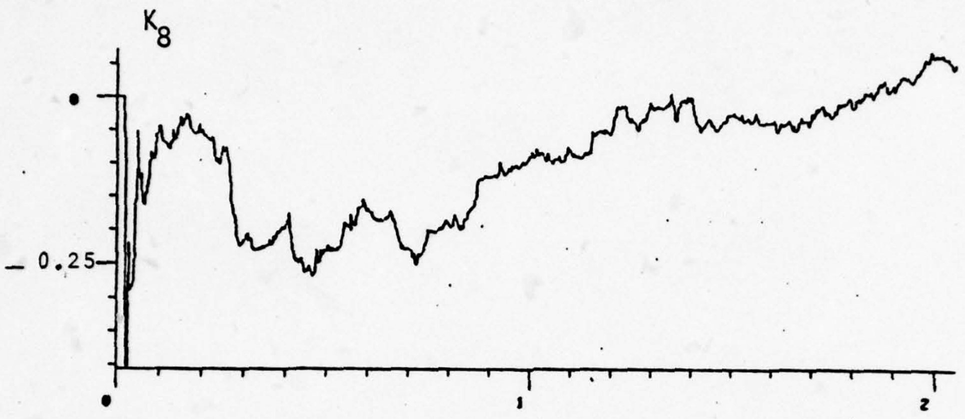


Figure 9(d). Estimated K_8 of gaussian input only, using time-varying weighting factor.

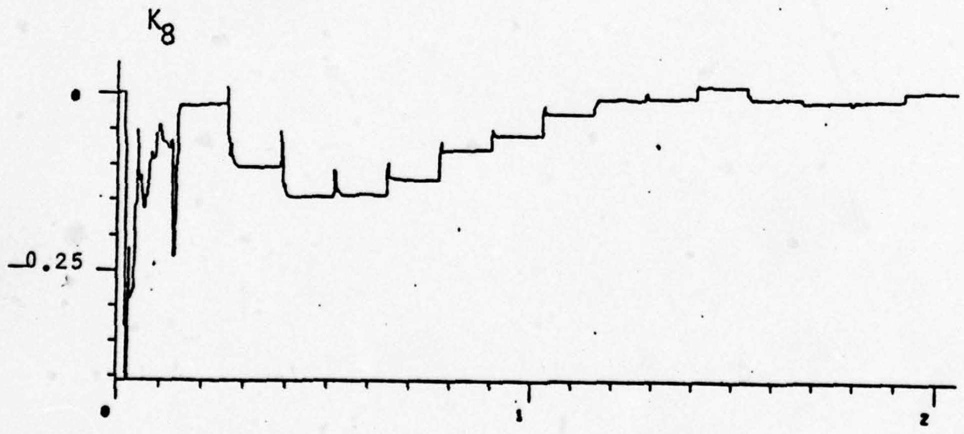


Figure 9(e). Estimated K_8 of non-gaussian input, using time-varying weighting factor.

5.2.2 Results on Real Speech Data

5.2.2(a) Local Behaviour of Reflection Coefficients

This test is to demonstrate the dynamic behaviour of the estimated reflection coefficient during a segment of voiced speech data. Two sets of speech data were used. The first set is taken from a series of high resolution, sampling rate of 20,000 samples per second, speech data. The data was the vowel /e/ in the word "the". This test will illustrate the local behaviour of reflection coefficients within a few pitch periods.The total number of samples in the segment is 960 and the time constant of the tracking rate was set at 300. The estimated reflection coefficient, the normalized prediction errors or innovations, and gamma, which is part of the likelihood ratio, were illustrated for the following order: 1st, 5th, 10th, and 15th.

Observations

The 15th order PARCOR coefficient K[15] converges to almost zero for voiced sounds. Reflection coefficients clearly show piecewise constant behavior.

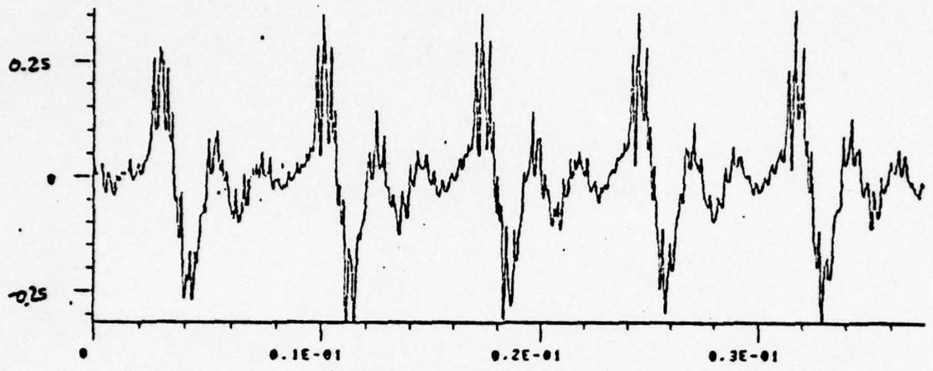


Fig.11 Speech waveform of /ē/ in "the" - 960 samples.
Sampling rate = 20,000 Hz.

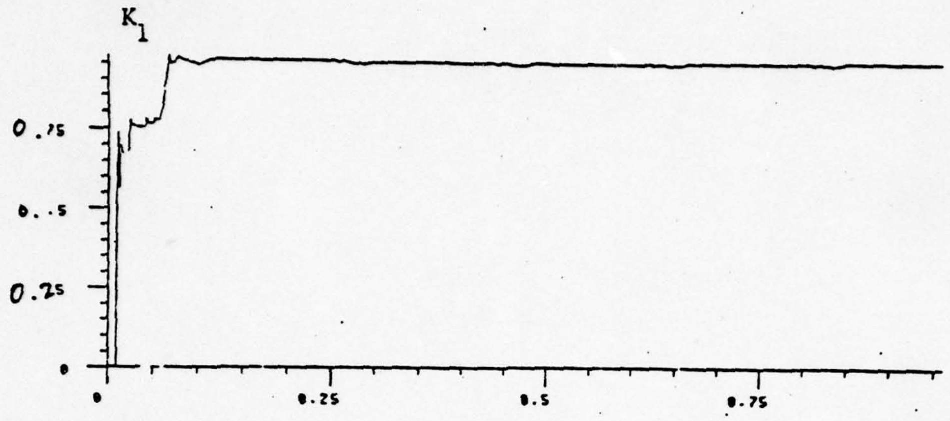


Fig. 12(a) 1st order reflection coefficient.

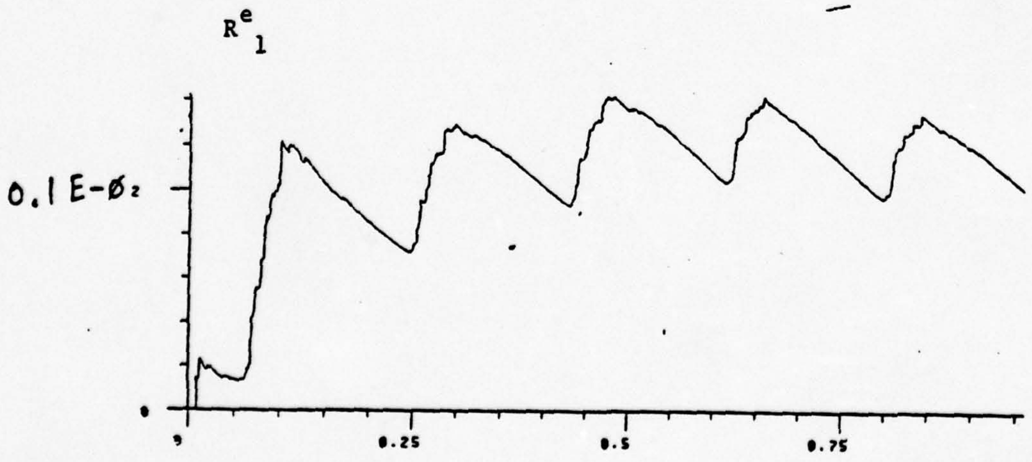


Fig. 12(b) 1st order prediction error covariance.

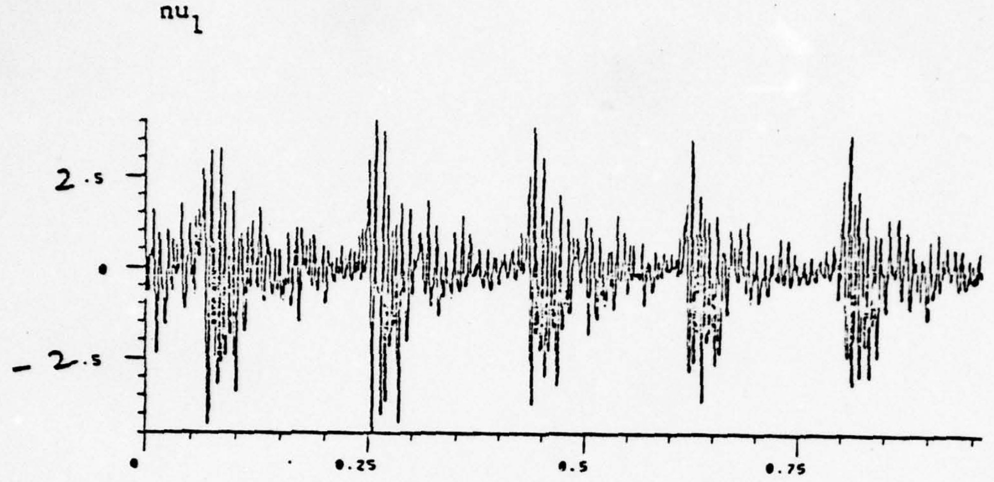


Fig.12(c) 1si order innovation.

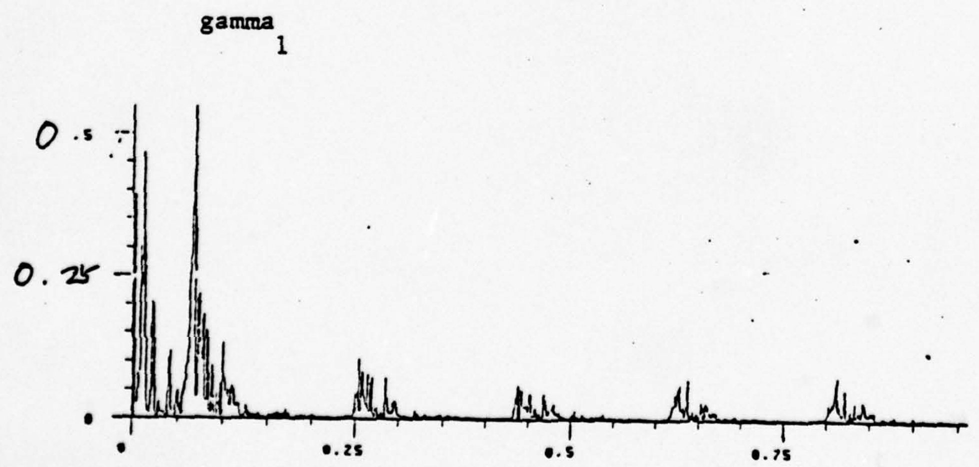


Fig.12(d)1st order gamma.

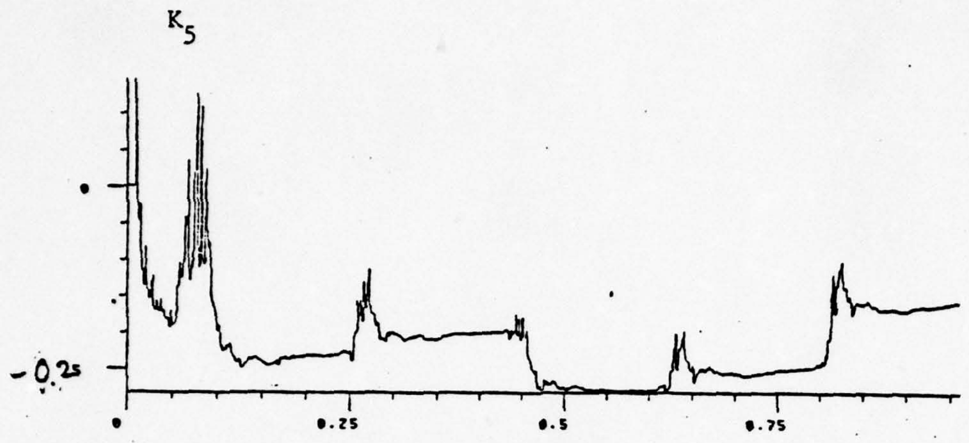


Fig.13(a)5th order reflection coefficient.

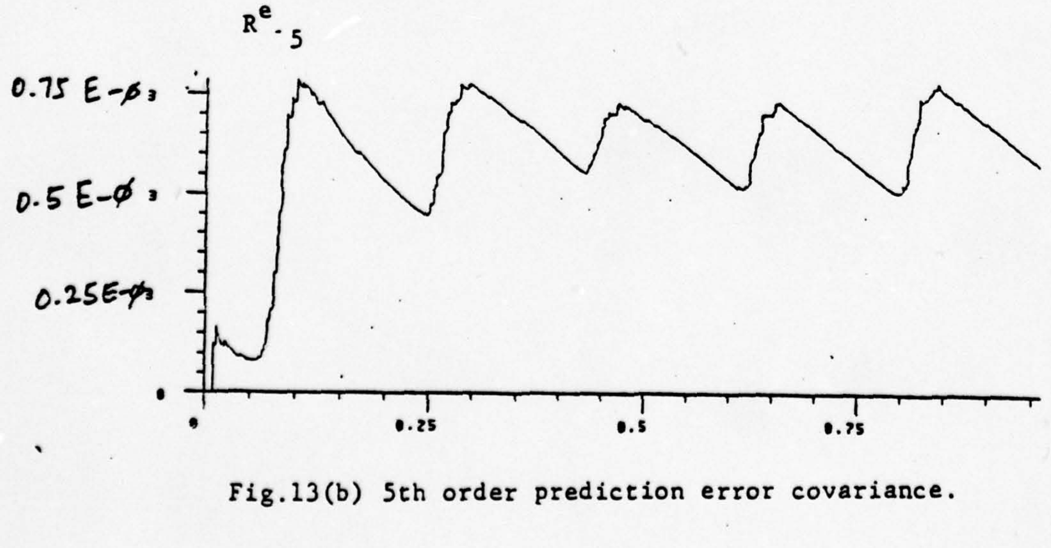


Fig.13(b) 5th order prediction error covariance.

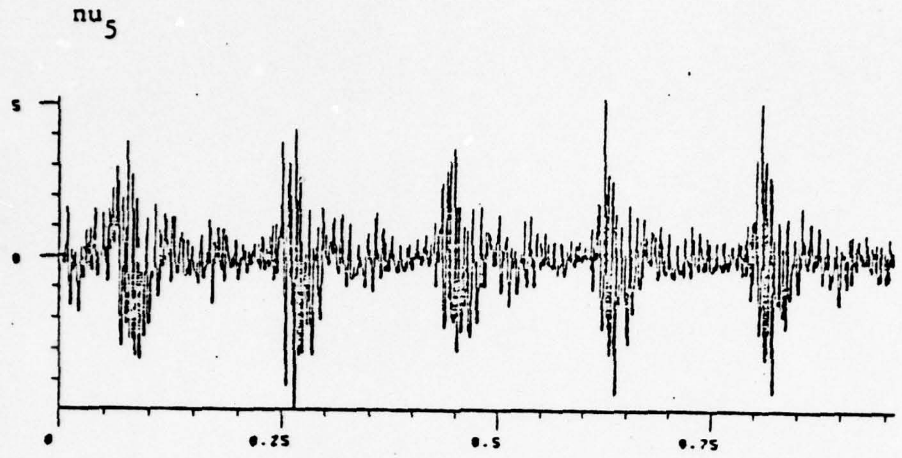


Fig.13(c).5th order innovation.

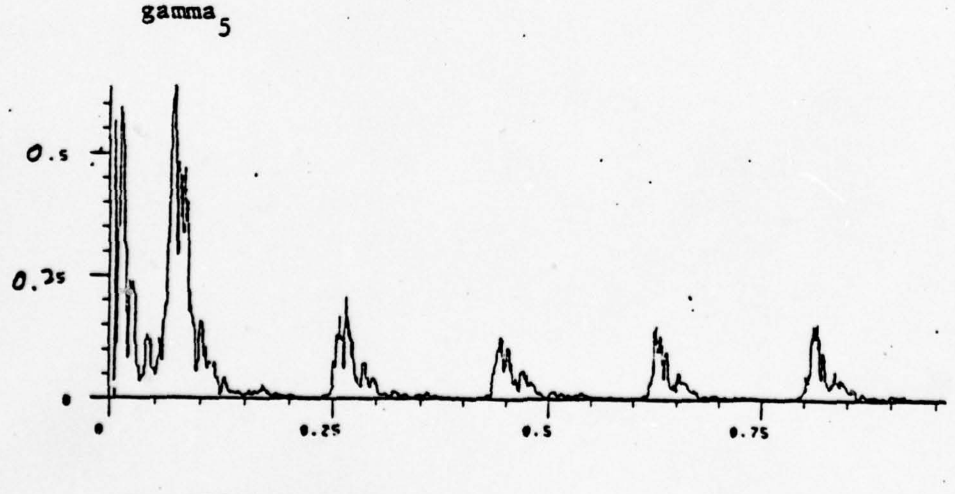


Fig.13(d) 5th order gamma.

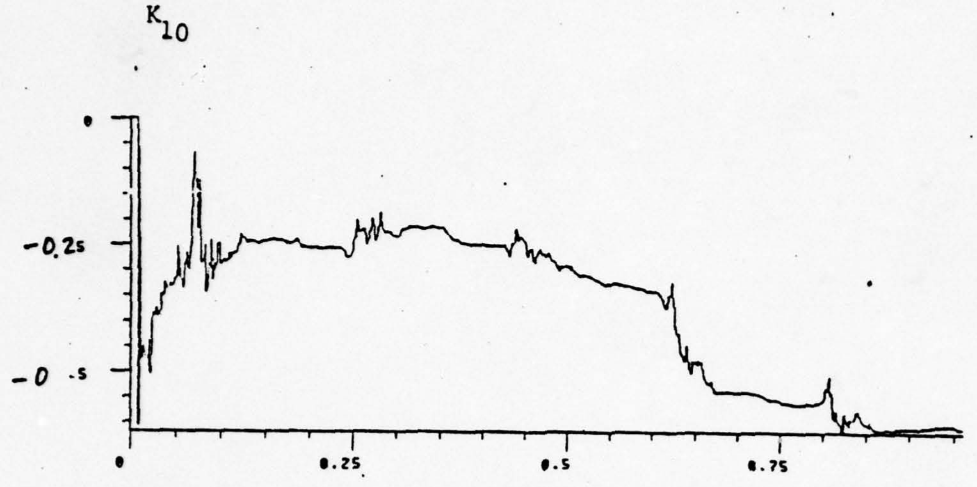


Fig.14(a) 10th order reflection coefficient.

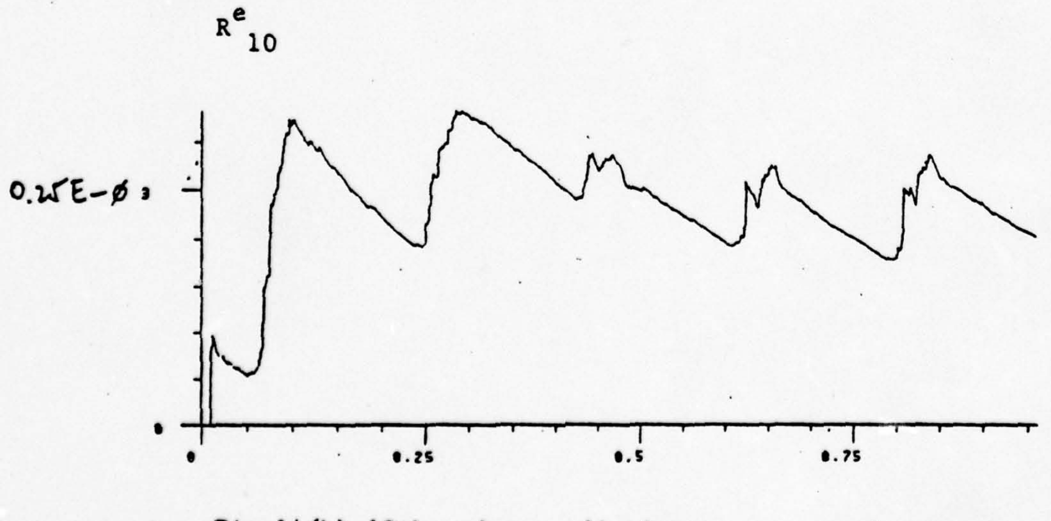


Fig.14(b) 10th order prediction error covariance.

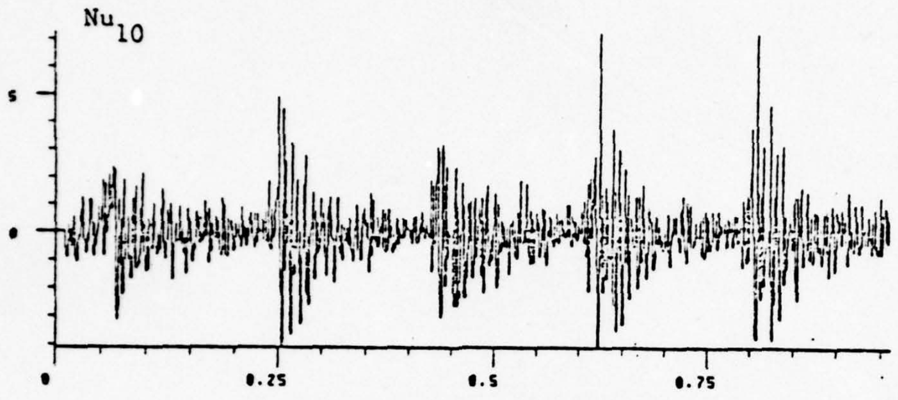


Fig.14(c) 10th order innovation.

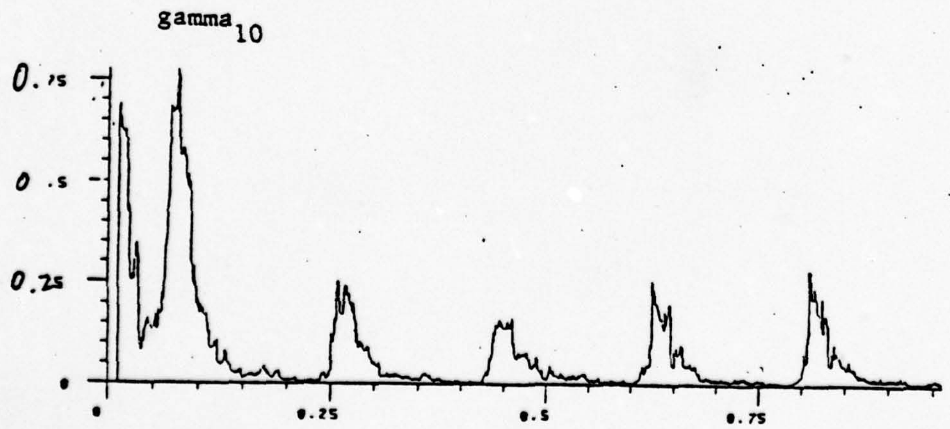


Fig.14(d) Gamma_{10}.

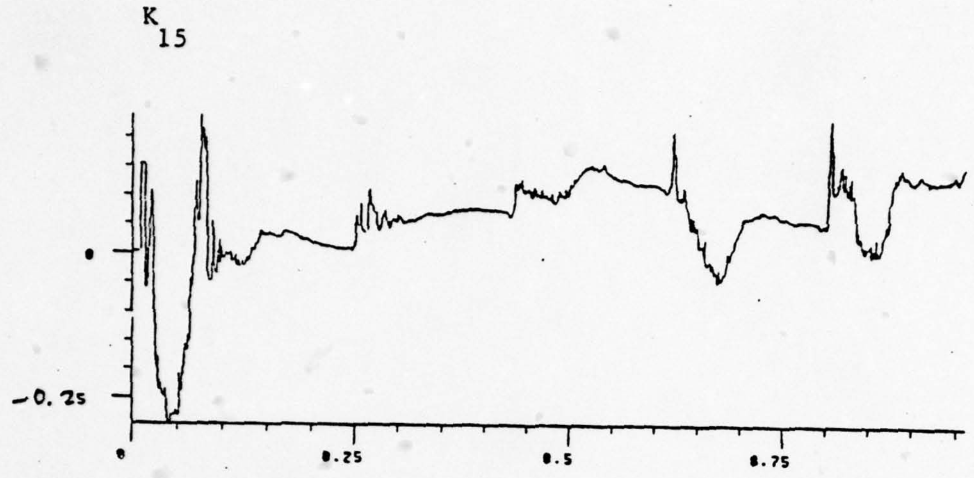


Figure15(a) 15th order reflection coefficient.

