

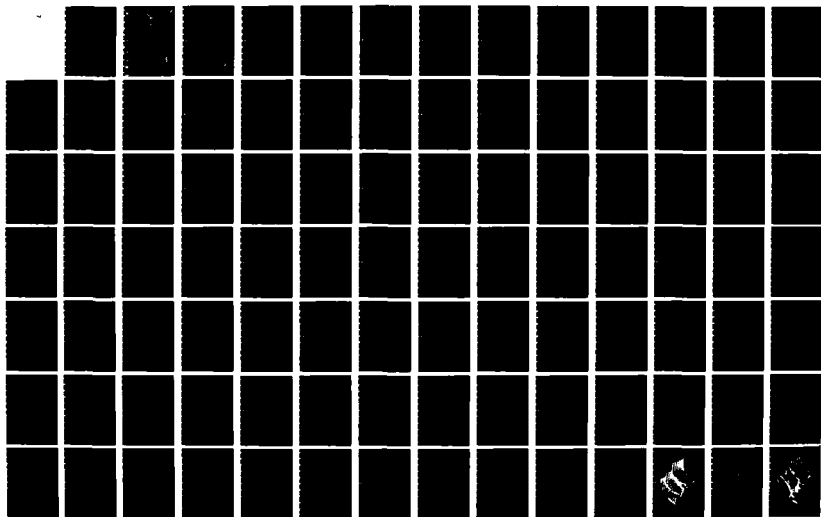
HD-A138 008

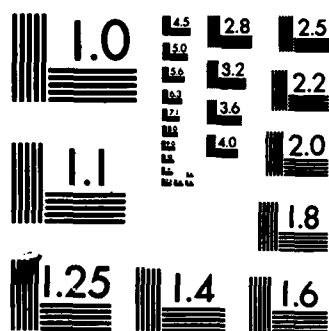
A RECURSIVE LINEAR PREDICTIVE VOCODER(U) AIR FORCE INST 1/4.
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
W A JANSSEN DEC 83 AFIT/GE/EE/83D-33

UNCLASSIFIED

F/G 17/2

NL





AD A138008



A RECURSIVE LINEAR PREDICTIVE VOCODER

THESIS

AFIT/GE/EE/83D-33

Willis A. Janssen

Capt

USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC
ELECTE
S FEB 21 1984

D

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution unlimited

84 02 17 056

AFIT/GE/EE/83D-33

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A/1	



A RECURSIVE LINEAR PREDICTIVE VOCODER

THESIS

AFIT/GE/EE/83D-33 Willis A. Janssen

Capt USAF

DTIC
ELECTRONIC
S FEB 21 1984
D

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

A RECURSIVE LINEAR PREDICTIVE VOCODER

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Willis A. Janssen, B.S.E.E.

Capt

USAF

Graduate Electrical Engineering

December 1983

Approved for public release; distribution unlimited

Preface

This work was motivated by the ideas and direction of Major Larry R. Kizer, Asst. Professor of Electrical Engineering Air Force Institute of Technology. My research effort produced a vocoder using recursive linear predictive coding. The following report describes recursive linear predictive coding, the vocoder, its outputs and utility programs for data analysis.

I would like to thank Major Larry Kizer for guidance through the research project. In addition, Major Kizer did a tremendous job ensuring that the Data General computer stayed operational. Dr. Matthew Kabrisky and Dr. Syed were most helpful in providing insight into the subjective effects of digital signal processing of speech. I would also like to thank Craig McKown for his debugging help and Kathy Ward for the signal processing programs she created. For other various types of help I would like to thank Larry Tieman, Mark Travis, and Dan Zambon. Finally I would like to thank my wife Kathi for helping me through another degree.