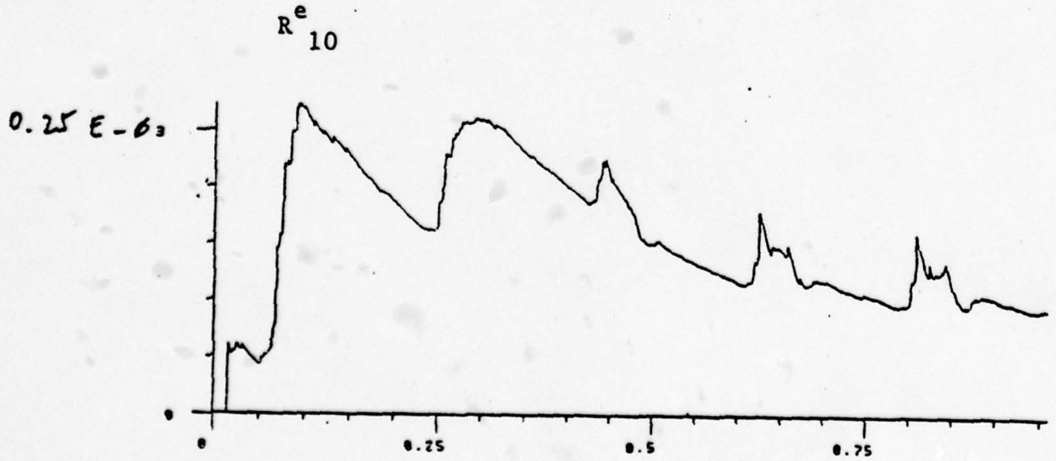


Fig.15(b) 15th order prediction error covariance.

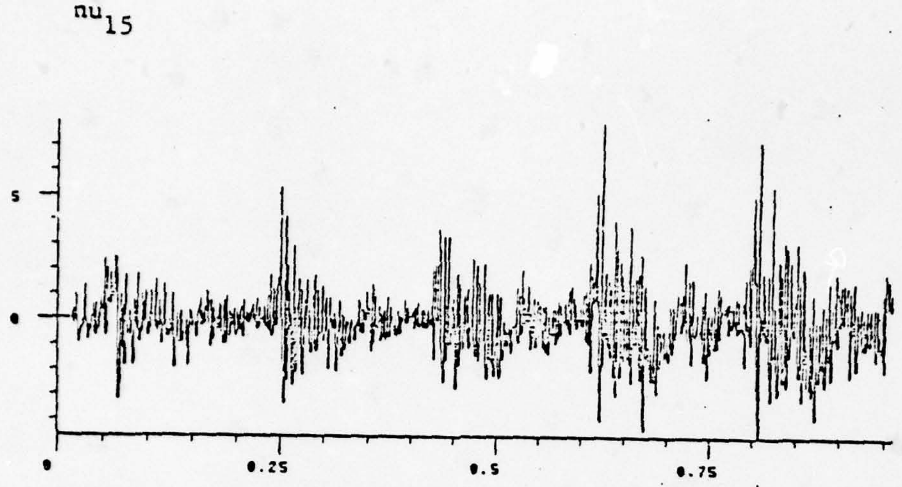


Figure15(c) 15th order innovation.

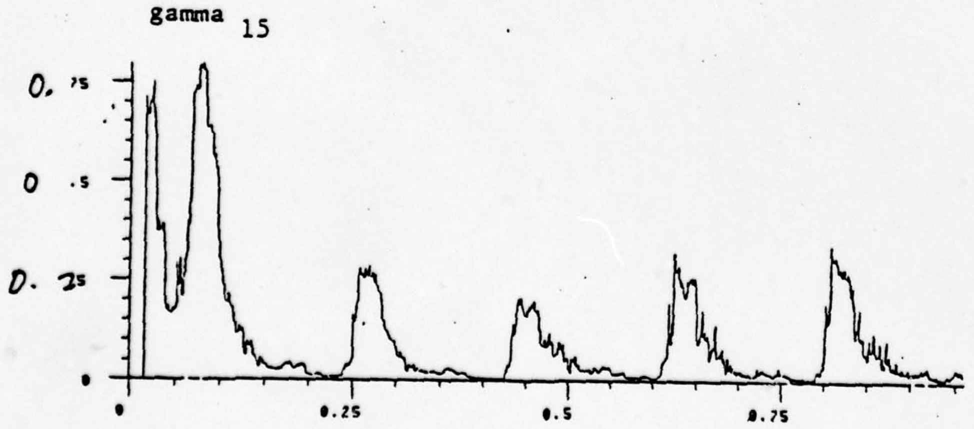


Figure 15(d) Gamma$_{15}$.

5.2.2(b) Global Behaviour of Reflection Coefficients

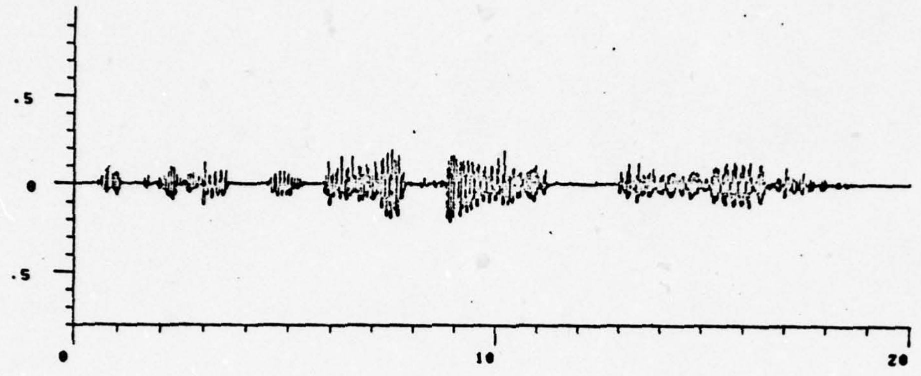
In this set of simulations, speech data sampled at 8000 samples per second was used. The sentence is "the pipe began to rust while new", female voice. In this set of simulations, the algorithm was set to track starting at the first non-zero input and was stopped at the last input sample. The various illustrations described below are just exerpts from the entire run in order to demonstrate some of the more interesting features.The tracking time constant was set at 160 samples, i.e. equivalent to two frame widths.

Formant Transition

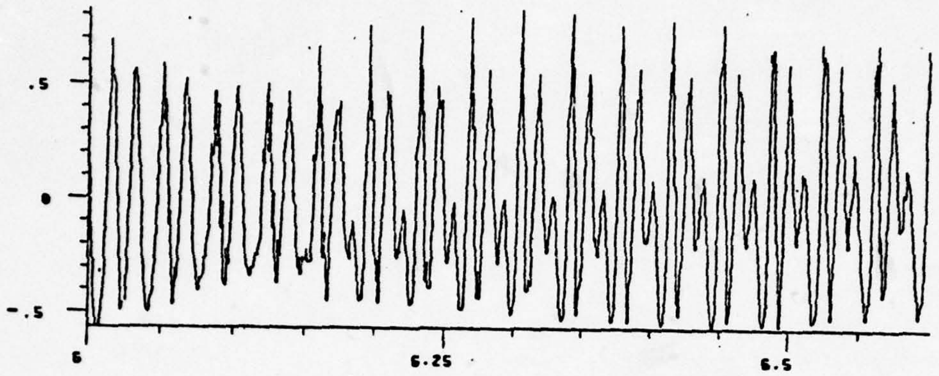
The first set of illustrations show the behaviors of the reflection coefficients up to to 13th order during a formant transition which occurs at about sample 6150. Observe the variations of the reflection coefficients during the change.

Phoneme String

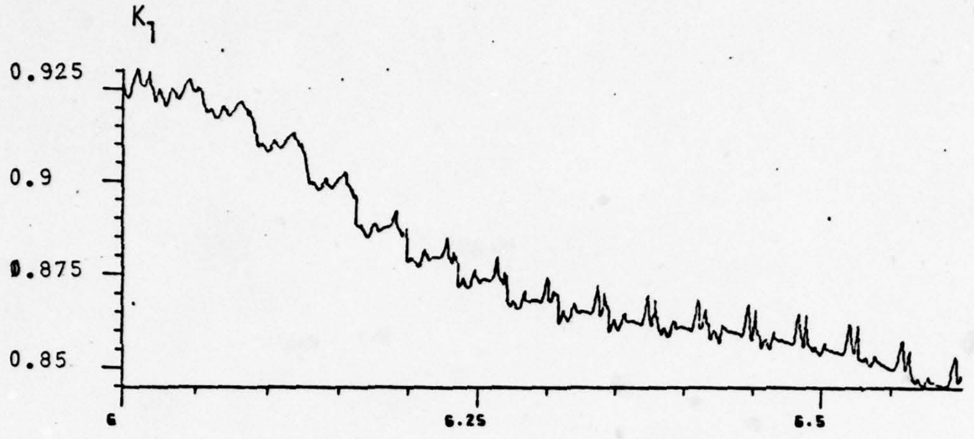
In this set of illustrations, reflection coefficients of the entire word "pipe" are displayed, showing the changes in reflection coefficients during the entire passage.



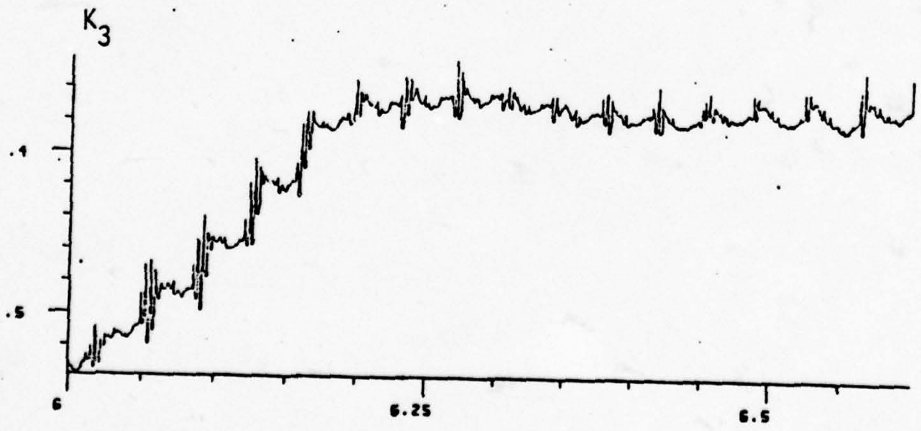
Original sentence: "the pipe began to rust while new",
female speaker, sampling rate: 8,000Hz.



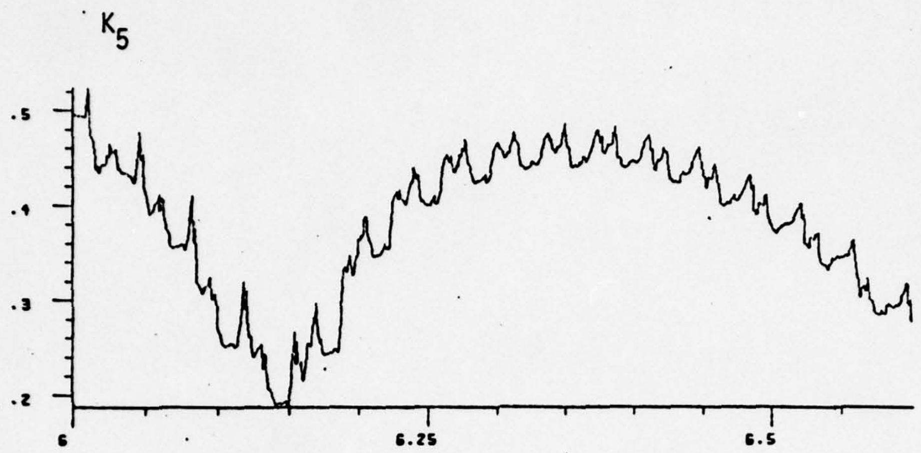
Samples #6000 to #6700.



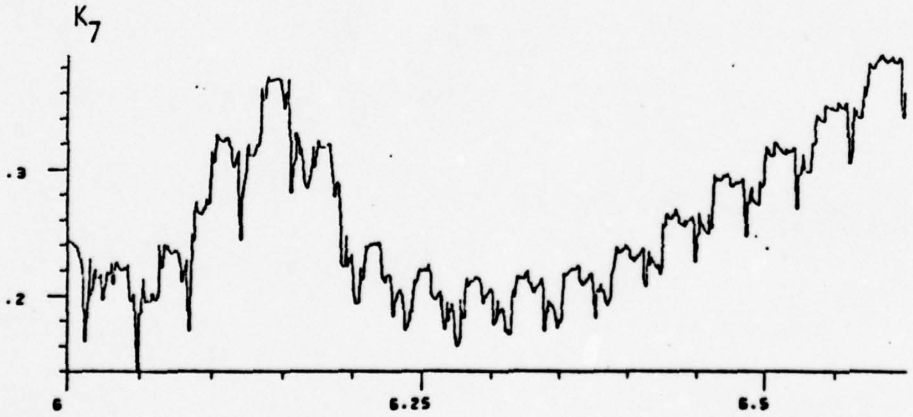
K_1 , first reflection coefficient.



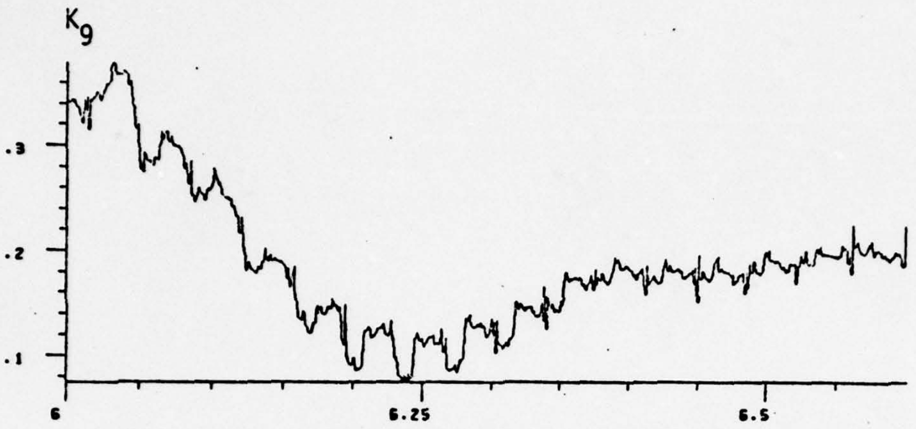
K_3 , third reflection coefficient.



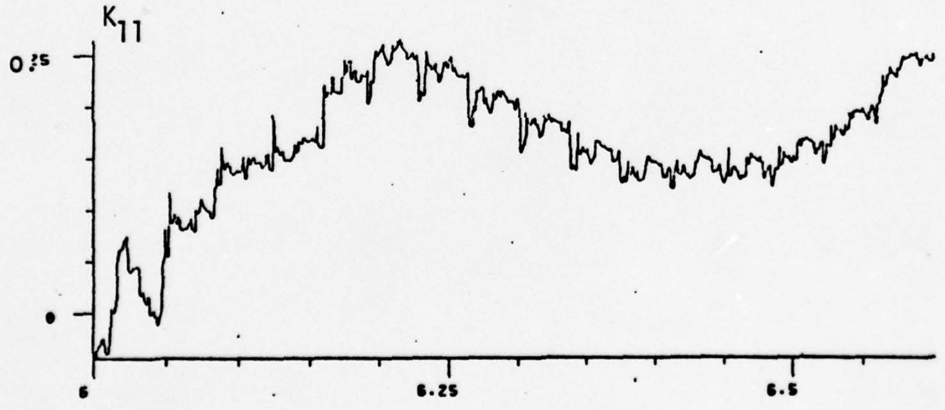
K_5 , fifth reflection coefficient.



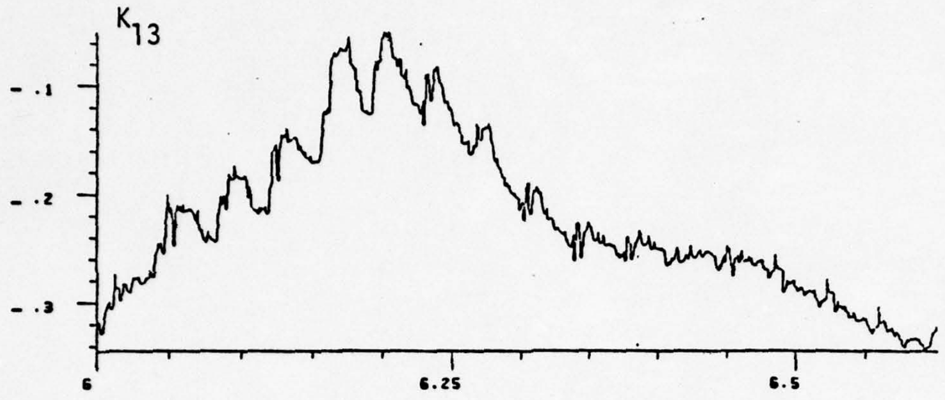
K_7 , seventh reflection coefficient.



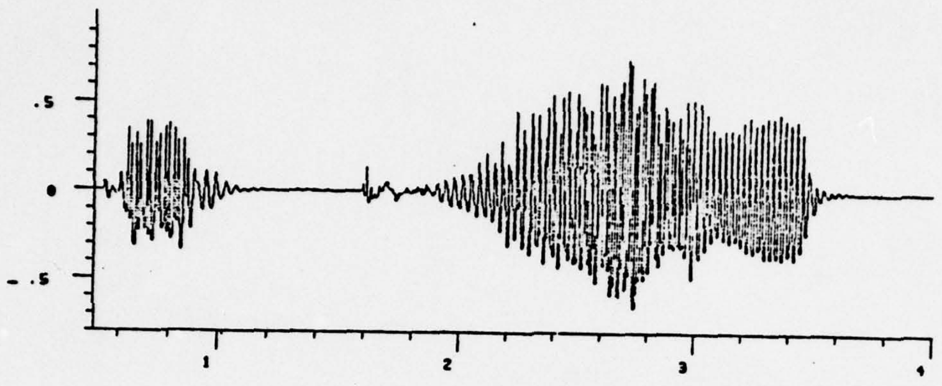
K_9 , ninth reflection coefficient.



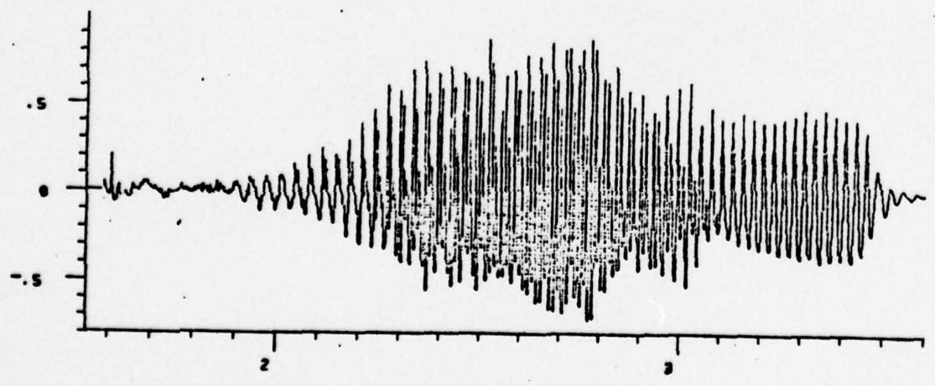
K_{11} , eleventh reflection coefficient.



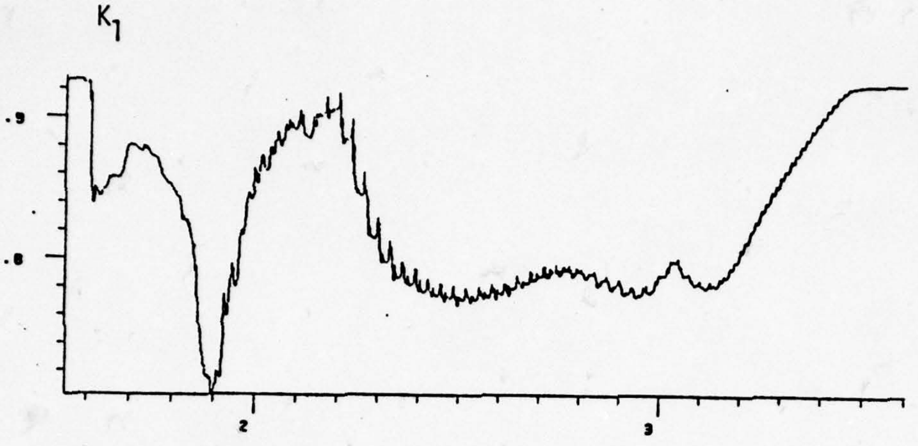
K_{13}, thirteenth reflection coefficient.

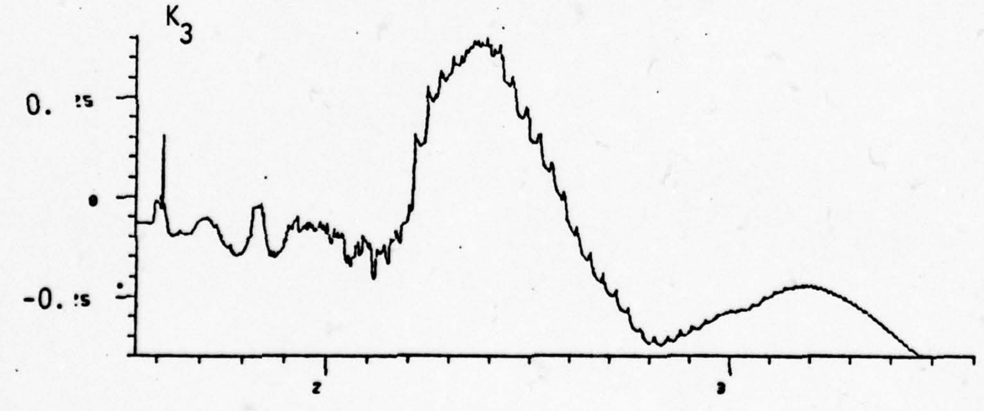


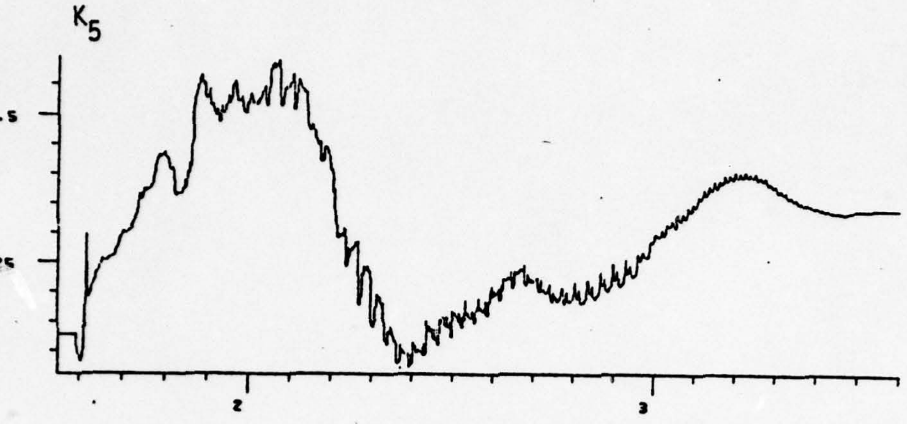
Samples #550 to #4000: "the pipe".