Willis A Janssen

Contents

	<u>Page</u>
Preface.	ii
List of Figures.	iv
Abstract	viii
I. Introduction.	I- 1
Justification	- 1
Background.	- 3
Problem	- 3
Scope	- 4
Approach.	- 4
II. Linear Predictive Coding.	II- 1
Linear Predictive Coding.	- 2
Autocorrelation Method.	- 7
Recursive Linear Predictive Coding.	- 9
III. Vocoder Model and Procedure	III- 1
Pitch Detector.	- 1
Speech Analysis	- 6
Speech Synthesis.	- 9
IV. Results and Conclusions	IV- 1
Tests	- 1
Vocoder Outputs	- 7
Data Rate	-17
Noise	-19
Conclusion.	-23
V. Recommendations.	V- 1
Bibliography	BIBO
Appendix A	A-1
Appendix B	B-1
Appendix C	C-1

List of Figures

<u>Figure</u>	<u>Page</u>
II-1 Recursive autocorrelation system (a) autocorrelation system diagram (b) digital filter.	II-14
III-1 AUTOC pitch detector block diagram.	III-15
III-2 SIFT pitch detector block diagram	III-16
III-3 RLPC analysis system block diagram.	III-17
III-4 RLPC synthesis system block diagram	III-18
III-5 Pulse generator block diagram	III-19
IV-1 Pitch period plots of speech file containing "The pipe began to rust while using (a) the AUTOC pitch detector (b) the SIFT pitch detector (c) the handpainted pitch plot. . .	IV-26
IV-2 Recursive autocorrelation window shapes for (a) $\alpha = .97$ (b) $\alpha = .98$	IV-27
IV-3 Skeleton axis for speech plots	IV-28
IV-4 Plot of the original speech file "The pipe began to rust while new"	IV-29
IV-5 Plot of vocoder output using "The pipe began to rust while new" as an input.	IV-37
IV-6 Skeleton axis for 3-D formant versus time plot	IV-45
IV-7 Formant versus time plot of the speech file "The pipe began to rust while new".	IV-46
IV-8 Plot of the original speech file "Thieves who rob friends deserve jail"	IV-103
IV-9 Plot of vocoder output using "Thieves who rob friends deserve jail" as an input	IV-111
IV-10 Pitch period plots of speech file containing "Thieves who rob friends deserve jail" using (a) the AUTOC pitch detector (b) the SIFT pitch detector (c) the handpainted pitch plot.	IV-119

IV-11	Plot of original speech file "One...two...three...four"	IV-120
IV-12	Plot of vocoder output using "One...two...three...four" as an input	IV-129
IV-13	Formant versus time plot using the word "one" with 80 samples (10 msec) between formant plots.	IV-138
IV-14	Formant versus time plot using the word "two" with 80 samples (10 msec) between formant plots.	IV-139
IV-15	Formant versus time plot using the word "three" with 80 samples (10 msec) between formant plots.	IV-140
IV-16	Formant versus time plot using the word "four" with 80 samples (10 msec) between formant plots.	IV-141
IV-17	Pitch period plot of the speech file "One...two...three...four" where (a) is without noise in input speech file (b) is with noise noise on the speech file.	IV-142
IV-18	Plot of the original speech file containing "Five...six...seven...eight".	IV-143
IV-19	Plot of vocoder output using "Five...six...seven...eight" as an input.	IV-151
IV-20	Formant versus time plot of the word "Five" with 10 samples (1.25msec) between formant plots	IV-159
IV-21	Formant versus time plot using the word "five" with 80 samples (10 msec) between formant plots	IV-166
IV-22	Formant versus time plot using the word "six" with 80 samples (10 msec) between formant plots	IV-167
IV-23	Formant versus time plot using the word "seven" with 80 samples (10 msec) between formant plots	IV-168
IV-24	Formant versus time plot using the word "eight" with 80 samples (10 msec) between formant plots	IV-169



Abstract

A non-real time 10 pole recursive autocorrelation linear predictive coding vocoder was created for use in studying effects of recursive autocorrelation on speech. The vocoder is composed of two interchangeable pitch detectors, a speech analyzer, and speech synthesizer. The time between updating filter coefficients is allowed to vary from .125 msec to 20 msec. The best quality was found using .125 msec between each update. The greatest change in quality was noted when changing from 20 msec/update to 10 msec/update.

Pitch period plots for the center clipping autocorrelation pitch detector (AUTOC) and simplified inverse filtering technique (SIFT) are provided. Plots of speech into and out of the vocoder are given. Formant ^{three dimensional} versus time ~~3-D~~ plots are shown.

Effects of noise on pitch detection and formants are shown. Noise effects the voiced/unvoiced decision process causing voiced speech to be re-constructed as unvoiced.



I. INTRODUCTION

Linear predictive coding (LPC) is one of the most powerful speech analysis methods. The importance of this method comes from its ability to provide extremely accurate estimates of the speech parameters and its speed of computation. Since the United States Air Force officially adopted LPC10 it is useful to study ways to improve it.