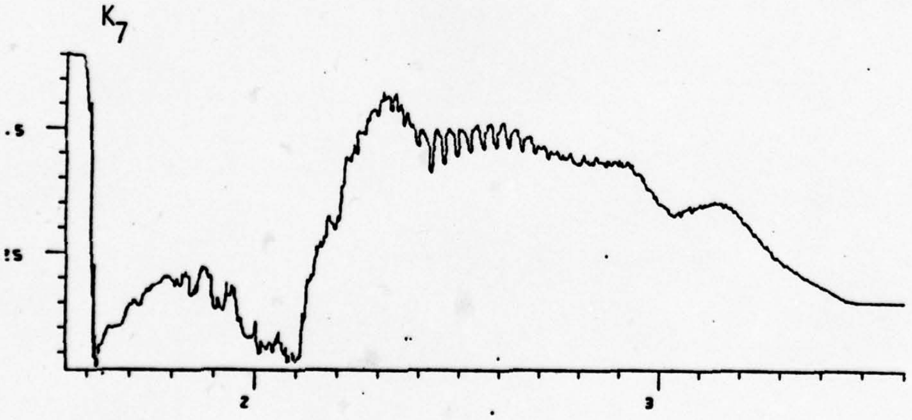


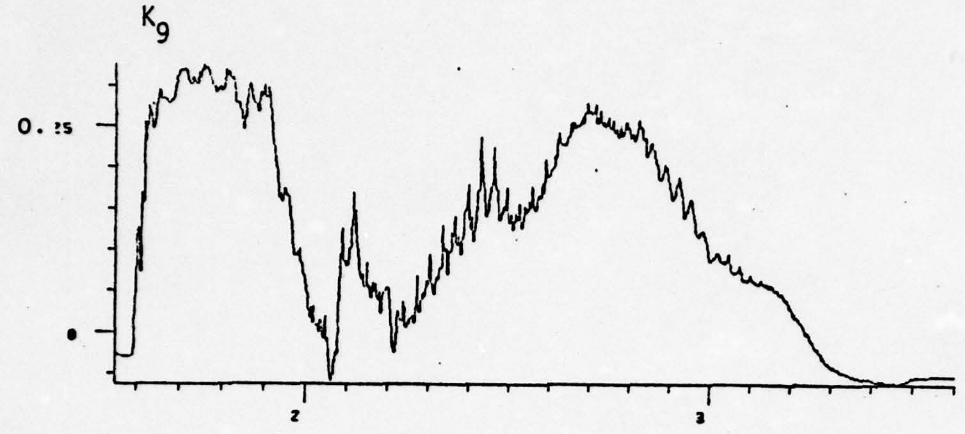
"pipe"


K1
.9
.8
2
3

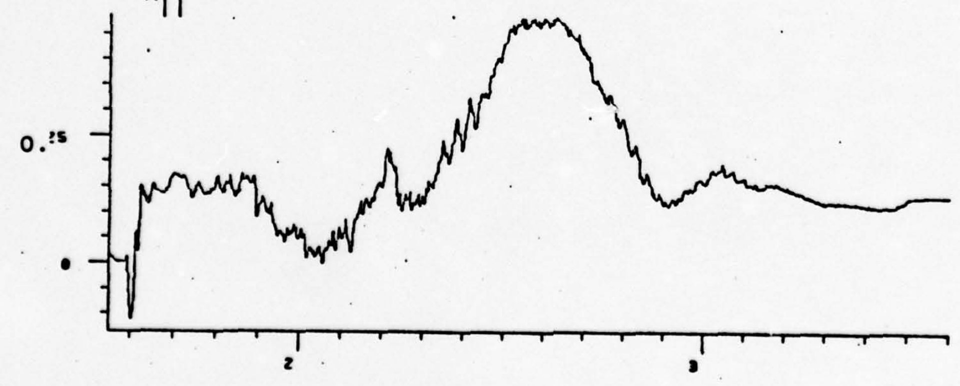

K3
0. :5
0
-0. :5
2
3


K5
.5
.25
2
3

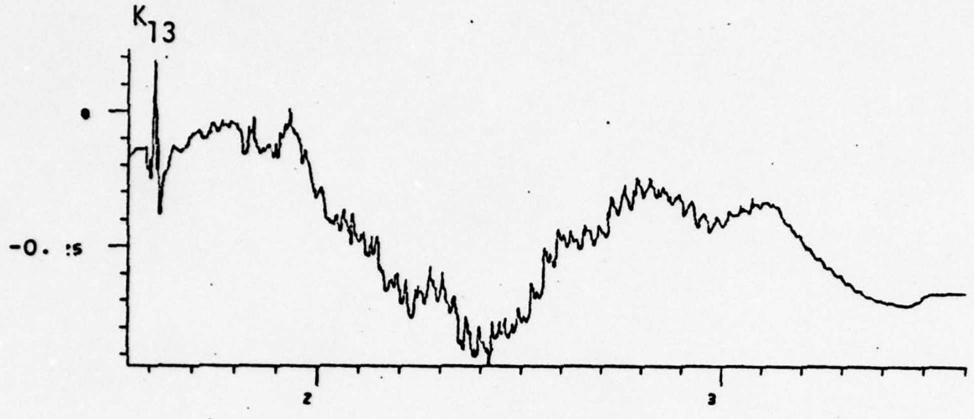

K7
.5
.25
2
3



K_9
0.25
0
2
3



K_{11}
0.25
0
2
3

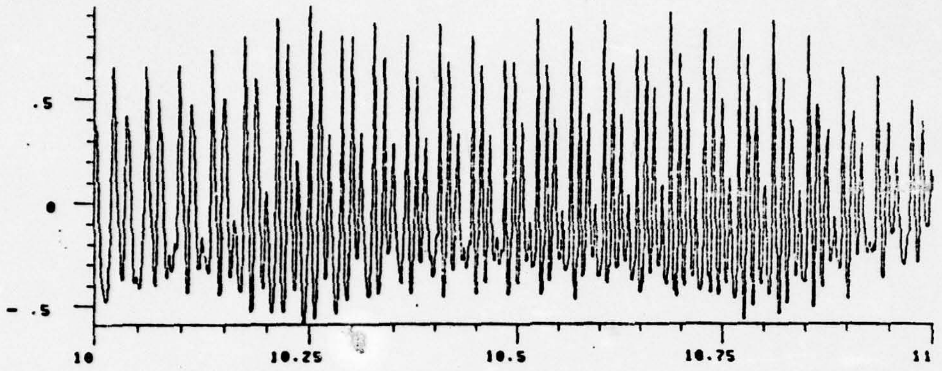


K_{13}
0
-0. :s
2
3

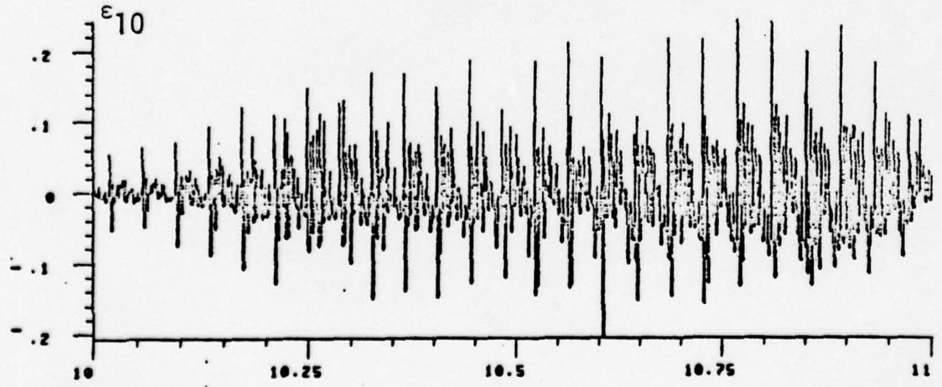
5.2.2(c) Pitch Detection Scheme

The first set of illustrations decribes a pitch detection scheme that is based on the likelihood ratio parameters, namely γ (gamma parameters).

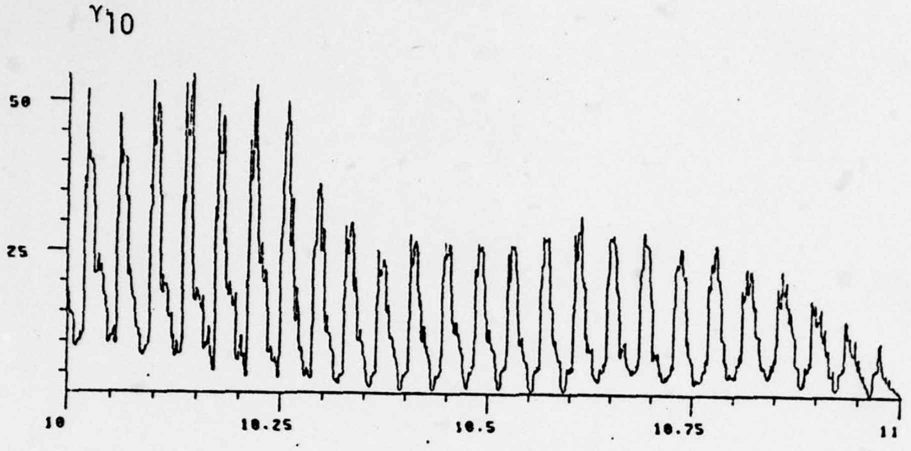
The second set shows the results of using a different implementation of the log likelihood ratio in detecting the pitch of a different segment of voiced speech, namely a) γ_p only, b) γ_p plus ln-$R^{\epsilon}{}_p$ part and c) the true likelihood ratio, indicating the progressive contributions of the *ln* part. (c.f. likelihood formulas above.)



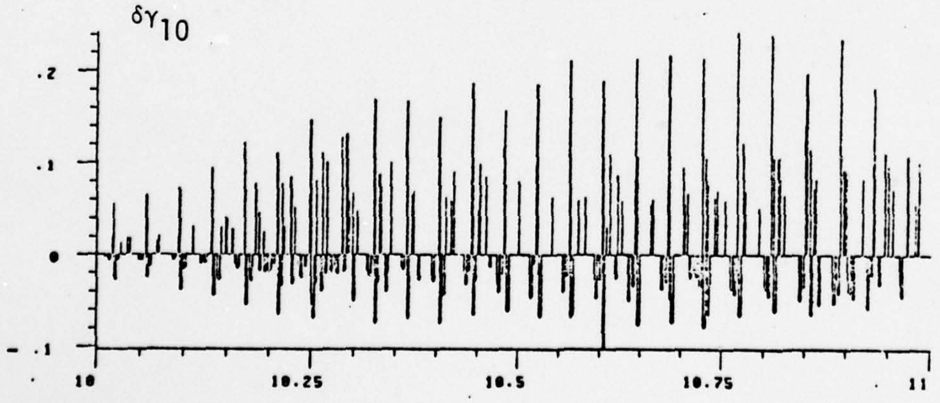
Original Speech Waveform.



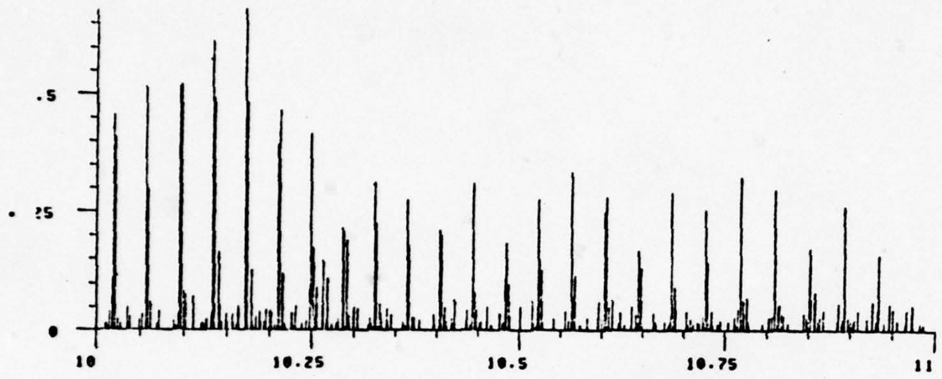
10th order innovations.



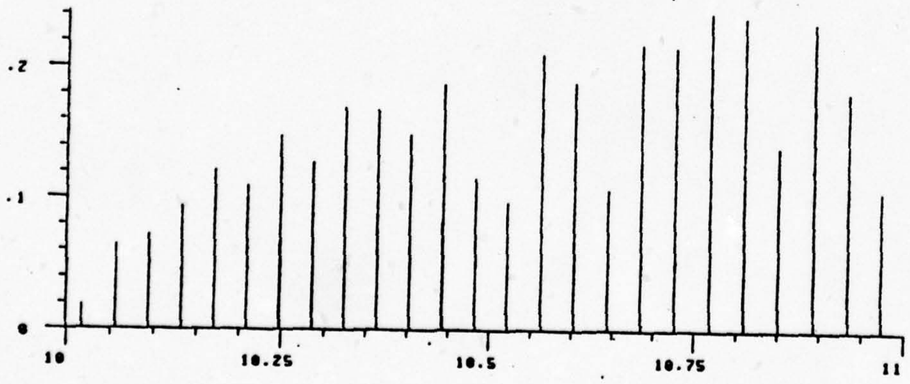
Gamma$_{10}$. (likelihood function)



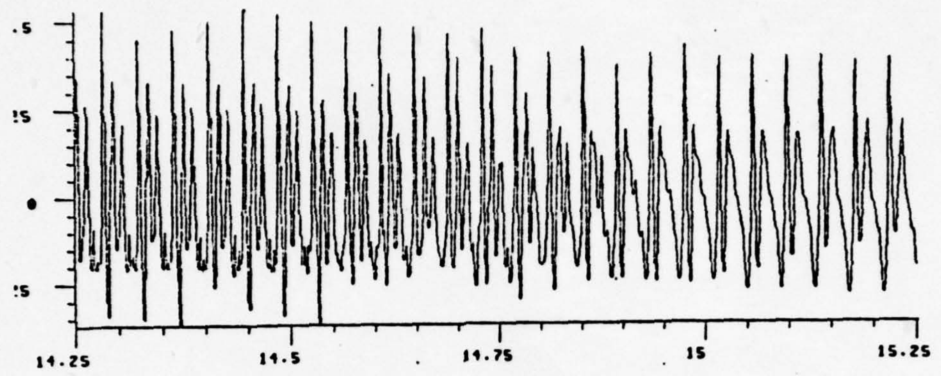
Delta gamma$_{10}$ (Log-likelihood ratio)



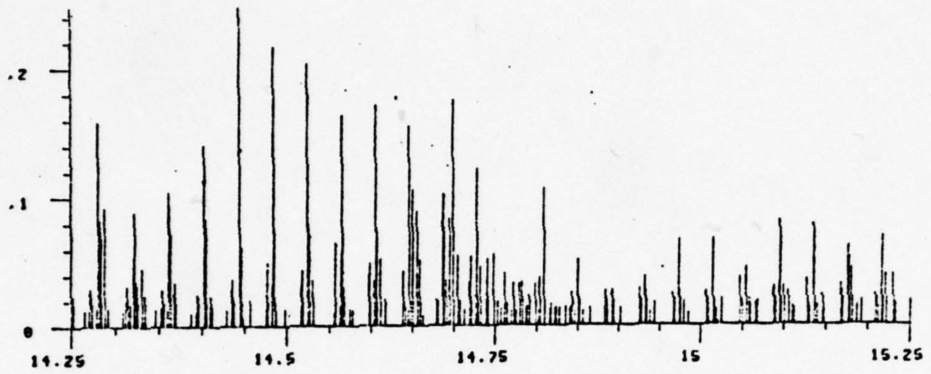
$| \delta \gamma_{10} \times \varepsilon_{10} |$ (Innovations masked by log-likelihood ratio)



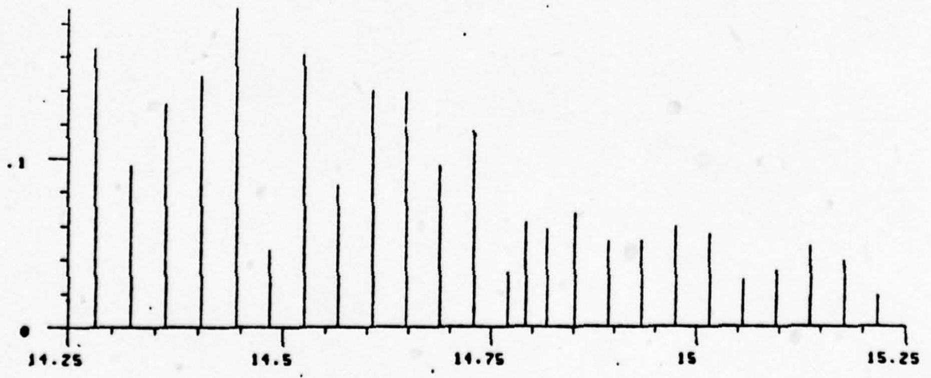
Detected Pitch pulses.



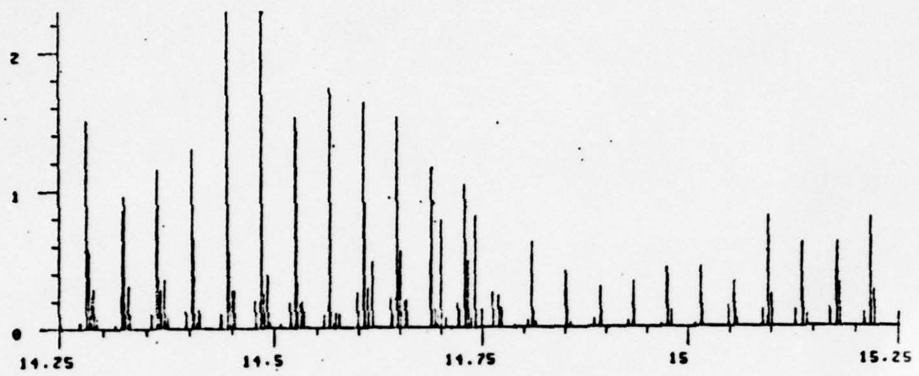
Original Speech Waveform.



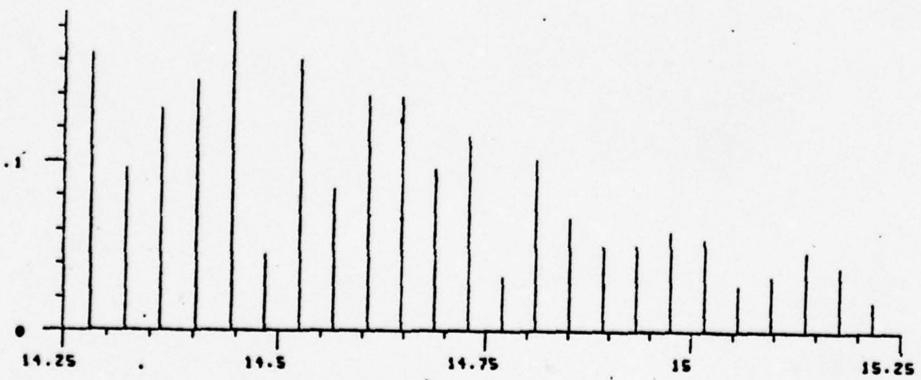
$\delta\ \gamma_{10}$



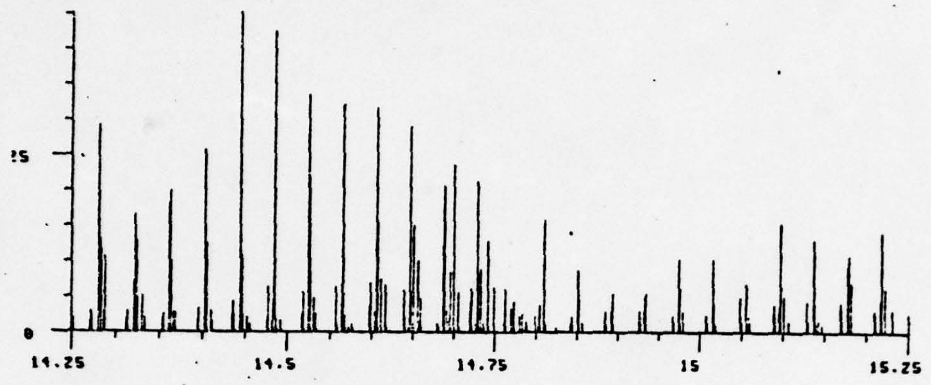
Detected pitch pulses using $\delta\ \gamma_{10}$



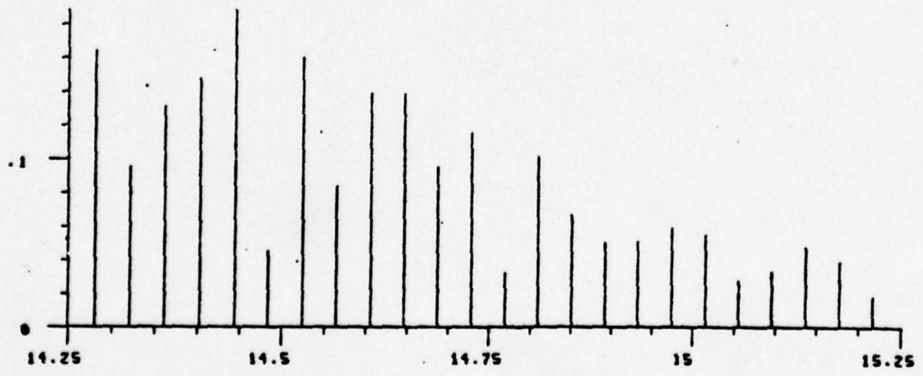
Weighted mean of $\delta\gamma_{10}$ and $\ln R^{\varepsilon}_{10}$.



Detected pitch pulses using weight mean.



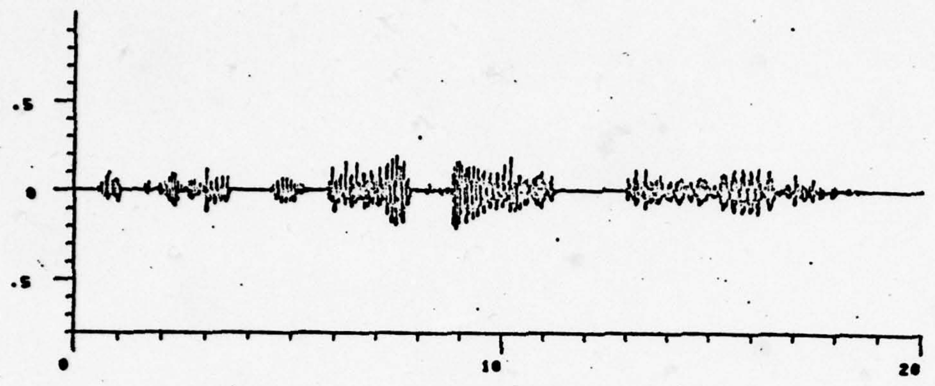
True Likelihood Function.



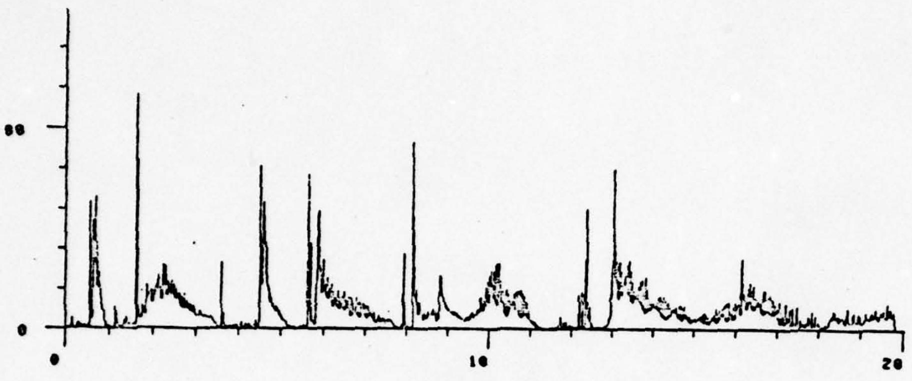
Detected pitch pulses using true likelihood function.

5.2.2(d) Global Behaviour of Log likelihood function.

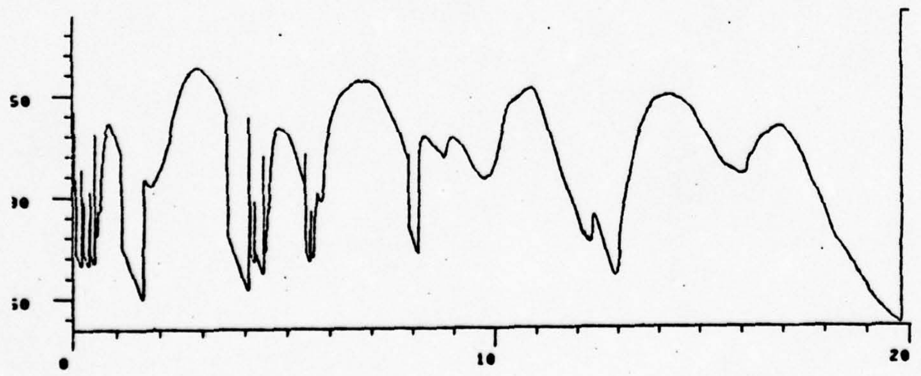
This illustration shows the global picture of the log likelihood ratio over the whole sentence, and its potential for segmenting speech.



Original sentence: "the pipe began to rust while new", female speaker, sampling rate: 8,000Hz.



γ_{10} over the entire sentence.



Likelihood Function over the entire sentence.

6. Conclusions

The results of the initial testing of our new speech modeling algorithms have been very promising. These algorithms seem to be very well suited for speech modeling as indicated by the "constant parameter" behavior during the pitch period, good tracking capability and a novel likelihood ratio based pitch detection technique associated with the ladder form. Their performance is very promising both in terms of achievable speech quality and potential for data rate reduction.

Due to the short term and limited scope of this research effort, our results can only be considered preliminary. Further research and testing is needed, particularly in the area of pole-zero (ARMA) modeling and parameter quantization (coding/decoding) methods. The pitch detection problem (not part of the scope of this project) also requires further study and refinements, especially in the context of mixed processes.

It is expected that improved system performance can result by pursuing our approach further and by performing more extensive testing to "tune" the transmission system. Real-time special-purpose processor hardware would be needed to effectively test various algorithms that have shown promise with a larger data base. The current development and simulations are clearly processor limited if done on a general purpose time-sharing system.

On a more general level, a number of important research problems are incompletely resolved at this point and need further study. They are not just of interest to speech modeling since they include basic pertaining to representations of mixed processes such as Gaussian and poisson-type jump processes (potentially useful for modeling voiced speech). Good performance measures for speech transmission systems, such as acceptable distortion measures, and the establishment of the basic limits of speech compression, i.e. the distortion rate function of speech processes, are needed for further investigation in speech modeling.

References

[BJ] Box, P and Jenkins, L.,*Time Series Analysis and Forcasting*, San Francisco: Holden Day, 1973.

[Buzo] Buzo, L., Ph.D. Dissertation, Stanford University, August 1978.

[FMKL] Friendlander, B., M. Morf, T. Kailath and L. Ljung, "New Inversion Formulas for Matrices Classified in Terms of Their Distance from Toeplitz Matrices,"submitted to SIAM J. A. Math.

[IS] Itakura,F., and S. Saito,"Digital Filtering Techniques for Speech Analysis and Synthesis," Conf. Rec., 7th Int. Congr. Acoust.,Budapest,1971,Paper 25 C 1.

[Lev] Levinson, N.,"The Wiener RMS (root mean square) Error Criterion in Filter Design and Prediction," J. Math. Phys.,vol.25,1947, pp.261-278.

[Lju] Ljung, L., "Analysis of Recursive Stochastic Algorithms," IEEE Trans. Automatic Control, Vol. AC-22, Aug. 1977, pp.551-576.

[Makh] Makhoul, J. "Linear Prediction: A Tutorial Review," Proc. IEEE, vol. 63, 1975, pp. 561-580.

[MDKV] Morf, M., B. Dickenson, T. Kailath and A. Vieira, "Efficient Solution of Covariance Equations for Linear Prediction," submitted to IEEE-ASSP 1976.

[MG] Markel, J.D., and A.H. Gray, Jr.,*Linear Prediction of Speech*, Springer-Verlag, Berlin: 1976.

[MKD] Morf, M., T. Kailath and B. Dicksinson, "General Speech Models and Linear Estimation Theory," in *Speech Recognition*, pp.157-182, New York: Academic Press, 1975.

[MLNV] Morf, M., D. T. Lee, J. R. Nickolls and A. Vieira, "A Classification of Algorithms for ARMA Models and Ladder Realizations," Conf. Rec., 1977 IEEE Int. Conference on Acoustics, Speech & Signal Processing, Hartford, 1977, pp. 13 - 19.

[Mo74] Morf, M., "Fast Algorithms for Multivariable Systems," Ph.D. dissertation, Stanford University, Stanford, California 1974.

[Mo77] Morf, M., "Ladder Forms in Estimation and System Identification," 11th Annual Asilomar Conference on Circuits, Systems, and Computers, Monterey, Ca., Nov. 7 - 9, 1977.

[MVLK] Morf. M., A. Vieira, D. T. Lee and T. Kailath, "Recursive Multi-channel Maximum Entropy Method," Proc. 1977 Joint Automatic Control Conf.,San Francisco,1977, pp.113-117.

[MVL] Morf. M., A. Vieira and D. T. Lee, "Ladder Forms for Identification and Speech Processing," Proc. 1977 IEEE Conference on Decision and Control, New Orleans, La.,Dec. 7 - 9, 1977.

[MVLK] Morf. M., A. Vieira, D. T. Lee and T. Kailath, "Recursive Multi-channel Maximum Entropy Method," Proc. 1977 Joint Automatic Control Conf.,San Francisco,1977, pp.113-117.

[Rab] Rabiner, L. R., "A Comparative Performance Study of Several Pitch Detection Algorithms," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-24, No.5, Oct. 1976, pp.399-418.

[RAS] Rabiner, L. R., E.S. Atal, and M. R. Sambur,"LPC Prediction Error - Analysis of Its Variation with the Position of Analysis Frame," IEEE Trans. Acoust. Speech Signal Proc., Vol. ASSP-25, Oct. 1977, pp.434-442.

[Reis] Reiser, J. F., "SAIL", Stanford Artificial Intelligence Laboratory Memo AIM-289, Aug. 1976.

[Rob] Robinson, E. A.,*Multichannel Time Series Analysis with Digital Computer Programs* San Francisco:Holden-Day, 1967.

[SV] Srinath, M.S.,M. M. Viswanathan,"Sequential Algorithm for Identification of Parameters of an Autoregressive Process," IEEE Trans. Aut. Contr.,AC-20,1975,pp.542-546.

[Vieira] Vieira, A., "Matrix Orthogonal Polynomials with Applications to Autoregressive Modeling and Ladder Forms", Ph.D. Dissertation, Stanford University, December 1977.

[Wak] Wakita, H., "Estimation of the Vocal Tract Shape by Optimal Inverse Filtering and Acoustic/Articulatory Conversion Methods," Monograph

No.9, Speech Communications Res. Lab. Inc., Santa Barbara, Calif., July 1972.

[Wil] Wilcox, C. R., "MAINSAIL Language Manual", internal memo for NIH Grant RR-00785, June 1977.

[WR] Wiggins, R.A.and E.A. Robinson,"Recursive Solution to the Multichannel Filtering Problem," J. Geophys. Res., vol. 70,1965, pp.1885-1891.

APPENDIX A

A Classification of Algorithms for ARMA Models and Ladder Realizations *

M. Morf, D. T. Lee, J. R. Nickolls and A. Vieira

Information Systems Laboratory
Stanford University, Stanford, CA 94305

Abstract

Applications of linear systems modeling have recently developed quite rapidly in speech modeling, seismic data processing, and other areas. Due to the diversity of these developments, there exists a plethora of methods for estimating the parameters of linear models given input-output data, transfer functions, or covariance functions. This paper attempts a systematic classification of existing least-squares modeling methods. Within this framework, we shall point out some recently developed algorithms that have many computational advantages over existing ones.

In particular, the methods of interest will be classified according to how the input/output data is acessed and according to its type. Data can be accessed either sequentially or in blocks; the data can be either input/output signals, transfer functions, or covariance functions. Since we consider state-space, autoregressive-moving average models, and the related ladder realizations, we shall distinguish the following three classes of algorithms: Riccati or square-root type methods, recently developed "fast" algorithms, and their ladder forms. While the first class typically requires computations of $O(n^3)$ or $O(n^2)$ with n equal to the number of model parameters, the "fast" forms only require operations and storage of $O(n)$. The ladder realizations have several advantages, such as lowest computational complexity and their stability "by inspection" properties.

In the appendices, we present an example of our new exact least-squares recursions for ladder forms, and show how to obtain stable partial minimal realizations of the joint impulse response - and covariance - matching type.

* *This work was supported by the Defense Communications Agency under contract DCA100-77-C-0005; the National Science Foundation under Contract NSF Eng75-18952; the Air Force Office of Scientific Research, Air Force Systems Command under Contract AF44-620-74-C-0068; the Joint Services Electronics Program under Contract N00014-75-C-0601; the Insitiute of Biomedical Technology in Zurich Switzerland; and by ARPA through the use of the Stanford Artificial Intelligence Laboratory facilities.*

I. Introduction

The long history and widespread use of linear modeling [K-S74] has resulted in many independently developed methods for determining such models. We attempt a classification of exact least-squares procedures that are *recursive* and *optimal* in some sense, and discuss some recently developed methods that have computational and structural advantages over existing ones. We shall only indicate examples of the much larger class of suboptimal or approximate methods.

In Section II we introduce the modeling problem by reviewing *external* and *internal* (linear) models, and consider the different types of observed data. We then introduce a *systematic classification in tableau form* of the various methods to be discussed. It should be stressed here that these least-squares modeling methods can in general only determine the unique *innovations representation* model [K-S74]. The parameters of this model are chosen to produce behavior *statistically* equivalent to the observed data.

In Section III we consider *batch* methods that are best suited for cases where data is accessed in *blocks*. This situation typically occurs when the data is in the form of a covariance function or transfer function.

In Section IV *recursive* (in time) algorithms that access the input/output signals *sequentially* are discussed. In the control context they are considered *on-line* methods [AE], [MKL]. In both sections III and IV we point out the *fast* versions which take advantage of certain matrix properties.

In Section V the *ladder* (or lattice) type realizations of the *fast* algorithms discussed in Sections III and IV are introduced [IS], [Mo]. These new methods have several nice properties from both the theoretical and application viewpoints.

In Appendix A we show how to obtain *stable partial minimal realizations* of the joint impulse response- and covariance-matching type. In Appendix B, we present an example of our new *exact least-squares recursions* for ladder forms.

II. The Modeling Problem

The many different types of linear models can be classified into "external" or input/output descriptions, and "internal" or state-space descriptions. We will first consider "external" descriptions, which are sometimes referred to as transfer function type models.

Let us consider a finite-dimensional linear system (FDLS) with inputs $\{ u(\cdot) \}$ and outputs $\{ y(\cdot) \}$. The outputs represent sampled data, such as speech where y's are scalars, or seismic signals from a geophone array where y's are r-vectors. The input-output relationship can be described by an *autoregressive - moving average* (ARMA) model.

$$y_i + a_1 y_{i-1} + \ . \ . \ . + a_n y_{i-n} = w_i \ , \tag{2.1a}$$

$$w_i = b_0 u_i + \ . \ . \ . + b_q u_{i-q} \ , \quad i \geq 0 \ , \quad n > q \geq 0 \ , \tag{2.1b}$$

where $\{ w(\cdot) \}$ is a moving average of a white noise sequence $\{ u(\cdot) \}$ and the values $\{ y_{-1}, \ . \ . \ . \ . \ , \ y_{-n} \}$ and $\{ u_{-1}, \ . \ . \ . \ , \ u_{-q} \}$ are *initial conditions.* The modeling problem here is to determine the model parameters a_i and b_i . Applying the *z*-transform

$$y(z) = \sum_{i=0}^{\infty} y_i z^{-i} \tag{2.2}$$

to equation (2.1) yields

$$a(z)y(z) = b(z)u(z) + \{ \textit{terms involving the initial conditions} \} \ , \tag{2.3a}$$

$$a(z) = z^n + a_1 z^{n-1} + \ . \ . \ . + a_n \ , \tag{2.3b}$$

$$b(z) = b_0 z^n + b_1 z^{n-1} + \ . \ . \ . + b_q z^{n-q} \ . \tag{2.3c}$$

With zero initial conditions and scalar processes, the ratio of $y(z)$ and $u(z)$ gives the transfer function $T(z) = b(z)/a(z)$. When $b(z) = z^k$, $k \geq 0$ then $\{ w(\cdot) \}$ is a white noise sequence and $\{ y(\cdot) \}$ is called an *autoregressive* (AR) process; the model is referred to as *all-pole*. Alternatively, when $a(z) = z^n$, $\{ y(\cdot) \}$ is a *moving average* (MA) process and an *all-zero* model is obtained.

Turning to "internal" models of the FDLS, we can consider $\{ y(\cdot) \}$ as being generated from $\{ u(\cdot) \}$ with a suitable initial condition $\{x_0\}$ by the *state-space* model

$$x_{i+1} = \Phi x_i + \Gamma u_{i+1} ,$$
$$y_i = \mathrm{H} x_i , \quad i \geq 0 . \qquad (2.4)$$

This model can be chosen to have the given transfer function

$$T(z) = \mathrm{H} (zI - \Phi)^{-1} z \Gamma = \frac{b(z)}{a(z)} . \qquad (2.5)$$

Note that for convenience we have used a model driven by u_{i+1}, rather than the more commonly used model driven by u_i [MKD] since they can be related [Mo].

Given the transfer function, a simple way to choose the matrices $\{ \mathrm{H}, \Phi, \Gamma \}$ is the "observer canonical" form

$$\Phi = Z - a_{1:n} \mathrm{H} , \quad \mathrm{H} = e_1^{\mathrm{T}} , \quad \Gamma = [b_q^{\mathrm{T}}, 0^{\mathrm{T}}]^{\mathrm{T}} , \qquad (2.6)$$

where $^{\mathrm{T}}$ denotes transpose, $a_{1:n}$ *f*0∧= $[a_1, \ldots, a_n]^{\mathrm{T}}$, b_q *f*0∧= $[b_0, \ldots, b_q]^{\mathrm{T}}$, Z is the "delay matrix": $Z_{i,j}$ *f*0∧= *if* $(j-i = 1)$ *then* 1 *else* 0 , and e_1 *f*0∧= $[1, 0, \ldots, 0]^{\mathrm{T}}$ is the first unit vector. The state-space model provides a convenient way of computing the covariance function of the output process. Even though the underlying model $\{ \mathrm{H}, \Phi, \Gamma \}$ or $\{ a_{1:n}, b_q \}$ is time-invariant, the output process $\{ y(\cdot) \}$ is in general not stationary due to "transients" caused by the initial conditions. However, if Φ is a stability matrix where all eigenvalues have magnitude less than one, then as $i \to \infty$ the transients eventually die out and the process becomes stationary. In the stationary case, the covariance is a function of $|i-j|$ given by $R_y(i,j) = \mathrm{H} \Phi^{|i-j|} \Pi \mathrm{H}^{\mathrm{T}}$, where Π is the state covariance matrix as $i \to \infty$ (see [DKM]).

ARMA models and state-space descriptions are just two different methods of representing the input-output relations of a FDLS, and they can be closely related to each other using matrix fraction descriptions (MFD's) [DKM]. A lesser-known class of FDLS representations are the *ladder realizations*, which are discussed in section V and are also related to the ARMA and state-space models.

In modeling a process as a FDLS driven by white noise, the observed data is usually available in one or more of the following forms:

a) input-output pairs $\{ u(\cdot), y(\cdot) \}$;

b) impulse response or related sequences such as moments, (or moment estimates, e.g. obtained from input-output pairs);

c) covariance functions or second order knowledge of inputs and/or outputs, (or estimates, e.g. from the impulse response).

Batch methods are used when data is accessed in blocks, as in b) and c); efficient methods for determining model parameters are recursive in order. *Recursive* (in time) methods are appropriate when data is accessed sequentially, as in a). Model parameters can then be estimated recursively both in order and time. Table 1 illustrates the modeling methods that we will discuss; they are divided first into batch and recursive groups, then by model class: Riccati or square-root methods, "fast" algorithms, and their ladder forms. Within each class the name or code for each method appears along with some pertinent references.

III. Batch Methods

When observed data is available in blocks, *batch* modeling methods are convenient. We will first consider AR models because of their widespread use [Makh]. The r [illegible] developed "linear predictive coding" (LPC) speech compression schem [illegible] example, are a direct application of least-squares fitting of AR models [AH], [IS], [Wak] (for a survey see [MG]). AR models have also been very useful in statistics [Par], [BJ], spectral analysis [UB], [Aka], and multichannel geophysical applications [Rob], [WR].

Normal Equations

It is well known in least-squares problems that the parameters of an AR model satisfy a set of *linear* equations called the *normal equations* (see [K-S74], [MG]) :

$$\mathbf{a}_n^T R_n = [1, -a_1, \ldots, -a_n] R_n = [R_\varepsilon, 0, \ldots, 0] \ . \tag{3.1}$$

An alternate form is the Yule-Walker equation [Par] :

$$\mathbf{a}_{1:n}^T R_{n-1} = [a_1, \ldots, a_n] R_{n-1} = [r_1, \ldots, r_n] \ . \tag{3.2}$$

In both forms R is a covariance matrix and the a_i's are the "predictor" or AR model parameters. R_ε is the "prediction error" or innovations covariance, a non-increasing function of n (typically the model order). In speech processing, two popular methods of obtaining the normal equation are the "autocorrelation" method and the "covariance" method [MG], but there exist many ways of estimating the covariance R_n [BJ], [MDKV], [Di]. General methods for solving such linear equations include Gaussian elimination (GE), Cholesky decomposition, Householder transforms [Hou], [GGMS]; however, they all require computations of $O(n^3)$.

The Levinson-Wiggins-Robinson (LWR) Algorithm

An algorithm that requires only $O(n^2)$ computations for the recursive solution of normal equations with *Töplitz* covariance matrices (corresponding to an assumption of *stationarity* of the process) was first described by Levinson [Lev] and later extended by Wiggins and Robinson [WR]. By making use of certain *shift-invariance* properties of Töplitz matrices (the i,j-th entries are only a function of $i-j$), this algorithm solves the normal equation via a set of recursions that update the AR parameters or the predictor parameters in increasing order [Rob], [K-S74]. The Levinson recursions are also closely related to the orthogonal Szegö polynomials [Sze], [GS], [KVM]. Levinson's algorithm plays a central role because it can be generalized to handle multi-channel data [WR], multi-dimensional or image processing problems [LKM], nonstationary processes with "shift-low-rank" [Mo], [FMKL], ladder realizations [MV], [MVK] and ARMA or state-space models [MKD], [MKL], [DKM].

ARMA Models

In state-space terms, the problem is to find a triple $\{H, \Phi, \Gamma\}$ such that $T_i = H\Phi^i\Gamma$, where $\{T_i\}$ is a given set of "first order" data characterizing the impulse response of a linear system. This is the *partial realization* problem [KFA], [DMK]. The central role in this realization theory is played by the *Hankel* matrix with entries $H_{i,j} = T_{i+j-1}$. The columns or rows of this matrix are known to span the state-space, so any method for finding a basis is a viable realization method. Of particular interest are methods for finding the smallest basis resulting in minimal order n realizations [HK], [Si], [YT]; they all require $O(n^3)$ operations.

From a transfer function point-of-view, the partial minimal realization problem is that of finding two relatively prime (matrix) polynomials $a(z)$ and $b(z)$ such that the given power series $T(z)$ matches say k terms of the expansion of $b(z)/a(z)$. This is the classical Padé approximation problem, which in the scalar case yields $T(z)\,a(z) = b(z) + \{\textit{terms in } z^{-i},\ i > k-n\}$. Equating coefficients of z^{-i}, $0 \le i \le n$, we get

$$\mathcal{H}_n a_n = 0_n\text{ , or }\ \mathcal{H}_n [a_n, \ldots, a_1]^T = -[T_{n+1}, \ldots, T_{2n}]^T, \qquad (3.3)$$

where $\mathcal{H}_n$ is a Töplitz matrix containing the reversed column ordered Hankel matrix H_n. Note the similarities here to the normal equation (3.1) and to Prony's method [MG].

Again, standard methods could be used to solve for a_n, but they all require computations of $O(n^3)$. However, if one takes advantage of the structure of the Hankel or Töplitz matrices, fast algorithms can be found. Such algorithms have been developed (in a coding theory context) by Berlekamp [Be] and for minimal realization by Massey [Mass]. Multivariable versions have also been developed [Mo], [DMK]. These recursions are strikingly similar to Levinson's recursions; the Berlekamp-Massey algorithm can also be related to orthogonal Lanczos polynomials [Lanc], [Mo]. An alternative method for obtaining *stable partial minimal realizations* is discussed in Appendix A. It can also be derived by considering a Gram-Schmidt (GS) orthogonalization on the columns of the Hankel matrix H_n or the Töplitz matrix H_n [Mo], or more generally by using projection methods [KKM].

Spectral Factorization and Innovations Representations

The problem of obtaining a model of a process $\{ y(\cdot) \}$, given its covariance function or second order information, is called *stochastic realization*. We are interested in representing $\{ y(\cdot) \}$ as the output of a linear model driven by white noise. In general, there exist many such models, however the only stable and stably invertible model is the (unique) *innovations representation* (IR) [K-S74]. The inverse model is the whitening filter that produces a white noise process, the innovations $\{ \epsilon(\cdot) \}$, when driven by the observed data. In discrete-time or time series analysis, the innovations are the one-step prediction errors of the observations. If the process $\{ y(\cdot) \}$ is stationary, the problem of obtaining the IR essentially reduces to one of spectral factorization.

An efficient method for obtaining the spectral factors of $S_y(z)$

$$S_y(z) = b(z)\, b(-z) \,/\, a(z)\, a(-z) \,, \tag{3.4}$$

is given as a two-step procedure in [DKM], [MKD], [Mo]. In the stationary and scalar process case considered here, the truncated or one-sided power spectrum $S_+(z)$ of $\{ y(\cdot) \}$ is formed from the covariance sequence. A minimal realization algorithm is then used to approximate $S_+(z)$ by $q(z)/a(z)$, where $1/a(z)$ is the AR-part of the desired model. From

$$\begin{aligned} S_w(z) &= a(z)\, S_y(z)\, a(z^{-1}) = a(z)q(z^{-1}) + q(z)a(z^{-1}) \\ &= b(z)\, b(z^{-1}) \qquad (3.5) \end{aligned}$$

it can be seen that the spectral factorization problem for $\{ y(\cdot) \}$ is now reduced to the simpler factorization problem for a MA-process $\{ w(\cdot) \}$, where the factor $b(z)$ is the MA-part of the desired model. It can be obtained by Cholesky and other factorization procedures. Thus $\{ y(\cdot) \}$ is modeled by the cascade of the AR and MA parts driven by the innovations, a white noise.

In the time-domain this corresponds to a factorization of the (stationary) covariance (a Töplitz matrix) into triangular factors. The "fast-Cholesky" factorization given by [Mo] is an efficient algorithm for stationary and non-stationary covariance matrices with "shift-low-rank". It should be noted that many popular covariance estimates have this property.

IV. Recursive "in Time" Methods

When the observed data is available as input-output pairs that must be accessed sequentially, recursive modeling methods are the most appropriate. Many recursive least-squares methods have been developed in the identification and control area; they typically involve solving Riccati-type equations and have computation and storage requirements of $O(n^3)$ and $O(n^2)$ respectively. Recently "fast" algorithms have been developed with reduced computations and storage of $O(n)$ using ARMA or ladder realizations .

An important set of least-squares recursive methods for AR-type models is described in detail in [SLG] and more recently in [MKL]. The discussion includes least-squares (LS), weighted least-squares (WLS), generalized least-squares (GLS), instrumental variable (IV) and recursive maximum-likelihood (RML1,2) methods for ARMA models. All these methods solve a Riccati equation that recursively updates the inverse of the matrix appearing in the normal equation of the problem.

An alternative to the Riccati equations are the square-root forms discussed for instance in [MK]. They make use of the numerically preferrable orthogonal transformations [Hous], [GGMS].

A special case of the IV method is obtained by using the n-step delayed outputs as instrumental variables. This can be shown to be equivalent to a minimal realization problem given (estimated) covariances R . Recall that in the given (estimated) covariance case we discussed a two step procedure. The first step was to obtain a minimal realization, or rational approximation of $S_+(z)$ by $q(z)/a(z)$, or in the time-domain of R_+ by $Q\,A^{-1}$ [DKM], [MKD]. In matrix notation we obtain

$$R_+ A = Q \, , \tag{4.1}$$

where A and Q are banded matrices of "band width" n , if the underlying linear model is of that minimal order. The first column of (4.1) corresponds to equation (3.3)

$$R_n a_n = H_n a_n = 0_n \, . \tag{4.2}$$

where $\bar{R}_n$ is the (2,1) block entry of the appropriately partitioned triangular Töplitz matrix R_+. The matrix $\bar{R}_n$ is the cross-covariance of the last n observations and. the same set n time-steps delayed. $\bar{R}_n$ clearly plays here the role of the reversed ordered Hankel matrix H_n. By noting that the Riccati-type equation can be interpreted as a recursion for (low rank) updating of the inverse of a matrix, we see that the IV method can be viewed as a recursive (in time) updating procedure for the minimal realization solution for a in equation (3.3).

Fast Algorithms for Recursive "in Time" Methods

In [MKL] the development of "fast" algorithms for the recursive least-squares methods is discussed in detail. Efficient recursions for time and/or order updates for AR-type models were first derived in [Mo]. The basic idea there was the observation that the matrices encountered in many least-squares problems have a "shift invariance" or a "shift-low-rank" property. It can be characterized by the (low) rank ρ of the shifted difference of a matrix M: $\rho\,[\,M - Z^T M Z\,]$, where the "delay" matrix Z was defined in Section II. This property is generated by the fact that these matrices are sums of products of Töplitz or Hankel matrices. It can also be used to obtain fast Cholesky algorithms for MA processes, thus obtaining *recursive* whitening filters of the AR type (e.g. RMH5 algorithm in [Mo]).

Similarly we can obtain general LWR recursions in order and time for AR processes, i.e. updating the MA prediction (whitening filter) parameters a_i. A surprising feature of the fixed-order recursive-in-time algorithms is that explicit updating of the covariance estimate is unnecessary, basically because the model parameters are an *implicit characterization of the covariance*. Since the details of these algorithms can be found in [MKD],[DKM],[Mo],[FMKL], we shall only give a comparison of the LWR-type algorithms, assuming that the reader is already familiar with the Levinson recursions as described in [Wie], [WR], or more recently [K-S74].

The recursions for the generalization of the LRW algorithm for covariance matrices exhibiting a *shift invariance* property have a very similar form to the original Levinson recursions. However, in contrast to the two solutions required in the LWR algorithm, the so-called forward and backward predictors, we require in general more solutions for non-Töplitz matrices. It turns out that the "shift-low-rank" ρ of the covariance matrix, regardless of its size, is equal to the number of solutions required in the recursions. For the case of covariance estimates that can be written as products of two Töplitz matrices (typically containing input-output data), the number of solutions required is at most four in the scalar case, and $2m+2$ for m-channel data.

For combined ARMA models we can either attempt to model first the AR or the MA part and then try to estimate the remaining part of the model. In the batch methods of Section III we discussed ways of obtaining the AR part first via minimal realization and then the MA part via spectral factorization. The other order of first obtaining the MA part could be obtained by working with (an estimate of) the inverse of the covariance matrix, the so-called *information* matrix [MK].

The cascaded approach can be carried out also in time recursive form by estimating the AR part via a (fast) recursive form of the IV method, as discussed above, cascaded by the fast Cholesky recursions for a MA process (e.g. RMH5 in [Mo]). The only difficulty now is that both parts estimate the models in the so-called controller or "tapped delay line" realization, a dual form to the observer form, which cannot be merged by inspection.

The joint innovations representation approach discussed in Section III and Appendix A is ideally suited for recursive in time methods. Even though the driving inputs (conceptually the innovations) are usually not available, they can be replaced with their best estimates obtained by using the best current parameter estimates. This is clearly only possible for methods with sequential data access. It turns out that this seemingly "suboptimal" approach of substitution has itself optimality properties (see e.g. [SLG]); a similar situation occurs in detection of unknown signals, and in the famous separation result of linear quadratic control using state estimates [KFA]. The recursive maximum likelihood methods in [SLG] and [MKL] can be derived from such an approach. Once estimates of current prediction errors are obtained, they can conceptually be treated as known data, and entered for instance in normal equation expressions. The only problem that might arise lies in theoretical proofs of the convergence of such methods.

V. Ladder Realizations

In Section II we discussed the realization of a given transfer function $T(z) = b(z)/a(z)$ via transfer function or state-space type models such as ARMA, controller, or observer canonical forms. If the roots of $a(z)$ are known, $T(z)$ can be represented by a partial fraction expansion. Using polynomial evaluation formulas we can obtain the so called Jordan canonical or parallel decomposition form (even for matrix transfer functions) [Mo]. The Jordan form has the nice property that stability can be checked by merely inspecting the magnitude of the roots. Finding the roots, however, is numerically sensitive.

The ladder (or lattice) canonical realizations provide a very promising alternative. They also have the property that stability and even minimum-phase can be checked by inspecting the PARCOR or *reflection coefficients* [IS], [Wak], [MG], [Mo], [Cla2]. In contrast to the methods for finding roots of polynomials this requires only a *finite* algorithm, the Schur-Cohn test for stability. This is actually equivalent to the Levinson or orthogonal Szegõ polynomial recursions performed in decreasing order on $\mathbf{a}_i$ or $a_i(z)$, [K-S74]. Given the stationary covariance matrix R, i.e. second order information, the $\mathbf{a}_i$'s and the reflection coefficients can be computed via the LWR algorithm. From a stochastic process point of view we can identify these coefficients with the *partial correlation (PARCOR) coefficients* or singular values. They also have physical significance in the scattering theory of waves [IS], [Wak], [K-S74], [MV], [MVK].

The Levinson algorithm can actually be carried out using *only* the reflection coefficients as parametrization, since the inner product k_i of a vector $\mathbf{r}\, f0\wedge = [r_1, \ldots, r_i]^T$ and the coefficient vector $\mathbf{a}_{i-1}$ can be obtained as the current output k_i of a ladder realization driven by a previous input sequence containing $\{r_1, \ldots, r_i\}$. Similarly we can translate other algorithms for AR models, such as the various generalized Levinson algorithms [Mo], [DKM], [LKM], Appendix A, into their ladder form equivalents as in [MVK], [MV], Appendix B. These forms are of interest by virtue of their stability properties and their numerical robustness -- they typically require sample correlation operations. These forms have also canonical [Mo] and invariance properties [MVK], as well as minimal storage requirements for modeling algorithms, as seen from a comparison of the the recursions for the PARCOR and the a_i parameters in Appendix B.

ARMA models

Recall that the first step of realizing an ARMA model in Section III was a minimal realization problem. The solution to this MR problem can be carried out in ladder form by using a Berlekamp Massey (BM) - type algorithm. These recursions are actually very similar to the LWR recursions, as noted in [Mo]; therefore we can use an analogous derivation to obtain ladder forms for the BM recursions, as presented in [GrMo].

Alternatively, the joint IR approach explored in Appendix A, leads (even for scalar) processes to the theory of multichannel ladder realizations of the AR type discussed above. Since we embedded the underlying ARMA model in a two channel AR model, the IR model will again be of order $2n$, i.e. non-minimal. This would also hold for a ladder realization.

Minimal models can be obtained by merging the AR and MA parts in the observer form. It is also possible to obtain a minimal rational ladder form [MG], [Mo]. The basic idea in state-space terms is to add a suitable input matrix (Γ) or output matrix (H) to a ladder form realization of the AR part of the model; this is possible since the ladder forms are controllable (or their dual observable) [Mo].

The second step of the stochastic realization procedure of Section III requires a spectral factorization for the determination of the MA part. As indicated, we need to determine the triangular factors of the (banded) covariance matrix of the MA process. They can be obtained from the Cholesky factors or the RHS of the LWR recursions. Similarly it is possible to obtain the ladder realizations of the MA part, since the fast Cholesky recursions "by rows" have the form of a state-space equation with a dynamic matrix Φ that has precisely the same form as the Φ matrix of a feedback ladder form in state-space notation [Mo]. As for the LWR recursions, there similarly exists a ladder form of the fast Cholesky algorithm that requires

AD-A070 050 STANFORD UNIV CALIF INFORMATION SYSTEMS LAB F/G 17/2
FAST ALGORITHMS FOR SPEECH MODELING.(U)
DEC 78 M MORF, D T LEE DCA100-77-C-0005
UNCLASSIFIED NL

3 OF 3
AD
A070050

END
DATE
FILMED
7-79
DDC

only reflection coefficients as parametrization.

Ladder Forms for Recursive "in Time" Methods

The ladder forms for *exact* least-squares solution to AR modeling have been developed in [MV]. In Appendix B, we shall present the simplest one of the many possibilities of such ladder forms, the "prewindowing" case [MDKV]. It is interesting to note that the partial correlation coefficients are computed as sample cross correlations between the "forward" and "backward" prediction errors as expected from the *stochastic derivations of* the ladder *forms* [Wak], [Mo], [SKM].

The ladder forms of the GLS and RML1/2 methods discussed in [SLG], [MKL] and Section IV can be obtained by embedding the models in an appropriately augmented AR model as in Appendix A . The IV method led to nonsymmetric Riccati equations, therefore the fast versions also require a nonsymmetric form of the LWR recursions. However it is clear that these recursions are then of the BM type since this algorithm also works with nonsymmetric (though triangular) Töplitz matrices. Therefore, we could obtain "nonsymmetric" ladder forms of the type given in [GrMo]. Although the final algorithms of these ladder forms are simple to implement, the exposure of the "shift-invariance" is in general nontrivial [MV].

Our preliminary experience with the numerical properties of these algorithms has been very encouraging; in general ladder realizations are superior to direct forms for computing estimates of the coefficients of $a(z)$ and $b(z)$. Stochastic approximation or gradient type methods using ladder forms can be obtained easily, e.g. [SV]; however, they have drawbacks similar to other stochastic approximation methods, especially for covariance matrices with extreme eigenvalue distributions.

Table I: Classification of Least-Squares Modeling Methods

MODEL	METHOD	Riccati, Square-Root Arb. Linear Equations $O(n^3)$ or $O(n^2)$	Chandrasekhar, Fast Cholesky, Levinson Shift-Invariant, Orthogonal Polynomials $O(n^2)$ or $O(n)$	Ladder type Orthogonality $O(n)$
	BATCH			
AR-type	Normal Equation	GE,GS,HH: [GGMS],[Hou]	(Gen.)Levinson,LWR,Szegö-poly.: [K-S74]	Orthog:[IS],[Wak],[MG],[Mo]
ARMA	AR-Part, Minimal Realization	Hankel-Matrix: [HK],[Si]	Töplitz-M.: [Lanc],[Be],[Mass], [DMK], Appendix A	Lanczos Ladder forms: [Mo], [GrMo], [KKM]
	MA-part, Spectral Factorization	Innov.Rep.: [GK]	Chandrasekhar,Fast-Cholesky: [DKM], [Mo]	Fast-Cholesky-Ladder: [Mo], [GrMo]
	RECURSIVE			
AR-type	Recursive Least-Squares	LS,GLS,RE,SQ: [SLG],[MK]	Gen.LWR,RMH: [MKL],[MDK], [MDKV], [Mo]	Recursive Ladder Forms: [MV], Appendix B
ARMA	Augmented Rec. L.-S.	RML1,2,RE,SQ: [SLG],[MK]	Aug.Gen.LWR: [MKL]	Aug.Rec.Ladder Forms: [MV]
AR(MA)	Aug. Nonsym. Rec. L-S	IV,Nonsym.RE,SQ:[SLG],[MK]	Nonsym.Gen.LWR,RMH: [MKL],[Mo]	Nonsym.Ladder Forms: (?)
ARMA	AR-part, Rec. Minimal Realiz.	Spec.IV,Nos.RE,-SQ:Sec.IV	Nonsym.Gen.LWR,RMH,BM-type: Section IV, [Mo]	Nonsym.Ladder Forms: Sec.IV
	MA-part, Rec. Spectral Fact.	Cholesky,RE,SQ: [GK],Sec.IV	Fast Cholesky,RMHS: Sec.IV,[Mo]	Rec.Feedb.Ladder: Sec.IV,[Mo]

Appendix A: Stable Partial Minimal Realizations

In this appendix we show how to obtain stable partial minimal realizations of the joint impulse-response and covariance-matching type. It will turn out that we can obtain an ARMA model by imbedding it in a two channel AR modeling problem. Given an ARMA model as represented by the difference equation of section II, we can rewrite it as (let $q = n$)

$$y_t + a_1 y_{t-1} + \ldots + a_n y_{t-n} - b_1 u_{t-1} - \ldots - b_n u_{t-n} = b_0 u_t , \qquad \text{(A1)}$$

or

$$\mathbf{a}^T \mathbf{y}_t - \mathbf{b}_1^T \mathbf{u}_t = b_0 u_t , \quad \text{where}$$

$$\mathbf{a}^T = [\, 1 , a_1 , \ldots , a_n \,], \quad \mathbf{y}_t^T = [\, y_t , \ldots , y_{t-n} \,],$$

$$\mathbf{b}_1^T = [\, 0 , b_1 , \ldots , b_n \,], \quad \mathbf{u}_t^T = [\, u_t , \ldots , y_{t-n} \,] .$$

Now consider the following augmented equation

$$\begin{bmatrix} \mathbf{a}^T & -\mathbf{b}_1^T \\ 0 & e_1^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_t \\ \mathbf{u}_t \end{bmatrix} = \begin{bmatrix} b_0 u_t \\ u_t \end{bmatrix}, \qquad \text{(A2)}$$

(e_1^T is the first unit vector). This equation can be interpreted as an AR model for the joint process $\{ y, u \}$ [Mo], since the RHS is equal to the joint innovations of $\{ y, u \}$, since

$$\varepsilon_t = \begin{bmatrix} \varepsilon^y_t \\ \varepsilon^u_t \end{bmatrix} = \begin{bmatrix} y_t - \hat{y}_{t|t-1} \\ u_t - \hat{u}_{t|t-1} \end{bmatrix} = \begin{bmatrix} b_0 u_t \\ u_t \end{bmatrix} . \qquad \text{(A3)}$$

Deterministic Case

We first consider the deterministic case where we are given impulse response data or the Markov parameters. Writing the input/output relationship in matrix notation (see sec. III) yields

$$\begin{bmatrix} T_n & 0 \\ H_T & T_T \end{bmatrix} \begin{bmatrix} \mathbf{a}_n \\ 0_T \end{bmatrix} = \begin{bmatrix} \mathbf{b}_n \\ 0_T \end{bmatrix} , \qquad \text{(A4)}$$

where T_n is a lower-triangular and H_T is a full Töplitz matrix of the Markov parameters (H_T is the column reverse ordered Hankel matrix). Letting $T \to \infty$, we get the normal equation

$$\begin{bmatrix} T_n^T & H_T^T \\ I & 0 \end{bmatrix} \begin{bmatrix} T_n & I \\ H_T & 0 \end{bmatrix} \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} =$$

$$= \begin{bmatrix} R_n & T_n^T \\ T_n & I \end{bmatrix} \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} = \begin{bmatrix} e_1 b_0 b_0^T & e_1 b_0 \\ e_1 b_0 & e_1 \end{bmatrix} . \qquad \text{(A5)}$$

Stochastic Case

From a stochastic process point of view we can express the normal equation associated with the augmented AR model as

$$E\left\{ \begin{bmatrix} y_t \\ u_t \end{bmatrix} [\, y_t^T \; u_t^T \,] \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} \right\} = E\left\{ \begin{bmatrix} y_t \\ u_t \end{bmatrix} [\, u_t b_0 \; u_t \,] \right\}$$

$$= \begin{bmatrix} R_n & T_n^T \\ T_n & I_n \end{bmatrix} \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} = \begin{bmatrix} e_1 b_0 b_0^T & e_1 b_0 \\ e_1 b_0 & e_1 \end{bmatrix} . \qquad \text{(A6)}$$

We can solve for the normal equation of a_n :

$$R_n a_n = [\, R_n - T_n^T T_n \,] a_n = [\, H_\infty^T H_\infty \,] a_n = e_1 R^\epsilon{}_n . \qquad \text{(A7)}$$

The equations (A5), (A6), and therefore the non-Töplitz equations (A7) (!) can be solved recursively with the LWR algorithm. Note that if $R^\epsilon{}_k = 0$, the minimal order $n = k$. We could bring equations (A5), (A6) into a more familiar form by the interleaving permutation $(1, 3, 5, \ldots, 2n-1, 2, 4, 6, \ldots, 2n+2)$, cf. [MDHV], to convert the two-process covariance matrix into a n by n block Töplitz matrix, with 2 by 2 blocks, however the LWR algorithm clearly applies to both representations with suitable modifications.

Thus we have shown that the joint impulse response/covariance matching problem is equivalent to solving a set of normal equations associated with a two channel AR modeling problem. Since the predictor for the joint process is *triangular* and *minimum phase*, the denominator $\underline{a}_n$ of the underlying ARMA model is also *minimum phase* and therefore *stable*, (for all k).

Equations (A5) and the elegant stability proof were actually first obtained by Claerbout [Cla1] via a least-squares rational approximation. The connections between the joint innovations representation, the augmented normal equations, and the Hankel matrix were pointed out in [Mo] and also in [MDKV], [MKD], [DKM], where algorithms were given to solve equations of the type seen in (A6) and (A7). For the special case where R has a "shift-low-rank" of one of the type (B5), called the post-windowing method in [MDKV], a Levinson-type algorithm was given recently by [MR]. The stability property of the AR model was proved there using a somewhat more complicated Lyapunov technique.

Appendix B: LS-Recursions for Ladder Forms

The Prewindowing Case

Given a series of observations $\{y(t), 0 \le t \le T\}$, where $\{y(\cdot)\}$ can be m vectors, we wish to find the least-squares one-step predictor of order p parametrized by the (matrix) coefficients $\{A_{p,T}(i), i=1, \ldots, p\}$. We can define many different squared error criteria $E_{p,T}$, for instance as a function of s and f in

$$E_{p,T} \triangleq \sum_{t=s}^{f} \epsilon^T{}_{p,T}(t)\, \epsilon_{p,T}(t) \quad , \quad \epsilon_{p,t} \triangleq A^T{}_{p,T}\, y[t:t-p] \, ,$$

$$A^T{}_{p,T} \triangleq [\, I \, , A^T{}_{p,T}(1), \ldots , A^T{}_{p,T}(p)] \, , \quad y^T[t:t-p] \triangleq [y_t{}^T , \ldots , y_{t-p}{}^T] \qquad \text{(B1)}$$

An obvious choice from an *innovations* point-of-view is $(s=0, f=T)$, the "pre-windowing" case [MDKV]. If $s = p$ and $f = T$ the so-called "covariance" method is obtained, and if $s = 0$ and $f = T + p$ we get the pre- and post-windowed case or the "correlation" method [MG]. The total squared error can be expressed as

$$E_{p,T} = tr\,\{ A^T{}_{p,T} R_{p,T} A_{p,T} \} \quad , \quad R_{p,T} = Y_{p,T} Y^T{}_{p,T} \, ,$$

$$Y_{p,T} \triangleq [\, y[0:-p] \, , y[1:-p+1] \, , \ldots , y[T:T-p] \,] \qquad \text{(B2)}$$

Thus the problem of determining $A_{p,T}$ by minimizing $E_{p,T}$ leads to

$$R_{p,T} A_{p,T} = [R^\epsilon{}_{p,T} \, , 0, \ldots , 0]^T , \quad tr\, R^\epsilon{}_{p,T} = min\, E_{p,T} \, . \qquad \text{(B3)}$$

Although $R_{p,T}$ is not Töplitz, it still carries a certain shift-invariance structure, given by the following identities

$$\begin{aligned} R_{p,T} &= R_{p,T-1} + y[T:T-p]\, y[T:T-p]^T \\ &= \begin{bmatrix} x & x & x \\ & x & R_{p-1,T-1} \end{bmatrix} = \begin{bmatrix} R_{p-1,T} & & x \\ & x & x & x \end{bmatrix} . \end{aligned} \qquad \text{(B5)}$$

Define the backward predictor $B_{p,T}$ and the smoothing errors $C_{p,T}$

$$B^T{}_{p,T} R_{p,T} \triangleq [\, 0 , \ldots , 0 \, , R^r{}_{p,T}] ; \quad C^T{}_{p,T} R_{p,T} \triangleq y[T:T-p] \, . \qquad \text{(B6)}$$

Then the forward and backward prediction errors (innovations), $\epsilon_{p,T}$, and $r_{p,T}$, and an auxiliary scalar $\gamma_{p,T}$ can be defined by

$$[\, \epsilon^T{}_{p,T} \, , r^T{}_{p,T} \, , \gamma_{p,T} \,] \triangleq y^T[T:T-p]\, [\, A_{p,T} \, , B_{p,T} \, , C_{p,T} \,] \, .$$

Order Update Recursions

Using the three shift-invariance identities for $R_{p,T}$ (B5) and using some symmetry properties, the order update recursions for $A_{p,T}$, $B_{p,T}$, $C_{p,T}$, $R^{\epsilon}_{p,T}$, and $R^{r}_{p,T}$ are

$$\begin{aligned} A^{T}_{p+1,T} &= [A_{p,T}, 0]^{T} - K^{T}_{p,T} R^{-r}_{p,T-1} [0, B^{T}_{p,T-1}]^{T} \\ B^{T}_{p+1,T} &= [0, B_{p,T-1}]^{T} - K_{p,T} R^{-\epsilon}_{p,T} [A^{T}_{p,T}, 0]^{T} \\ C^{T}_{p+1,T} &= [C^{T}_{p,T}, 0]^{T} + r^{T}_{p+1,T} R^{-r}_{p+1,T} B^{T}_{p+1,T} \quad \text{where} \\ K_{p,T} &= [\textit{last block row of } R_{p+1,T}] [A^{T}_{p,T}, 0]^{T} \\ &= [0, B^{T}_{p,T-1}] [\textit{first block row of } R_{p+1,T}]^{T}. \end{aligned} \tag{B7}$$

$$\begin{aligned} R^{\epsilon}_{p+1,T} &= R^{\epsilon}_{p,T} - K^{T}_{p,T} R^{-r}_{p,T-1} K_{p,T} \\ R^{r}_{p+1,T} &= R^{r}_{p,T-1} - K_{p,T} R^{-\epsilon}_{p,T} K^{T}_{p,T}. \end{aligned} \tag{B8}$$

The order update recursions are very similar to the multivariate version of the Levinson algorithm, and a similar set of recursions for time-update can also be obtained [MDKV],[Mo].

Ladder Type Realization

Premultiplying the above equations by $y[T{:}T-p+1]$, we obtain the following order update recursions for $\epsilon_{p,T}$, $r_{p,T}$, $\gamma_{p,T}$

$$\begin{aligned} \epsilon_{p+1,T} &= \epsilon_{p,T} - K^{T}_{p,T} R^{-r}_{p,T-1} r_{p,T-1} \\ r_{p+1,T} &= r_{p,T-1} - K_{p,T} R^{-\epsilon}_{p,T} \epsilon_{p,T} \\ \gamma_{p+1,T} &= \gamma_{p,T} + r^{T}_{p+1,T} R^{-r}_{p+1,T} r_{p+1,T}. \end{aligned} \tag{B9}$$

The "*Kalman gain*" $K_{p,T}$ is obtained from [MV] as follows

$$K_{p,T+1} = K_{p,T} + r_{p,T} \epsilon^{T}_{p,T+1} / (1 - \gamma_{p-1,T}) \tag{B10}$$

and the *reflection* or *PARCOR coefficients* are obtained by

$$K^{\epsilon}_{i,T} \triangleq K_{i,T} R^{-\epsilon}_{i,T}; \quad K^{r}_{i,T} \triangleq K^{T}_{i,T} R^{-r}_{i,T-1}. \tag{B11}$$

The initial conditions are given by

$$\epsilon_{0,T} = r_{0,T} = y_T \; ; \qquad \gamma_{-1,T} = 0 \; ;$$

$$R^{\epsilon}_{0,T} = R^{r}_{0,T} = \sum_{t=0}^{T} y_t y_t^{\mathsf{T}} = R^{\epsilon}_{0,T-1} + y_T y^{\mathsf{T}}_T \; ;$$

for $p \geq T$:

$$\epsilon_{p,T} = \epsilon_{T,T} ; \; r_{p,T} = r_{T,T} ; \; \gamma_{p,T} = \gamma_{T,T} \; ;$$

$$R^{\epsilon}_{p,T} = R^{\epsilon}_{T,T} ; \; R^{r}_{p,T} = R^{r}_{T,T} ; \; K_{p,T} = 0 ;$$

$$K_{p,p+1} = y_0 \; \epsilon_{p,p+1}{}^{\mathsf{T}} .$$

As the dual to the stochastic forms in [IS], [Wak], [Mo], [SKM], equations (B8)-(B11) are a complete set of *order and time update recursions* to obtain the ***exact least-squares*** ladder form predictor, which is shown in Figure 1.

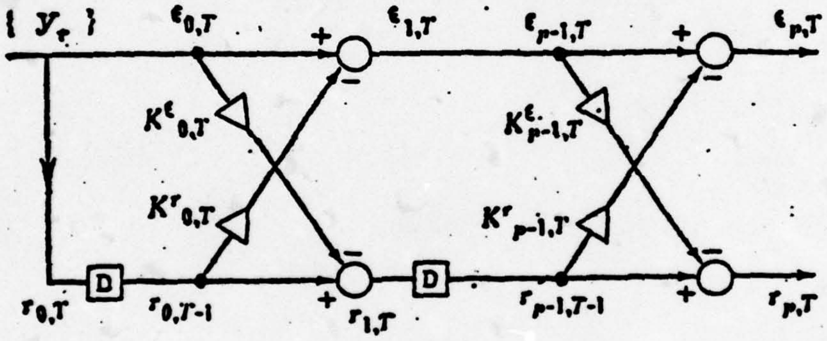


Figure 1. Ladder realization of exact one-step least-squares predictor.

The recursion (B10) computes the *sample cross-covariance* of the *forward and backward innovations*, using the *optimal weighting* $1/(1-\gamma_{.,.})$, compared to other suboptimal schemes [SV]. In the scalar case $R_{p,T}>0$ if $y_0 \neq 0$, or in general if $\gamma_{p-1,T}<1$, since $0 \leq \gamma_{p,T} \leq 1$ [MDKV]. If $m>1$, we require $T \geq p+m$. These singularities can be avoided by including a priori estimates of the covariance R_n, or equivalently including a weighted norm of the predictor a_n in the error criteria $E_{p,T}$. Several such modifications have been proven useful in actual implementations.

References

[AE] Astrom,K.J. and P.Eykhoff,"System Identification - A Survey", Automatica,v.7,1971,pp.123-162.

[AH] Atal,B.S.,S.L.Hanauer,"Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Amer., v.5,1971,pp.637-655.

[Aka] Akaike,H.,"Power Spectrum Estimation through Autoregressive Model Fitting," Ann. Inst. Stat. Math., vol. 21,1969,pp.407-419.

[Am] Ambartsumian, V. A.,"Diffuse Reflection of Light by a Foggy Medium," Dokl. Akad. Sci. SSSR, vol.38, 1943, pp.229-322.

[Be] Berlekamp, E.R., *Algebraic Coding Theory*, McGraw - Hill, New York, 1968.

[BJ] Box, G.E., and G. M. Jenkins, *Time Series Analysis Forecasting Forecasting and Control*. San Francisco, Ca.:Holden-Day, 1970.

[Chan] Chandrasekhar, S., Radiative Transfer, Oxford Univ. Press: Oxford 1950; also Dover: New York 1960.

[Cla1] Claerbout, J. F.,"Estimation of a Rational Function on the Unit Circle,". unpublished memo.

[Cla2] Claerbout, J. F., *Fundamentals of Geophysical Data Processing*, McGraw-Hill: New York, 1976.

[Di] Dickinson, B. W.,"Two Recursive Estimates of Autoregressive Models Based on Maximum Likelihood," submitted for publication.

[DKM] Dickinson, B., T. Kailath and M. Morf, "Canonical Matrix Fraction and State-Space Descriptions for Determinictic and Stochastic Linear Systems," IEEE Transactions of Automatic Control, vol. AC-19, 1974, pp. 656-667.

[DMK] Dickinson, B., M. Morf and T. Kailath, "A Minimal Realization Algorithm for Matrix Sequences," IEEE Transactions on Automatic Control, vol. AC-19, 1974, pp. 31-38.

[FMKL] Friendlander, B., M. Morf, T. Kailath and L. Ljung, "New Inversion Formulas for Matrices Classified in Terms of Their Distance from Toeplitz

Matrices,"submitted to SIAM J. A. Math.

[GGMS] Gill, P.E., G.H. Golub, W. Murray, and M.A. Saunders, "Methods for Modifying Matrix Factorizations," Stanford Univ. Computer Science Rept. STAN-CS-72-322, 1972.

[GK] Gevers, M., and T. Kailath, "An Innovations Approach to Least - Squares Estimation, Part VI," IEEE Trans. Automat. Contr. vol. AC-18, 1973, pp. 588-600.

[GrMo] Gray, R. M.,"Source Encoding," lecture by M. Morf, ISL Report, NSF-Workshop, Stanford University, Summer 1974.

[GS] Grenander, U., and G. Szegö, *Toeplitz Forms and Their Applications.* Berkeley, Calif.:Univ. California Press,1958.

[HK] Ho,B.L., and R.E. Kalman, "Effective Construction of Linear State - Variable Models from Input - Output Functions," Regelungstechnik, vol. 14, 1966, pp. 545-548.

[Hou] Householder, A. S., *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York: 1964.

[IS] Itakura,F., and S. Saito,"Digital Filtering Techniques for Speech Analysis and Synthesis," Conf. Rec., 7th Int. Congr. Acoust.,Budapest,1971,Paper 25 C 1.

[KFA] Kalman, R.E., P.L. Falb and M.A. Arbib,*Topics in Mathematical System Theory*, McGraw-Hill, New York: 1969.

[KKM] Kung. S, T. Kailath, and M. Morf, "A Fast Projection Method for Canonical Minimal Realization," Proc. IEEE Conf. on Decision and: Control, pp.1301-1302, Dec. 1976.

[KVM] Kailath, T., A. Vieira and M. Morf, "Inverses of Toeplitz Operatiors, Innovations, and Orthogonal Polynomials," submitted for publication.

[Lanc] Lanczos, C.,"An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," J.Res.Nat. Bur.Standards, vol.45,1950,pp.255-282.

[Lev] Levinson, N.,"The Wiener RMS (root mean square) Error Criterion in

Filter Design and Prediction," J. Math. Phys.,vol.25,1947, pp.261-278.

[LKF] L.Ljung,T.Kailath,B.Friedlander,Proc. IEEE,v.63,1976,pp.131-139.

[LKM] Lévy,B.,S.Y.Kung,M.Morf,Multi.Dim.Sys.Conf.,Monterey,Dec.1976.

[Mass] Massey, J.L., "Shift-Register Synthesis and BCH Decoding," IEEE Trans. on Inform. Theory, vol. IT-15, 1969, pp. 122-127.

[Makh] Makhoul, J. "Linear Prediction: A Tutorial Review," Proc. IEEE, vol. 63, 1975, pp. 561-580.

[MDKV] Morf, M., B. Dickenson, T. Kailath and A. Vieira, "Efficient Solution of Covariance Equations for Linear Prediction," submitted to IEEE-ASSP 1976.

[MG] Markel, J.D., and A.H. Gray, Jr.,*Linear Prediction of Speech*, Springer-Verlag, Berlin: 1976.

[MK] Morf, M., and T. Kailath, "Square-Root Algorithms for Least- Squares Estimation ,"IEEE Trans.on Automat. Control,vol. AC-20, 1975, pp. 487-497.

[MKD] Morf, M., T. Kailath, and B. Dickinson, "General Speech Models and Linear Estimation Theory," in Speech Recognition, pp.157-182, New York:Academic Press, 1975.

[MKL] Morf, M., T. Kailath and L. Ljung, "Fast Algorithms for Recursive Identification," Proc. IEEE Conf. on Decision and Control,pp.916-921,Dec. 1976.

[Mo] Morf, M., "Fast Algorithms for Multivariable Systems," Ph.D. dissertation, Stanford University, Stanford, California 1974.

[MR] Mullis, C. T., and R. A. Roberts, "The use of Second-Order Information in the Approximation of Discrete-Time Linear Systems," IEEE Trans. Acoust. Signal Processing, vol. ASSP-24,1976, pp.226-238.

[MV1] Morf. M., A. Vieira "Recursive Least-Squares Estimation of Partial Correlations," submitted to IEEE-ASSP 1977.

[MV2] Morf. M., A. Vieira "Multichannel Least-Squares Estimation of Partial Correlations," submitted to IEEE-AC,1977.

[MVK] Morf. M., A. Vieira and T. Kailath, "The Multi-channel Maximum Entropy Method," submitted for publication.

[Par] Parzen, E.,"Multiple Time Series Modeling," in *Multivariate Analysis 2*, ed. by P. R. Krishnaiah, pp.389-409, Academic: New York, 1969.

[Rob] Robinson, E. A.,*Multichannel Time Series Analysis with Digital Computer Programs* San Francisco:Holden-Day, 1967.

[SKM] Sidhu, G. S., T. Kailath and M. Morf, "Development of Fast Algorithms via Innovations Decompositions," Proc. of 7th Hawii Intl. Conf. on Inf. and System Sciences, Honolulu, Hawaii, pp 192-195, Jan. 1974.

[Si] Silverman, L., "Realization of Linear Dynamical Systems," IEEE Trans. Automat. Contr., vol. AC-16, 1971, pp. 554-567.

[SLG] Soderstrom, T., L.Ljung, and I. Gustavsson, "A Comparative Study of Recursive Identification," Rep. 7427, Div. of Auto. Contr., Lund Inst. of Tech., 1974.

[SV] Srinath, M.S.,M. M. Viswanathan,"Sequential Algorithm for Identification of Parameters of an Autoregressive Process," IEEE Trans. Aut. Contr.,AC-20,1975,pp.542-546.

[Sze] Szegö,G.,"Orthogonal polynomials," Amer. Math. Soc. Colloq. Publ., vol. 23,1939; 2nd ed., 1958; 3rd ed. 1967.

[UB] Ulrych, T. J., T. N. Bisshop,"Maximum Entropy Spectral Analysis and Autoregressive Decomposition,"Rev. Geophys. Space Phys.,v.13, 1975,pp.183-200.

[Wak] Wakita, H., "Estimation of the Vocal Tract Shape by Optimal Inverse Filtering and Acoustic/Articulatory Conversion Methods," Monograph No.9, Speech Communications Res. Lab. Inc., Santa Barbara, Calif., July 1972.

[Wie] Wiener N., *Time Series*,M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1949.

[WR] Wiggins, R.A.and E.A. Robinson,"Recursive Solution to the Multichannel Filtering Problem," J. Geophys. Res., vol. 70,1965, pp.1885-1891.

[YT] Youla, D.C. and P. Tissi, "N-Port Synthesis via Reactance Extraction Part I," IEEE Int. Convention Record, vol. 14, pt. 7, 1966, pp. 183-205.

Ladder Forms in Estimation and System Identification

M. Morf

Information Systems Laboratory
Stanford University, Stanford, CA 94305

Abstract

Ladder forms are probably the most promising canonical forms in estimation, and system identification. Many recent applications, such as in geophysical signal processing, high resolution ("maximum entropy") spectral estimation and speech encoding, justify the interest in these forms. They appear in many contexts, such as scattering and network theory and the theory of orthogonal polynomials. The state-space model ladder realizations are very closely related to (block) Schwarz matrix canonical forms, which generally appear in the context of stability analysis. In fact they are the natural "stability canonical form" for (discrete-time) Lyapunov equations since the associated positive definite (covariance) matrices are diagonal resp. an Identity. This fact leads also to close connections to square-root algorithms including the ones of Cholesky and Chandrasekhar type, since again ladder forms are the natural canonical forms. In realization theory these forms are obtained via orthonormal state-space bases using Gram-Schmidt type procedures. Ladder forms have many other advantages, such as lowest computational complexity, good numerical behavior, stability "by inspection" properties and relations to physical properties such as reflection or partial correlation coefficients, and perhaps absorption coefficients.

We shall present an outline of some newer results connecting these topics and present new examples of our new exact least-squares recursions for ("adaptive") ladder forms with poles and zeros. We close with a few simulation examples, including the identification of a layered media (via ultra-sound).

I. Introduction

Ladder forms have attracted much attention recently because they are probably the most promising canonical forms in estimation and system identification. These forms have appeared in many applications such as geophysical signal processing for quite some time , and more recently such models are being used in high resolution ("maximum entropy") spectral estimation and speech encoding. Ladder (sometimes called lattice - a term we would like to reserve for two and higher dimensional extensions [LKM]) forms appear in many contexts, first perhaps in scattering and network theory where the scattering of waves in layered media or in (non-homogeneous) transmission lines leads very naturally to ladder forms, see e.g. [Cla2], [LKF], [RMY], [Kelly].

Ladder forms appear explicitly but more often implicitly in many contexts. They are directly related to the scattering of waves and therefore perhaps first introduced in physics. Some of the associated mathematics are used in network theory, where the cascade structure of the ladder forms plays an important role. The notion of transfer functions leads very naturally to the next connection, the theory of orthogonal polynomials. They in turn also appear in the stability analysis of linear systems. The state-space models that are related to orthogonal (matrix) polynomials are the so-called (block) Schwarz matrix canonical forms, see e.g. [AJM], [SS]. However, the special structure of these matrices leads very directly to the ladder realizations [Mo]. In fact they are the natural "stability canonical form" for (discrete-time) Lyapunov equations, since the associated positive definite (covariance) matrices are diagonal or respectively an identity matrix. The similarity transformations to this form involve a matrix square-root of the associated covariance matrix. The ladder forms are therefore closely connected to square-root algorithms including the ones of Cholesky and Chandrasekhar type. In realization theory these forms are obtained via orthonormal state-space bases using Gram-Schmidt type procedures, due to the fact that this ortho-normalization is again related to matrix square-root and orthogonal polynomials.

Ladder forms have many other interesting properties. Due to the fact that they are in many problems the "natural canonical form", they lead to algorithms with lowest computational complexity compared to other canonical forms. Although a detailed study is still outstanding, there are many indications that this form leads to good numerical behavior of the associated algorithms, a property that is not shared with most canonical forms. Furthermore, the stability "by inspection" property given the ladder coefficients is shared only by the Jordan or modal canonical form. However, the latter one requires the knowledge of the eigen-values that are in general not very easily obtained, compared to the finite algorithm required to get the ladder coefficients. They in turn have other interesting interpretations and relations to physical properties such as reflection, and perhaps absorption coefficients. In stochastic process modeling and spectral estimation the ladder coefficients turn out to be partial correlation or canonical correlation coefficients, which leads to very simple methods to determine these parameters either from covariance data or even directly from measured data.

In [MLNV] we presented a classification of exact least-squares modeling methods. The material discussed here is a sequel to the results discussed there, in particular we will concentrate here on the ladder forms and the associated algorithms. We shall present an outline of some newer results connecting these topics and present new examples of our new exact least-squares recursions for ("adaptive") ladder forms with poles and zeros. We close with a few simulation examples, including the identification of a layered media (via ultra-sound).

II. Ladder Realizations

In [MLNV] and [MVL] we discussed various ladder realizations. We assume here familiarity with this material and would like to give here only a missing link to state space realizations, namely the fact that the ladder forms can be obtained via an ortho-normalization of the state space. In this context it is well known, that various canonical state space realization can be obtained via methods that construct a basis of either the Hankel matrix of the Markov parameters, resp. the impulse response parameters of the system, or bases of the controllability or observability matrices of the system, see e.g. [K-S74]. We will present here an outline of the scalar discrete-time constant parameter case. For convenience we use an intermediate canonical form, the controller form. It has the property that the i^{th} component of the n state vector $x^i(t)$ can be obtained from the

* This work was supported by the Defense Communications Agency under contract DCA100-77-C-0005; the National Science Foundation under Contract NSF Eng75-18952; the Air Force Office of Scientific Research, Air Force Systems Command under Contract AF44-620-74-C-0068; the Joint Services Electronics Program under Contract N00014-75-C-0601; the Institute of Biomedical Technology in Zurich Switzerland; and by ARPA through the use of the Stanford Artificial Intelligence Laboratory facilities.

input $u(z)$ in Z-transform notation via $x^i(z) = u(z)z^{n-i}/a(z)$, where $a(z)$ is the characteristic polynomial. The ladder forms are obtained in a similar manner via $x_l^i(z) = u(z)b^i(z)/a(z)$, where $b^i(z)$ is the i^{th} (dual) ortho-normal polynomial on the unit circle [Sze]. Writing these facts in matrix notation, we obtain the results that the the state at time n is given by $x_n = C_n\, u[n-1,0]$, $u[i,j] = [u_i, \ldots, u_j]$, where C_n is the usual controllability matrix. For the controller form it is now not too surprising that $C_n^c = UT(a_1,\ldots,a_n)^{-1}$, the inverse of a unit upper triangular Toeplitz matrix of the coefficients of $a(z)$. the Ladder forms on the other hand result in (see [Mo]) $C_n^l = B_n\, UT(a_1,\ldots,a_n)^{-1}$, where B_n is a lower triangular matrix containing as rows the coefficient vectors of $b^i(z)$. Due to its orthogonality property $B_n B_n' = R_n$, the n by n Toeplitz matrix associated with the (Z-transformed) correlation function $R(z) = 1/|a(z)|^2$. R_n is also the steady state covariance matrix of the controller form R_n^c, if $u(z)$ the input is a white process. Now, since the similarity transform matrix S from one state space form to an other is given for instance by the ratio of the controllability matrices, it is clear that $S = C^l (C^c)^{-1} = B_n = (R_n^c)^{-1/2}$ i.e. from an arbitrary (controllable) state space form the similarity transform is given by the inverse square-root of the steady state covariance. This leads finally to the connection with Lyapunov equation type characterization of the ladder forms, namely that their covariances are an Identity (or diagonal) which is precisely the characteristic property of Schwarz matrizes, see [AJM], the state space feedback matrix of ladder forms [Mo]. Due to limitations we defer a more detailed discussion of the details and various extensions to [ML].

III. LS-Recursions for Ladder Forms

The Prewindowing Case

In [MLVK] we presented this case, for completeness and in order to correct some typographical errors we repeat some of the equations here. Given a series of observations $\{y(t), 0 \le t \le T\}$, where $\{y(\cdot)\}$ can be m vectors, we wish to find the least-squares one-step predictor of order p parametrized by the (matrix) coefficients $\{A_{p,T}(i), i=1,\ldots,p\}$. We can define many different squared error criteria $E_{p,T}$, for instance as a function of s and f in

$$E_{p,T} \mathrel{\hat{=}} \sum_{t=s}^{f} \epsilon'_{p,T}(t)\,\epsilon_{p,T}(t) \;,\quad \epsilon_{p,t} \mathrel{\hat{=}} A'_{p,T}\, y[t:t-p] \,,$$

$$A'_{p,T} \mathrel{\hat{=}} [\, I, A'_{p,T}(1), \ldots, A'_{p,T}(p)] ,\quad y'[t:t-p] \mathrel{\hat{=}} [y_t', \ldots, y_{t-p}'] \quad \text{(III-1)}$$

An obvious choice from an *innovations* point-of-view is $(s=0, f=T)$, the "pre-windowing" case [MDKV]. If $s = p$ and $f = T$ the so-called "covariance" method is obtained [MDKV], [MVL], and if $s = 0$ and $f = T + p$ we get the pre- and post-windowed case or the "correlation" method [MC]. The total squared error can be expressed as

$$E_{p,T} = tr\{A'_{p,T} R_{p,T} A_{p,T}\} \;,\quad R_{p,T} = Y_{p,T} Y'_{p,T} \,,$$
$$Y_{p,T} \mathrel{\hat{=}} [\, y[0:-p], y[1:-p+1], \ldots, y[T:T-p]\,] \qquad \text{(III-2)}$$

Thus the problem of determining $A_{p,T}$ by minimizing $E_{p,T}$ leads to

$$R_{p,T} A_{p,T} = [R^{\epsilon}_{p,T}, 0, \ldots, 0]' ,\quad tr\, R^{\epsilon}_{p,T} = \min E_{p,T} \quad \text{(III-3)}$$

Although $R_{p,T}$ is not Töplitz, it still carries a certain shift-invariance structure, given by the following identities

$$R_{p,T} = R_{p,T-1} + y[T:T-p]\; y[T:T-p]' \qquad \text{(III-4)}$$

$$= \begin{bmatrix} x & x & x \\ x & \multicolumn{2}{c}{R_{p-1,T-1}} \end{bmatrix} = \begin{bmatrix} R_{p-1,T} & & x \\ x & x & x \end{bmatrix} . \qquad \text{(III-5)}$$

Define the backward predictor $B_{p,T}$ and the smoothing errors $C_{p,T}$

$$B'_{p,T} R_{p,T} \mathrel{\hat{=}} [\,0, \ldots, 0, R^{r}_{p,T}] ;\quad C'_{p,T} R_{p,T} \mathrel{\hat{=}} y[T:T-p] \quad \text{(III-6)}$$

Then the forward and backward prediction errors (innovations), $\epsilon_{p,T}$, and $r_{p,T}$, and an auxiliary scalar $\gamma_{p,T}$ can be defined by

$$[\,\epsilon'_{p,T}, r'_{p,T}, \gamma_{p,T}\,] \mathrel{\hat{=}} y'[T:T-p]\; [\,A_{p,T}, B_{p,T}, C_{p,T}\,]$$

Order Update Recursions

Using the three shift-invariance identities for $R_{p,T}$ (III-5) and using some symmetry properties, the order update recursions for $A_{p,T}$, $B_{p,T}$, $C_{p,T}$, $R^{\epsilon}_{p,T}$, and $R^{r}_{p,T}$ are

$$A'_{p+1,T} = [\,A_{p,T}, 0\,]' - \Delta'_{p+1,T} R^{-r}_{p,T-1} [\,0, B'_{p,T-1}\,]'$$
$$B'_{p+1,T} = [\,0, B_{p,T-1}\,]' - \Delta_{p+1,T} R^{-\epsilon}_{p,T} [\,A'_{p,T}, 0\,]^{T} \qquad \text{(III-7)}$$
$$C'_{p+1,T} = [\,C'_{p,T}, 0\,]' + r'_{p+1,T} R^{-r}_{p+1,T} B'_{p+1,T} \quad \text{where}$$
$$\Delta_{p+1,T} = [\,\textit{last block row of } R_{p+1,T}\,]\,[\,A'_{p,T}, 0\,]'$$
$$= [\,0, B'_{p,T-1}\,]\,[\,\textit{first block row of } R_{p+1,T}\,]' .$$

$$R^{\epsilon}_{p+1,T} = R^{\epsilon}_{p,T} - \Delta'_{p+1,T} R^{-r}_{p,T-1} \Delta_{p+1,T}$$

$$R^{r}_{p+1,T} = R^{r}_{p,T} - \Delta_{p+1,T} R^{-\epsilon}_{p,T} \Delta'_{p+1,T} , \qquad \text{(III-8)}$$

The order update recursions are very similar to the multivariate version of the Levinson algorithm, and a similar set of recursions for time-update can also be obtained [MDKV],[Mc].

Ladder Type Realization

Premultiplying the above equations by $y[T:T-p+1]$, we obtain the following order update recursions for $\epsilon_{p,T}$, $r_{p,T}$, $\gamma_{p,T}$

$$\epsilon_{p+1,T} = \epsilon_{p,T} - \Delta'_{p+1,T} R^{-r}_{p,T-1} r_{p,T-1}$$
$$r_{p+1,T} = r_{p,T-1} - \Delta_{p+1,T} R^{-\epsilon}_{p,T} \epsilon_{p,T}$$
$$\gamma_{p+1,T} = \gamma_{p,T} + r'_{p+1,T} R^{-r}_{p+1,T} r_{p+1,T} . \qquad \text{(III-9)}$$

The "*Kalman gain*" $\Delta_{p+1,T}$ is obtained from (III-5),(III-7) (cf.[MV]) via

$$\Delta_{p+1,T+1} = \Delta_{p+1,T} + r_{p,T}\,\epsilon'_{p,T+1} \,/\, (1 - \gamma_{p-1,T}) \quad \text{(III-10)}$$

and the *reflection* or *PARCOR coefficients* are obtained by

$$K^{\epsilon}_{i+1,T} \mathrel{\hat{=}} \Delta_{i+1,T} R^{-\epsilon}_{i,T} ;\quad K^{r}_{i+1,T} \mathrel{\hat{=}} \Delta'_{i+1,T} R^{-r}_{i,T} \quad \text{(III-11)}$$

The initial conditions are given by

$$\epsilon_{0,T} = r_{0,T} = y_T ;\qquad \gamma_{-1,T} = 0 ;$$
$$R^{\epsilon}_{0,T} = R^{r}_{0,T} = \sum_{t=0}^{T} y_t\, y_t' = R^{\epsilon}_{0,T-1} + y_T\, y'_T ;$$

for $p \ge T$:

$$\epsilon_{p,T} = \epsilon_{T,T} ;\quad r_{p,T} = r_{T,T} ;\quad \gamma_{p,T} = \gamma_{T,T} ;$$
$$R^{\epsilon}_{p,T} = R^{\epsilon}_{T,T} ;\quad R^{r}_{p,T} = R^{r}_{T,T} ;\quad \Delta_{p+1,T} = 0 ;$$
$$\Delta_{p+1,p+1} = y_0\, \epsilon_{p,p+1}' .$$

As the dual to the stochastic forms in [IS], [Wak], [Mo], [SKM], equations (III-8)-(III-11) are a complete set of *order and time update recursions* to obtain the *exact least-squares* ladder form predictor, which is shown in Figure 1.

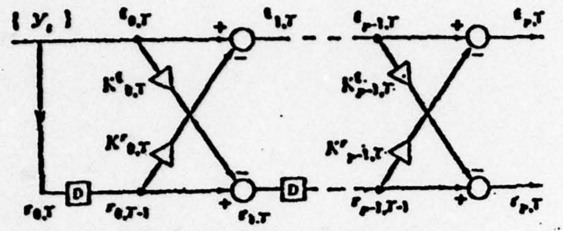


Figure 1. *Ladder realization of exact one-step least-squares predictor.*

The recursion (III-10) computes the *sample cross-covariance* of the *forward and backward innovations*, using the *optimal weighting* $1/(1-\gamma_{..})$, compared to other suboptimal schemes [SV]. See in the appendix a sample comparison of the exact versus two approximate methods. In the scalar case $R_{p,T}>0$ if $y_0=0$, or in general if $\gamma_{p-1,T}<1$, since $0\leq\gamma_{p,T}\leq 1$ [MDKV], [MVL]. If $m>1$, we require $T\geq p+m$. These singularities can be avoided by including a priori estimates of the covariance R_n, or equivalently including a weighted norm of the predictor a_n in the error criteria $E_{p,T}$. For tracking of time-varying parameters, e.g. in speech modeling methods, these equations can be modified, either by puting an exponential weight on past errors as discussed in [MKL], (implemented in the simulation in the appendix). Alternatively, the lower bound of the error criterion in (III-1) can be increased, e.g. $s = T - f$, where f is the (constant) "sliding" time frame width of the analysis. This corresponds also to a sliding window on the prediction errors. The resulting equations are similar to the ones in [MVL].

Instead of computing the scalars γ one can also work with a second set of prediction errors based on the "old" parameter estimates, since

$$\epsilon_{p,T}(T+1) = \epsilon_{p,T+1} \,/\, (1-\gamma_{p-1,T}) \,.$$

This alternate form was also found by J. Baker, IBM Yorktown (private communication). A similar situation occurs in the Fast Cholesky (least-squares) algorithms for estimating moving-average parameters via feedback filters described in [Mo], where a "second filter" or "predictor filter" appears that computes variables of the type $\epsilon_{p,T}(T+1)$. It is interesting to note in this context, that the unwindowed ("covariance") method actually also leads to ***signal feedback*** paths (actually a smoothing filter), see [MVL], but the simpler prewindowing case is feed forward only.

Many modifications have been proven useful in actual implementations, they are partially due to the fact that many additional identities exist and others are due to differences in numerical behavior and trade-offs in operations count and memory requirements. Systematic experiments are now in progress and will be reported on shortly.

IV. LS-Recursions for Rational Ladder Forms

Rational or ARMA Modeling

Rational or pole-zero or ARMA modeling methods were described in [SLC], [MKL] and their relation to joint innovations representation via an imbedding of the ARMA model in a two (m) channel AR model in [MLNV] and [Mo]. The same idea also leads to stable partial minimal realizations of the joint impulse-response and covariance-matching type [MLNV]. Given an ARMA model as represented by the difference equation we can rewrite it as

$$y_t + a_1 y_{t-1} + \dots a_n y_{t-n} - b_1 u_{t-1} - \dots b_n u_{t-n} = b_0 u_t\,, \qquad \text{(IV-1)}$$

or $\quad \mathbf{a}'y_t - b_1'u_t = b_0 u_t\,,\quad$ where

$$\mathbf{a}' = [\,1\,, a_1\,, \dots, a_n\,],\quad y_t' = [\,y_t\,, \dots, y_{t-n}\,],$$
$$b_1' = [\,0\,, b_1\,, \dots, b_n\,],\quad u_t' = [\,u_t\,, \dots, y_{t-n}\,]\,.$$

Now consider the following augmented equation

$$\begin{bmatrix} \mathbf{a}' & -\mathbf{b}_1' \\ 0 & e_1' \end{bmatrix} \begin{bmatrix} y_t \\ u_t \end{bmatrix} = \begin{bmatrix} b_0 u_t \\ u_t \end{bmatrix}, \qquad \text{(IV-2)}$$

(e_1' is the first unit vector). This equation can be interpreted as an AR model for the joint process $\{\,y, u\,\}$ [Mo], since the RHS is equal to the joint innovations of $\{\,y, u\,\}$, since

$$\epsilon_t = \begin{bmatrix} \epsilon^{y}_t \\ \epsilon^{u}_t \end{bmatrix} = \begin{bmatrix} y_t - \hat{y}_{t|t-1} \\ u_t - \hat{u}_{t|t-1} \end{bmatrix} = \begin{bmatrix} b_0 u_t \\ u_t \end{bmatrix}. \qquad \text{(IV-3)}$$

Stochastic Case

From a stochastic process point of view we can express the normal equation associated with the augmented AR model as

$$E\,\left\{\begin{bmatrix} y_t \\ u_t \end{bmatrix}[\,y_t'\;u_t'\,]\right\}\begin{bmatrix} \mathbf{a} & 0 \\ -b_1 & e_1 \end{bmatrix} = E\,\left\{\begin{bmatrix} y_t \\ u_t \end{bmatrix}[\,u_t b_0\; u_t\,]\right\}$$

$$= \begin{bmatrix} R_n & T_n' \\ T_n & I_n \end{bmatrix}\begin{bmatrix} \mathbf{a} & 0 \\ -b_1 & e_1 \end{bmatrix} = \begin{bmatrix} e_1 b_0 b_0' & e_1 b_0 \\ e_1 b_0 & e_1 \end{bmatrix}. \qquad \text{(IV-4)}$$

We can solve for the normal equation of $\mathbf{a}_n$:

$$\mathcal{R}_n \mathbf{a}_n = [\,R_n - T_n' T_n\,]\,\mathbf{a}_n = [\,\mathcal{H}_\infty' \mathcal{H}_\infty\,]\,\mathbf{a}_n = e_1 R^{\epsilon}_n \qquad \text{(IV-5)}$$

The equations (IV-4) and therefore the non-Töplitz equations (IV-5) (!) can be solved recursively with the LWR algorithm. Note that if $R^{\epsilon}_k = 0$, the minimal order $n = k$. We could bring equations (IV-4) into a more familiar form by the interleaving permutation $(1,3,5,\dots,2n-1,2,4,6,\dots,2n+2)$, cf. [MDKV], to convert the two-process covariance matrix into a n by n block Töplitz matrix, with 2 by 2 blocks, however the LWR algorithm clearly applies to both representations with suitable modifications.

Thus we have shown that the joint impulse response & covariance matching problem is equivalent to solving a set of normal equations associated with a two channel AR modeling problem. Since the predictor for the joint process is ***triangular*** and ***minimum phase***, the denominator $\mathbf{a}_n$ of the underlying ARMA model is also ***minimum phase*** and therefore ***stable***, (for all k).

Equations (IV-4) and the elegant stability proof were actually first obtained by Claerbout [Cla1] via a least-squares rational approximation. The connections between the joint innovations representation, the augmented normal equations, and the Hankel matrix were pointed out in [Mo] and also in [MDKV], [MKD], [DKM], where algorithms were given to solve equations of the type seen in (IV-4) and (IV-5).

Deterministic Case

In [MLNV] we considered the deterministic case where we are given impulse response data or the Markov parameters, here we shall assume that we are given a series of observations and we want to find a least-squares (deterministic) one-step ARMA predictor recursively from the data equivalent to the RML algorithms described in [SLC] and [MKL]. Our approach will not give a new way how to derive these algorithms, but it will also give us very quickly the ladder forms.

Writing the input/output relationship in matrix notation yields

$$\begin{bmatrix} T_{n,T} & 0 \\ H_T & T_T \end{bmatrix} \begin{bmatrix} a_{n,T} \\ 0_T \end{bmatrix} = \begin{bmatrix} b_{n,T} \\ 0_T \end{bmatrix}, \qquad \text{(IV-5)}$$

where $T_{n,T}$ lower-triangular and H_T is a full matrix, but both are a product of two Töplitz matrices containing the data and the normalized one-step prediction errors, which take place of the inputs $u = R^{-\epsilon/2}\,\epsilon$, $\epsilon = y_t - \hat{y}_{t|t-1}$, where $R^{\epsilon}{}_{n,T} = E_{n,T}$ in

$$\epsilon_t = \begin{bmatrix} \epsilon^{y}{}_t \\ \epsilon^{u}{}_t \end{bmatrix} = \begin{bmatrix} y_t - \hat{y}_{t|t-1} \\ u_t - \hat{u}_{t|t-1} \end{bmatrix} = \begin{bmatrix} b_0 u_t \\ u_t \end{bmatrix}.$$

$$\begin{bmatrix} T_{n,T}' & H_T' \\ I & 0 \end{bmatrix} \begin{bmatrix} T_{n,T} & I \\ H_T & 0 \end{bmatrix} \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} =$$

$$= \begin{bmatrix} R_n & T_{n,T}' \\ T_{n,T} & I \end{bmatrix} \begin{bmatrix} a & 0 \\ -b_1 & e_1 \end{bmatrix} = \begin{bmatrix} e_1 R^{\epsilon}{}_{n,T} & e_1 b_0 \\ e_1 b_0 & e_1 \end{bmatrix}. \qquad \text{(IV-6)}$$

Recursions for Rational Ladder Forms

This partitioning leads easily to the rational ladder recursions. Formally the same recursions can be used. However, the fact that the forward predictor is triangular simplifies and actually makes the recursions possible at all, since the one-step prediction errors are not needed until the next iteration, i.e. they are *feed back* and treated at the next time step as the ("other half" of the) observations. It is easily verified in the same context that half of the entries in $\Delta_{n,T}$ are zero, which guarantees that the one-step prediction errors are not used before they are required in the recursion. The following Figure 2. clearly shows this.

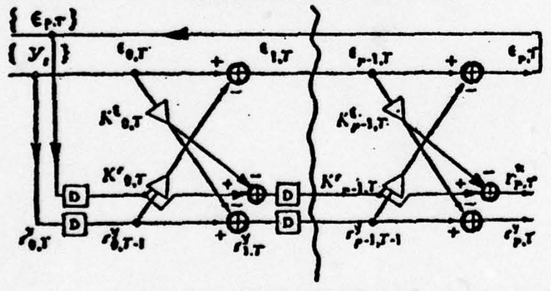


Figure 2. *Rational Ladder realization of exact one-step least-squares predictor.*

Appendix: Computer Simulations

Layered Media Identification

The modeling on of layeded media is of interest in many areas, notably in Geophysics, see e.g. Claerbout [Cla1,2] and more recently in medical imaging or nondestructive testing. There are two basic situations that occur in these areas. The first one, where the source is on the opposite side of the receiver is the straight forward case, it leads to autoregressive or all-pole models, wich can be readily identified by using the various methods to estimate reflection coefficients by cross-correlation of the forward and backward residuals in the whitening filter in ladder form, see e.g. [Cla1,2]. The second case, where the source and receiver are on the same side did up to now not lead to such simple processing as the first case, because the (imput) transfer function is rational in general, or in the best case where a total reflection occurs within some layer the transfer function is an all-pass network. In this case the zeros are equal to the reflected poles and the 'reflection coefficients' of the numerator polynomial are the negative of the ones of the denominator polynomial of the transfer function. We can readily see then, that our rational ladder form specializes and we get only one set of reflection coefficients that can be associated with the ones of the layered medium that generated the data. This particular case is treated from a circuit point of view by Kung [Kun]. Figure 3 shows an example using real ultrasound returns and the estimates of the reflection coefficients using a ladder structure.

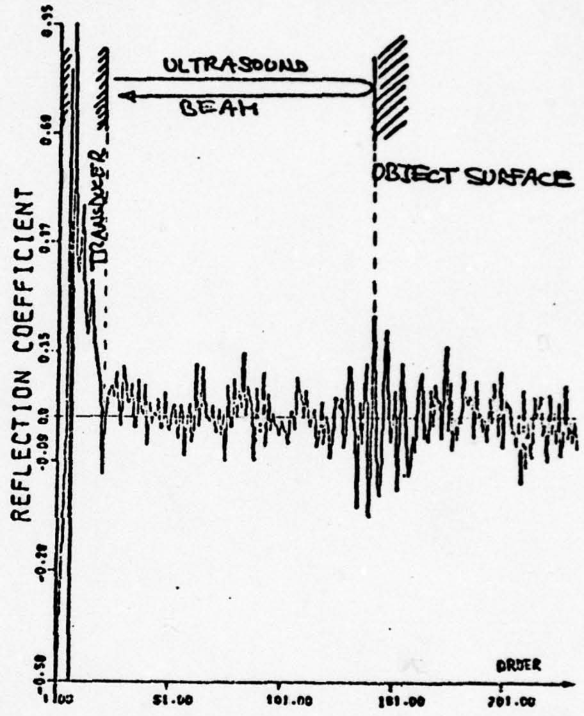


Figure 3: *Identification of a layered media via ultrasound.*

The experiments were performed by Linda Joint and Dough Boyd in the Stanford Electronics Laboratory of Prof. J. Meindl. The reflection coefficient estimates appear to be much smaller than anticipated from the experimental set up, this is due to several factors: The ladder structure actually identifies not only the medium and the single reflecting plate in the path of the ultrasound beam, but also computes an equivalent layer model for the transducer. The estimated values of the first large reflection coefficients show, that the transducer is very inefficient and not very well matched because the largest value is very close to one, which tend to "turn off" all higher order reflection coefficients. Further more, because of the wavelengs used the layers of the medium have a continuous reflection coefficient density which indicates that this direct scheme must fail since we tried to estimate the derivative of a function (with noisy data!) It would require the use of a modified ladder form that is parameterized by the equivalent of the "area function" used for instance in the speech modeling context [Wak].

Sample Comparison of Different Reflection Coefficient Estimates

Ladder coefficient estimates were in the past said to converge very slowly, indeed this is the case for approximate recursive methods as demonstrated in the Figure 4, where three methods are compared. Two approximate recursive methods using arithmetic mean definitions of the prediction error (see e.g. [Mak], [MLVK]) and and other computationaly attractive method using the average of the product of the signs of the forward and backward predictior error, which arises from L1 norm considerations see Claerbout[Cla3] and has often been used in circuit design.

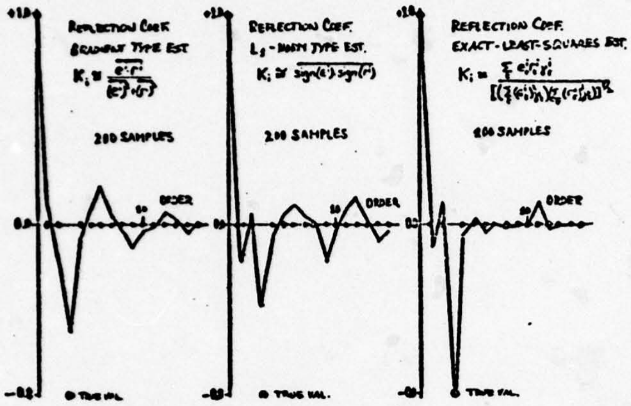


Figure 4: Comparison of Two Approximate and One Exact Recursive Method.

The third method uses our recursive exact least-squares equations (pre-windowed case). The first two schemes give very similar results, i.e. a bias of 50% on the only nonzero third reflection coefficient (-.8) and sidelobes as high as 20% and 10% typical value, whereas our exact method has virtualy no bias and half the maximal and typical sidelobe values. Furthermore, they actually converged already after around 30 samples compared with the 200 samples used in Figure 4 . We may note that the other schemes took much longer to actually converge.

References

Ce[AJM] Anderson, B.D.O., E. Jury, M. Mansour, "Schwarz Matrix Properties for Continuous- and Discrete-Time Systems," Int. Journal of Contr., vol. 23, no. 1, Jan. 1976, pp.1-16.

[Cla1] Claerbout, J.F., "Estimation of a Rational Function on the Unit Circle," int. memo, Stanford Univ., 1969.

[Cla2] Claerbout, J. F., *Fundamentals of Geophysical Data Processing*, McGraw-Hill: New York, 1976.

[Cla3] Claerbout, J. F., Stanford Exploration Project, September 1976 Progress Report, Sep vol. 10, 1976.

[DKM] Dickinson, B., T. Kailath and M. Morf, "Canonical Matrix Fraction and State-Space Descriptions for Deterministic and Stochastic Linear Systems," IEEE Transactions of Automatic Control, vol. AC-19, 1974, pp. 656-667.

[DMK] Dickinson, B., M. Morf and T. Kailath, "A Minimal Realization Algorithm for Matrix Sequences," IEEE Transactions on Automatic Control, vol. AC-19, 1974, pp. 31-38.

[FMKL] Friendlander, B., M. Morf, T. Kailath, L. Ljung, "New Inversion Formulas for Matrices Classified in Terms of Their Distance from Toeplitz Matrices," subm. to SIAM J. A. Math.

[GS] Grenander, U., and G. Szegö, *Toeplitz Forms and Their Applications.* Berkeley, Calif.:Univ. California Press,1958.

[Hou] Householder, A. S., *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York: 1964.

[IS] Itakura,F., and S. Saito,"Digital Filtering Techniques for Speech Analysis and Synthesis," Conf. Rec., 7th Int. Congr. Acoust.,Budapest,1971,Paper 25 C 1.

[K-S74] Kailath, T., Survey, IEEE Trans.Inform.Theory, vol.IT-20, 1974, pp.146-180.

[KVM] Kailath, T., A. Vieira and M. Morf, "Inverses of Toeplitz Operatiors, Innovations, and Orthogonal Polynomials," submitted for publication.

[Kun] Kung, S.Y., "On-Line Identification of Layered Media," Assilomar Conf., December 1977.

[Lanc] Lanczos, C.,"An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," J.Res.Nat. Bur.Standards, vol.45,1950,pp.255-282.

[Lev] Levinson, N.,"The Wiener RMS (root mean square) Error Criterion in Filter Design and Prediction," J. Math. Phys.,vol.25,1947, pp.261-278.

[LKF] Ljung, L., T. Kailath, B. Friedlander, Proc. IEEE, vol.63, 1976, pp.131-139.

[LKM] Lévy,B., S.Y.Kung, M.Morf, "New Results in 2-D Systems Theory, 2-D State-Space Models - Realization and the Notions of Controllability, Observability and Minimality," Monterey Conference on Multidimensional Systems, November 1976.

[Makh] Makhoul, J. "Linear Prediction: A Tutorial Review," Proc. IEEE, vol. 63, 1975, pp. 561-580.

[MDKV] Morf, M., B. Dickinson, T. Kailath and A. Vieira, "Efficient Solution of Covariance Equations for Linear Prediction," submitted to IEEE-ASSP 1976.

[MG] Markel, J.D., and A.H. Gray, Jr.,*Linear Prediction of Speech*, Springer-Verlag, Berlin: 1976.

[MK] Morf, M., and T. Kailath, "Square-Root Algorithms for Least-Squares Estimation ,"IEEE Trans.on Automat. Control,vol. AC-20, 1975, pp. 487-497.

[ML] Morf, M., D.T. Lee, "Pole-Zero Ladder Forms for Least-Squares Estimation," submitted to ASSP, 1977.

[MKL] Morf, M., T. Kailath and L. Ljung, "Fast Algorithms for Recursive Identification," Proc. IEEE Conf. on Decision and Control,pp.916-921,Dec. 1976.

[MLVK] Morf, M., D.T. Lee, A. Vieira and T. Kailath, "Recursive Multichannel Maximum Entropy Method," Proc. Joint Automatic Control Conference, pp.113-117, San Francisco, June 1977.

[Mo] Morf, M., "Fast Algorithms for Multivariable Systems," Ph.D. dissertation, Stanford University, Stanford, California 1974.

[MVK] Morf, M., A. Vieira and T. Kailath, "Covariance Characterization by Partial Auto-Correlation Matrices", Annals of Statistics, May 1978.

[MVL] Morf, M., A. Vieira and D.T. Lee, "Recursive Ladderforms for Identification and Speech Processing," Proc. IEEE Decision and Control Conference, New Orleans, December 1977.

[Par] Parzen, E.,"Multiple Time Series Modeling," in *Multivariate Analysis* 2, ed. by P. R. Krishnaiah, pp.389-409, Academic: New York, 1969.

[RMY] Rhodes, J.D., P. Marston, D.C. Youla, "Explicit Solutions for the Synthesis of Two-variable Transmission-line Networks," IEEE Trans. on Ccts. and Systems Theory, vol. CT-20, no. 5, Sept. 1973, pp.504-511.

[Rob] Robinson, E. A., *Multichannel Time Series Analysis with Digital Computer Programs*, San Francisco: Holden-Day, 1967.

[SKM] Sidhu, G. S., T. Kailath and M. Morf, "Development of Fast Algorithms via Innovations Decompositions," Proc. of 7th Hawii Intl. Conf. on Inf. and System Sciences, Honolulu, Hawaii, pp 192-195, Jan. 1974.

[Si] Silverman, L., "Realization of Linear Dynamical Systems," IEEE Trans. Automat. Contr., vol. AC-16, 1971, pp. 554-567.

[SLG] Soderstrom, T., L.Ljung, and I. Gustavsson, "A Comparative Study of Recursive Identification," Rep. 7427, Div. of Auto. Contr., Lund Inst. of Tech., 1974.

[SS] Shieh,L.S., S.Sacheti, "A Matrix in the Schwarz Block Form and the Stability of Matrix Polynomials," Proc. 10th Asilomar Conf.on Ccts.and Systems.

[SV] Srinath, M.S.,M. M. Viswanathan,"Sequential Algorithm for Identification of Parameters of an Autoregressive Process," IEEE Trans. Aut. Contr.,AC-20, 1975, pp.542-546.

[Sze] Szegö, C., "Orthogonal polynomials," Amer. Math. Soc. Colloq. Publ., vol. 23,1939; 2nd ed., 1958; 3rd ed. 1967.

[UB] Ulrych, T.J., T.N. Bisshop, "Maximum Entropy Spectral Analysis and Autoregressive Decomposition, "Rev. Geophys. Space Phys., vol.13, 1975, pp.183-200.

[Vie] Vieira, A., Ph.D. Dissertation, Stanford University, December 1977.

[Wak] Wakita, H., "Estimation of the Vocal Tract Shape by Optimal Inverse Filtering and Acoustic/Articulatory Conversion Methods," Monograph No.9, Speech Communications Res. Lab. Inc., Santa Barbara, Calif., July 1972.

[Wie] Wiener N., *Time Series*,M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1949.

[WR] Wiggins, R.A.and E.A. Robinson,"Recursive Solution to the Multichannel Filtering Problem," J. Geophys. Res., vol. 70,1965, pp.1885-1891.

APPENDIX C

```
BEGIN "speech"

          comment
          This is a complete analysis program that compuute
          reflection coefficients, pitch, and energy information
          as each new data is sampled. The analysis is done
          independent of transmission frame size. It resets to
          start state whenever silence or pause is encountered.
          The pitch detection scheme is based on testing the log
          likelihood ratios at each sample and there is no time delay
          in the transmission.

          The main program first calls FILEOPEN to open necessaary files.
          Then it calls LADDER which perform time update operations.
          LADDER calls the following procedures:
                    INITIALIZE for initialization of global variables,
                    TRINIT for initialization of transmission strategy,
                    PITCHINIT for initialization of pitch detector,
                    ORDERUPDATE for update of ladder variables,
                    RESET when silence or bad numerical condition is encountered,
                    PITCHDETECT for pitch detection,
                    TRANSMITTER for transmitting reflection coefficients, pitch,
                              and energy information at each transmission frame
                              boundary,
          FILECLOSE is called at the end of input file;
```

```
require "msailm.sai[exp,lee]" source_file;
require "msailp.sai[exp,lee]" load_module;

#  Compiletime definitions of speech analysis constants;

define MAXP = 30;              # max order supported;
define MAXPP = 6;              # number of pitch pulses/ frame;

#  Compiletimme definitions of I/O buffer sizes, the values used
#  here are important only from efficiency considerations;

define BUFSIZE = 4096;         # number of speech samples in core buffers;
define OBUFSIZE = 128;         # in core transmission values;
define RBSIZE = 128;           # history registers;
define BUGBUF = 4096;          # buffer for debug file;

#  Housekeeping variables;

integer fpy;                   # pointers for input files;
integer fko, fpo, feo;         # pointers for output files;
integer fbug1, fbug2, fbug3,   # pointers for debug data files;
        fbug4, bt;

integer Fsample, Nsamples;     # first sample and total number of speech samples;
integer i, j;                  # general purpose loop counters;

#  Analysis ladder form parameters and data storage;

integer t,st,tt,tminpmax,lst;  # time indices;
integer tau;                   # time constant for weighting;
integer p, pmax;               # parameter order counters;
real ttau;
real delta;                    # prior value of covariance;
real tI, t1I, t0t1, t1t0;      # 1/t, 1/(t+1), t/(t+1), (t+1)/t ;
real resetsup,resetinf;        # upper and lower reset threshold;
real array                     # variables of analysis ladder;
        e,   eZ,  r,   rZ,
        D,   K,   Ke,  Kr,
        Re,  Rr,  RrZ [0 tc MAXP];
real array g, gZ[-1 tc MAXP];
real yt;                       # current input;

#  Data buffering considerations:
#  Data buffer management is handled independently from the
#  analysis.  The particular sizes of the following buffers are only
#  important in that particular values allow efficient operation
#  of the SU-AI disk system;

real array y[0 tc BUFSIZE-1];  # input data buffer;

# output buffers for transmission parameters;
# the first index changes per frame, the second index represents the
#  number of parameters of that type per frame;

real array                     # reflection coefficients;
        okb[0 tc OBUFSIZE,1 tc MAXP];
real array                     # pitch information;
        opitchb[0 tc OBUFSIZE,1 tc MAXPP];
real array                     # energy information;
        oenergyb[0 tc OBUFSIZE];
real array                     # debug data buffers;
        bug1,
        bug2,
        bug3,
        bug4[0 tc BUGBUF];

#  Buffer management variables:
#  In general, whenever a buffer empties or fills up, at that
#  point, an I/O call is made to refill or drain it;

integer obp;                     # output buffer pointer;
integer kbsize, pbsize;          # multiples of OBUFSIZE;

#  Transmision strategy variables;

boolean pp,lr,rs;                # pitchpulse, lower reset, reset;
integer framesize;               # number of samples per transmission;
real Energy;                     # per frame energy of speech samples;
integer array pi[1 tc MAXPP];    # pitch position indicator;
integer ppptr;                   # pointer into pi array;

#  These following variables are ring buffer management varialbes;
#  RBSIZE is ring buffer size, should be the largest pre-deadzone;
#  (The transmission module maintains recent history of the
#  reflection coefficients);

real array
        rkb[0 tc RBSIZE-1,1 tc MAXP];
integer rbp,trp;                 # buffer counters;

# dead zone sizes and related variables for transmission;
#  pre and post deadzones for pitch pulses, lower threshold resets
#  and upper threshold resets;
#  Variables nextXX, nextXXs, otdel, and dzcount reflect implementation
#  details of the deadzone strategy rather than algorithmic details;

integer pppredz,pppostdz,rspredz,rspostdz,lrpredz,lrpostdz;
integer pptotdz,rstotdz,lrtotdz,otdel;
integer nextpp,nextlr,nextrs;
integer nextpps,nextlrs,nextrss;
integer dzcount;


# pitch detector variables;

real dginf, dgmax, taup, rho, oldmaxpt, maxpt, pinf, pt, oldpt, dg, alphap;
integer deadzp, nextp, sgnp, lastptime, ppwindow;
```

```
#  These procedures are largely self-explanatory.  At the beginning
#  of processing, all the required files are opened, at the end
#  they are closed;

procedure fileOpen;
  ⊂ "fileOpen"  # Create files;
        fpy ← open ("speech input file: ", DATA ! INPUT ! PROMPT );
        fko ← open ("kkk.dat", DATA ! CREATE ! OUTPUT );
        fpo ← open ("pit.dat", DATA ! CREATE ! OUTPUT );
        feo ← open ("en.dat",  DATA ! CREATE ! OUTPUT );
        fbug1 ← open ("bug1.dat", DATA ! CREATE ! OUTPUT );
        fbug2 ← open ("bug2.dat", DATA ! CREATE ! OUTPUT );
        fbug3 ← open ("bug3.dat", DATA ! CREATE ! OUTPUT );
        fbug4 ← open ("bug4.dat", DATA ! CREATE ! OUTPUT );
  ⊃ "fileOpen"  ;

procedure fileClose;
 ⊂ "fileClose"
        close(fpy);
        close(fko);
        close(fpo);
        close(feo);
        close(fbug1);
                ttyWrite( "BUG1.DAT contains predetection pitch postition", nl);
        close(fbug2);
                ttyWrite( "BUG2.DAT contains postdetection pitch postition", nl);
        close(fbug3);
                ttyWrite( "BUG3.DAT contains dg", nl);
        close(fbug4);
                ttyWrite( "BUG4.DAT contains innovations", nl);
 ⊃ "fileClose";
```

```
procedure initialize;
  c "initialize"

        #  initialize analysis algorithm variables;

        ttyWrite("Initialization of tracking parameters:", nl);
        ttyWrite("       tau (160) = ");
        read(ttyRead, tau);
        ttyWrite("       Prior covariance (0.00001) = ");
        read(ttyRead, delta);
        ttyWrite("       upper reset threshold on g[pmax] (0.99) = ");
        read(ttyRead, resetsup );
        ttyWrite("       lower reset threshold on g[pmax] (0.0001)= ");
        read(ttyRead, resetinf );
        for i ← 0 upto pmax do
          c     e[i] ← r[i]  ← rZ[i] ← 0;
                D[i] ← Ke[i] ← Kr[i] ← K[i] ← 0;
                Re[i] ← Rr[i] ← RrZ[i] ← 0;
                g[i] ← gZ[i] ← 0;
          ɔ;
        g[-1] ← gZ[-1] ← 0;
        Re[0] ← Rr[0] ← delta ;

        st ← 0;             # set reset pointer to 0;
        lst ← -20;          # printout interval for last reset;

        # initialize data buffering;
        # briefly, the idea is to break up Nsamples into groups;
        # of no more than BUFSIZE, reading and writing is done on;
        # these smaller segments;

        # buffer management, initialize pointers;
        bt ← 0;             # pointer for debug file;
        rbp ← 0;
        obp ← 0;
        trp ← 0;
        kbsize ← OBUFSIZE * MAXP;
        pbsize ← OBUFSIZE * MAXPP;
        for i ← 0 upto RBSIZE-1 do
                for j ← 1 upto MAXP do
                        rkb[i,j] ← 0;
        for i ← 0 upto OBUFSIZE-1 do
          c     for j ← 1 upto MAXP do
                        okb[i,j] ← 0;
                for j ← 1 upto MAXPP do
                        opitchb[i,j] ← -1;
                oenergyb[i] ← 0;
          ɔ;
  ɔ "initialize";
```

```
procedure reset;
  ⊂ "reset"

        #  This procedure is nearly equivilent to the analysis
        #  algorithm parts of the initialize procedure above;

        if (t - lst) geq 20 then
                ⊂       if rs then ttyWrite ("Upper Reset at t = ", t, nl)
                              else ttyWrite ("Lower Reset at t = ", t, nl);
                        lst ← t;
                ⊃;
        st ← -1;
        for i ← 0 upto pmax do
          ⊂     D[i] ← Ke[i] ← Kr[i] ← K[i] ← 0;
                Re[i]← Rr[i] ← RrZ[i] ← 0;
                e[i] ← r[i] ← rZ[i] ← 0;
                g[i] ← gZ[i] ← 0;
          ⊃;
        Re[0] ← Rr[0] ← delta;
  ⊃ "reset";
```

```
procedure trinit;
  ⊂ "trinit"

        # initialize transmission strategy variables;

        Energy ← 0;

        ttyWrite ("Initialization of transmitter parameters:", nl);
        ttyWrite ( "    Transmission framesize = ");
        read (ttyRead, framesize);
        ttyWrite ( "    pp pre dz (3)= ");
        read ( ttyRead, pppredz );
        ttyWrite ( "    pp post dz (20)= ");
        read ( ttyRead, pppostdz );
        ttyWrite ( "    low reset pre dz (0)= ");
        read ( ttyRead, lrpredz );
        ttyWrite ( "    low reset post dz (20)= ");
        read ( ttyRead, lrpostdz );
        ttyWrite ( "    reset pre dz (pmax)= ");
        read ( ttyRead, rspredz );
        ttyWrite ( "    reset post dz (20)= ");
        read ( ttyRead, rspostdz );
        pptotdz ← pppredz + pppostdz;
        lrtotdz ← lrpredz + lrpostdz;
        rstotdz ← rspredz + rspostdz;
        otdel ← pppredz max ( lrpredz max rspredz );
        nextpps ← otdel - pppredz + 1;
        nextlrs ← otdel - lrpredz + 1;
        nextrss ← otdel - rspredz + 1;
  ⊃ "trinit";
```

```
procedure pitchinit;
  ⊂ "pitchinit"

        # initialize pitch detector variables;

        ttyWrite ( "Initialization of pitch detector parameters:",nl);
        ttyWrite ( "    post detect deadzone = ( 30) ");
        read ( ttyRead, deadzp );
        ttyWrite ( "    scanner time constant taup = (50) ");
        read ( ttyRead, taup );
        ttyWrite ( "    scanner upperthreshold factor = (3) ");
        read ( ttyRead, alphap);
        ttyWrite ( "    pre scanner window = ( 10) ");
        read ( ttyRead, ppwindow );
        ttyWrite ( "    scanner lower threshold Pinf = ( 0.003) ");
        read ( ttyRead, pinf );
        ttyWrite ( "    inf for dg (0.01)= " );
        read ( ttyRead, dginf );
        ttyWrite ( "    norm for dgmaxp (0.25)= " );
        read ( ttyRead, dgmax );
        oldpt ← 0;
        oldmaxpt  ← maxpt ← pinf;
        rho ← exp ( -1/taup);
        lastptime ← nextp ← 0;
        for i ← 1 upto MAXPP do pi[i] ← -1;
        ppptr ← 0;

  ⊃ "pitchinit";
```

```
procedure orderUpdate;
  ⊂  "orderUpdate"

        #  See algorithm descriptions for an explanation of this procedure;

        # order updates the innovations;
        for p ← 0 upto ( tminpmax - 1 ) do
          ⊂ "stepupOrder"

                #  Update the partial correlations;

                if tI*gZ[p-1] geq 1. then       # numerical conditioning control;
                  ⊂     ttyWrite("gZ[", p-1, "] >= ", 1./tI, "at t = ", t, NL);
                        gZ[p-1] ← 0.99/tI ;
                  ⊃;

                D[p+1]  ← D[p+1] + t1I*(rZ[p]*e[p]/(1 - tI*gZ[p-1]) - D[p+1]);

                g[p]  ← g[p-1]  +  r[p] * r[p] / Rr[p];
                Ke[p+1] ← D[p+1] / Re[p];
                Kr[p+1] ← t1t0*D[p+1] / RrZ[p];
                K [p+1] ← if ( D[p+1] < 0 )
                                then - sqrt( abs( Ke[p+1]*Kr[p+1]))
                                else   sqrt( abs( Ke[p+1]*Kr[p+1])) ;

                #  Update the prediction errors and their covariances;

                r[p+1] ← rZ[p] - Ke[p+1] * e[p];
                e[p+1] ← e[p] - Kr[p+1] * rZ[p];

                if p = ( st - 1 ) then
                  ⊂     # Order update on initial covariances;

                        Re[p+1] ← Re[p] - Kr[p+1] * D[p+1];
                        Rr[p+1] ← t0t1*RrZ[p] - Ke[p+1]*D[p+1];
                  ⊃
                else
                  ⊂     # Time update on subsequent covariances;

                Re[p+1] ← Re[p+1] + t1I*( e[p+1]*e[p+1]/(1-tI*gZ[p]) - Re[p+1] );
                Rr[p+1] ← Rr[p+1] + t1I*( r[p+1]*r[p+1]/(1-t1I*g[p]) - Rr[p+1] );

                  ⊃;

          ⊃ "stepupOrder";

        # Update g[tminpmax];
        g[tminpmax] ← g[tminpmax-1] + r[tminpmax]*r[tminpmax]/Rr[tminpmax];

  ⊃ "orderUpdate";
```

```
procedure pitchdetect;
   ⊂ "pitchdetect"

        # compute differential likelihood ratio;
        dg ← (tlI*g[tminpmax] - tI*gZ[tminpmax])/dgmax;
        if dg < dginf then dg ← 0;
        if dg neq 0
                 then   # extract nongaussian pulse;
                      ⊂ if abs(eZ[tminpmax]) > abs(e[tminpmax])
                                then pt ← eZ[tminpmax]
                                else pt ← e[tminpmax];
                      ⊃
                 else pt ← 0;
        if pt < 0 then  # invert negative pulse;
                     ⊂  pt ← - pt / 2;
                        sgnp ← -1;
                     ⊃
                  else  sgnp ← 1;
        bug3[bt] ← pt * sgnp;


        # start pitch scanner;
        if t > nextp
          then  # start exponential scanner;
             ⊂  maxpt ← ( maxpt * rho ) max pinf;
                if pt > maxpt then      # hit first pitch pulse;
                          ⊂     # set new scanner threshold;
                                maxpt ← pt min ( alphap * oldmaxpt );
                                oldmaxpt ← maxpt;
                                oldpt ← pt;
                                lastptime ← t;
                                nextp ← t + deadzp;
                                pp ← TRUE;
                          ⊃;
             ⊃
         else   if ( (t - lastptime) leq ppwindow ) and ( pt > 2 * oldpt )
                     then ⊂  # encounter 2nd higher pulse, restart scanner;
                                nextp ← t + deadzp;
                                maxpt ← pt min ( alphap * oldmaxpt );
                                oldmaxpt ← maxpt;
                                oldpt ← pt;
                                # discard and update current pitch location;
                                pi[ppptr max 1] ← t mod FRAMESIZE;
                          ⊃
                     else pp ← FALSE;

        # output debug variables;
        bug4[bt] ← e[tminpmax];
        bug2[bt] ← dg;
        if pp then bug1[bt] ← pt * sgnp  else bug1[bt] ← 0;
        bt ← ( bt + 1 ) mod BUGBUF;
        if bt = 0 then
                ⊂       aryWrite( fbug1, bug1, BUGBUF );
                        aryWrite( fbug2, bug2, BUGBUF );
                        aryWrite( fbug3, bug3, BUGBUF );
                        aryWrite( fbug4, bug4, BUGBUF );
                ⊃;

   ⊃ "pitchdetect";
```

```
procedure transmitter;
  ⊂ "transmitter"

        #  This procedure implements the transmission strategy;


        # mark pitch positions;
        if pp then
             ⊂  ppptr ← ppptr +1;
                if ppptr leq MAXPP
                       then pi[ppptr] ← t mod FRAMESIZE;
             ⊃;

        # compute total input energies since last transmission;
        Energy ← Energy + e[0]↑2;

        # save ref coeffs in ring buffers;

        rbp ← (rbp + 1) MOD RBSIZE;
        for i ← 1 upto pmax do
                rkb[rbp,i] ← K[i];

        #  This section of code implements the strategy for
        #  transmitting "reeasonable" reflection coefficients;

        # calculate transmit pointer in ring buffer;
        if pp then nextpp ← nextpps;  # const!;
        if lr then nextlr ← nextlrs;  # const!;
        if rs then nextrs ← nextrss;  # const!;
        if nextpp = 1 then dzcount ← dzcount max pptotdz;
        if nextlr = 1 then dzcount ← dzcount max lrtotdz;
        if nextrs = 1 then dzcount ← dzcount max rstotdz;
        if nextpp > 0 then nextpp ← nextpp - 1;
        if nextlr > 0 then nextlr ← nextlr - 1;
        if nextrs > 0 then nextrs ← nextrs - 1;

        #  if we are in a deadzone, then leave transmit pointer alone,
        #  otherwise, update it;

        if dzcount > 0
                then dzcount ← dzcount - 1
                else trp ← (rbp - otdel + RBSIZE) MOD RBSIZE;


        # if we are at a frame boundary then transmit parameter vector;
        if ((t+1) MOD framesize) = 0 then
          ⊂ "transmit"

                # move vector to output buffers;
                for i ← 1 upto pmax do
                       okb[obp,i] ← rkb[trp,i];
                for i ← 1 upto MAXPP do
                       opitchb[obp,i] ← pi[i];
                oenergyb[obp] ← Energy;

                #  if necessary, empty buffers;

                obp ← (obp + 1) MOD OBUFSIZE;
                if obp = 0 then
                   ⊂
                       aryWrite(fko,okb,kbsize);
                       aryWrite(fpo,opitchb,pbsize);
                       aryWrite(feo,oenergyb,OBUFSIZE);
                ⊃;

              #  reset pointers for new frame;

              for i ← 1 upto MAXPP do pi[i] ← -1;       # reset pitch indicators;
              ppptr ← 0;
              Energy ← 0;                               # reset residual energy;
          ⊃ "transmit";
    ⊃ "transmitter";
```

```
procedure ladder;
  ⊂ "ladder"
        # Perform one-step predictor ladder form;
        # Procedures required: initialize, pitchinit, trinit;
        #       orderupdate, transmitter;


        initialize;     # initialize ladder variables;
        trinit;         # initialize transmission strategy;
        pitchinit;      # initialize pitch detector variables;

      # data prescan to eliminate leading zeros;
        # skip all zero inputs;
        i ← 0;
        setPos (fpy, Fsample);
        aryRead(fpy,y,BUFSIZE);
        while (y[i] = 0) do
          ⊂
                i← i+1;
                if (i mod BUFSIZE) = 0 then
                        aryRead(fpy,y,BUFSIZE);
          ⊃;
        yt ← y[i];
        # check threshold of first input data;
        If abs(yt) < delta  then
         ⊂ ttyWrite (" |yt= ",yt,"| < delta = ",delta,",
                        yt ← sgn(yt)*delta ",NL);
              yt ← (yt/abs(yt)) * sqrt(delta);
         ⊃;

        setpos(fpy, Fsample+i);

      # start recursive ladder form;

      tt ← 0;
      for t ← 0 upto Nsamples do
      ⊂ "mainloop"

        if (tt mod BUFSIZE) = 0
                then ⊂  tt ← 0;
                        aryRead(fpy,y,BUFSIZE);
                     ⊃;

        if (t mod 512) = 0 then ttyWrite ("$");


        # set time min order index since last reset;
        tminpmax ← st min pmax;

        if t ≠ 0 then yt ←  y[ tt ];

        # compute weighting factor;
        if tau = 0
          then  # time-weighted;
            ⊂   t1I ← 1./(t + 1.);
                t0t1 ← t/(t + 1.);
                if t > 0 then ⊂ tI ← 1./t;
                                t1t0 ← (t + 1.)/t;
                              ⊃
                         else tI ← t1t0 ← 1;
            ⊃
          else  # exponential weighting;
          ⊂     ttau ← ( st + 10*pmax ) min tau;
```

```
                tI ← 1./ttau;
                tlI ← 1./(ttau + 1.);
                t0tl ← ttau/(ttau + 1.);
                tlt0 ← (ttau + 1.)/ttau;
          ⊃;

        #  Update delayed values;
        for i ← 0 upto pmax do
          ⊂     rZ[i]    ← r[i];
                eZ[i]    ← e[i];
                RrZ[i]   ← Rr[i];
                gZ[i]    ← g[i];
          ⊃;

        # start zeroth order ladder;
        e[0] ← r[0] ← yt;
        Re[0] ← Rr[0] ← Rr[0] + tlI*(yt*yt - Rr[0]);

        # order update the ladder;
        orderUpdate;

        # test for reset;
        if (tlI*g[1] geq resetsup) then rs ← TRUE
                                    else rs ← FALSE;

        if (tlI*g[tminpmax] < resetinf) then lr ← TRUE
                                         else lr ← FALSE;
        if rs v lr then reset;

        # call pitch detector;
        pitchdetect;

        # Update time index since last reset;
        st ← st + 1;
        tt ← tt + 1;

        # call transmitter;
        transmitter;

      ⊃ "mainloop";
  ⊃ "ladder";
```

```
#                        MAIN PROGRAM;

          ttyWrite("first sample = ");
          read(ttyRead, Fsample);
          ttyWrite ( "Nsamples  = ");
          read ( ttyRead, Nsamples );
          ttyWrite ( "pmax = ");
          read ( ttyRead, pmax );

          fileOpen;
          ladder;
          fileClose;

END "speech"
```

```
BEGIN "synthesis"

        comment
        This is a speech synthesizer program, it takes
        as input vectors transmitted by SPEECH program
        and synthesis speech according to the framesize
        and order of filter prescribed;
```

```
require "msailm.sai[exp,lee]" source_file;
require "msailp.sai[exp,lee]" load_module;
define PMAX = 12;
define OBUF = 4096;
define IBUF = 128;
define MAXPP = 6;
integer    p, ppmax, Nsamples, fr, t, i;
integer    ptcount;
real     Re, gnoise, gain, pulse, npulse, nu;
integer fpy,fpg,fpe,fpp,fpk;    # file handles;
integer bt;                     # buffer pointer;
real     yt;
real  array    e,  r,  rZ [ 0 tc PMAX ];
real  array    K[ 1 tc PMAX ];
real  array    C[ 0 tc IBUF , 1 tc PMAX ];
real  array    pp[ 0 tc IBUF , 1 tc MAXPP ];
real  array    pitch[ 1 tc MAXPP ];
real  array    y[ 0 tc OBUF ];
real  array    En[ 0 tc IBUF ];
real  array    rannum [ 0 tc 4000];
integer frnum, framesize;
integer obp;
integer cbsize,ppsize;
```

```
procedure initialize;
  ⊂ "initialize"
        ttyWrite ( " number of frames = " );
        read ( ttyRead, frnum );
        ttyWrite ( " framesize = " );
        read ( ttyRead, framesize );
        fpk ← open ( "coefficient filename : ", data ! input ! prompt );
        fpp ← open ( "pit.dat", data ! input );
        fpe ← open ( "en.dat", data ! input );
        fpg ← open ( "grand.dá[sp,lee]", data ! input  );
                aryRead ( fpg , rannum , 4000 );
                close ( fpg );
        for i ← 0 upto PMAX do
          e[i]  ←  r[i]  ←  0;
        yt ← 0;
        Nsamples ← 1;
        fpy ← open ( "y.p36", data ! create ! output );

        obp ← 0;
        for i ← 0 upto OBUF - 1 do
                y[i] ← 0;
        cbsize ← PMAX * IBUF;
        ppsize ← MAXPP * IBUF;

  ⊃ "initialize";
```

```
procedure ladder;
  ⊂ "ladder"

        for p ← 0 upto (pmax - 1 ) do  rZ[p] ← r[p];
        gnoise ← rannum [ Nsamples mod 4000 ];
        if pitch[1] = -1
         then    nu ← gnoise * gain
         else ⊂ nu ← npulse;
                for i ← 1 upto MAXPP do
                        if (t = pitch[i]) then nu ← pulse;
            ⊃;

        if  Nsamples < PMAX
           then
                for i ← 1 upto Nsamples do
                               nu ← nu * sqrt ( ( 1 - K[i]*K[i] ) );

        ppmax ← Nsamples min PMAX;

        e [ ppmax ] ← nu;

        for p ←  ppmax  downto 1 do
                e[p-1]  ←  e[p]  +  K[p] * rZ[p-1];

        for  p ← 1  upto ppmax  do
                r[p]  ←  rZ[p-1]  -  K[p] * e[p-1];

        yt  ←  r[0]  ←  e[0];

        y[obp] ← yt;
        obp ← (obp + 1) mod OBUF;
        if obp = 0 then
                aryWrite(fpy,y,OBUF);

        if (Nsamples mod 512) = 0 then
                ttyWrite("$");

        Nsamples ← Nsamples + 1;

  ⊃ "ladder";
```

```
initialize;
aryRead(fpk,C,cbsize);
aryRead(fpe,En,IBUF);
aryRead(fpp,pp,ppsize);

bt ← 0;

for fr ← 0 upto frnum - 1 do
    ⊂   # synthesis one frame of speech;

        for i ← 1 upto PMAX  do
                K[i] ← C[bt,i];
        Re ←  En[bt];
        for i ← 1 upto PMAX do
                Re ← Re * ( 1 - K[i]↑2 );
        for i ← 1 upto MAXPP do
                pitch[i] ← pp[bt, i];
        if pitch[1] = -1
          then  gain ← sqrt( Re/FRAMESIZE )
          else
           ⊂    ptcount ← 0;
                for i ← 1 upto MAXPP do
                        if pitch[i] neq -1 then ptcount ← ptcount + 1;
                pulse ← sqrt ( Re/ptcount );
                npulse ← 0;
          ⊃;

        for t ← 0 upto framesize - 1 do
                ladder;

        bt ← (bt + 1) mod IBUF;
        if bt = 0
         then ⊂
                aryRead(fpk,C,cbsize);
                aryRead(fpe,En,IBUF);
                aryRead(fpp,pp,ppsize);
            ⊃;
  ⊃;

        close(fpy);
        close(fpk);
        close(fpe);
        close(fpp);
END "synthesis";
```