Justification

The need for further research on the topic of LPC stems from the three major areas within the general area of man-machine communication by voice. The major areas are:

1. Voice response systems;
2. Speaker recognition systems;
3. Speech recognition systems.

Each of these three areas have applications useful to the military. The voice response systems are designed to respond to a request for information such as a flight information system. The speaker recognition system could be used in speaker verification. Finally, speech recognition systems could be used on airplanes to update computer terminal displays. Table I-1 contains a list of several military tasks that would be useful to be automated in which LPC techniques would likely be part of the system. Therefore, any improvements in LPC systems bring us closer to realizing these applications.

Table I-1

Military Tasks for Possible Automation

- 1) Security
 - 1.1 Speaker Verification (Authentication)
 - 1.2 Speaker Identification (Recognition)
 - 1.3 Determining emotional state of speaker (e.g., stress effects)
 - 1.4 Recognition of spoken codes
 - 1.5 Secure access voice identification, whether or not in combination with fingerprints, identity card, ect.
 - 1.6 Surveillance of communication channels
- 2) Command and Control
 - 2.1 System control (ships, aircraft, fire control, situation displays, etc.)
 - 2.2 Voice-operation computer input/output (each telephone a terminal)
 - 2.3 Data handling and record control
 - 2.4 Material handling (mail, baggage, publications, industrial applications)
 - 2.5 Remote control (dangerous material)
 - 2.6 Administrative record control
- 3) Data Transmission and Communication
 - 3.1 Speech synthesis
 - 3.2 Vocoder systems
 - 3.3 Bandwidth reduction of, more general, bit-rate reduction
 - 3.4 Cipherring/coding/scrambling
- 4) Processing Distorted Speech
 - 4.1 Diver speech
 - 4.2 Astronaut Communication
 - 4.3 Underwater telephone
 - 4.4 Oxygen mask speech
 - 4.5 High "G" force speech

(Ref 3)

Background

Linear predictive coding (LPC) systems have been very successful in speech analysis and synthesis systems. They provide a robust, reliable and accurate method for estimating the parameters that characterize the linear time-varying speech system. Traditionally there are three commonly used methods for linear predictive analysis. These are the autocorrelation formulation, covariance method, and the lattice method. This thesis is concerned with the autocorrelation method.

The autocorrelation method requires windowing operations and buffering operations in addition to extensive computations (multiplies and adds). However, Barnwell (Ref 2) originated a recursive technique that uses an infinite length window which is also the impulse response of a recursive digital filter. Thus the autocorrelation can be done in a recursive manner resulting in computational advantages, great reductions in buffer storage requirements, and simpler control logic requirements. Since this idea is new many details of the recursive system contain unanswered questions, some of which will be the topic of this thesis.

Problem

The objective of this thesis is to create a recursive linear predictive coding (RLPC) vocoder system on the Data General Nova-Eclipse system. After implementation, some of the benefits of the recursive nature will be examined.

Scope

The vocoder is designed to be a tool for analyzing the effects of using recursive linear predictive coding with speech. The system and utilities will allow the user to create artificial vowels, plot synthesized speech, and make log magnitude plots for observing formant changes. The vocoder will not be a real time system.

Approach

The RLPC system will be created as a three part system allowing versatility for experimentation. The first part will be a pitch detector, the second will be the speech analyzer and the third will be the speech synthesizer. There are two particular advantages to this approach. First, intermediate data (for example, the predictor coefficients as a function of time out of the analyzer) are readily available for data analysis without having to run the complete system thus saving time. Second, a three part system allows the user to replace any part of the system with a different functional unit (for example, the Autoc pitch detector could be replaced with a Sift pitch detector).

II. Linear Predictive Coding

Linear predictive coding (LPC) is a commonly used technique for estimating the basic speech parameters (for example, pitch, formants, spectra, and vocal tract area functions). This method is important because it provides extremely accurate estimates of the speech parameters and is relatively fast.

Basically LPC is a technique that uses a linear combination of past speech samples to approximate a speech sample. A unique set of predictor coefficients are determined by minimizing the sum of the squared differences between the actual speech samples and the linearly predicted ones (Ref 7).

Three commonly used methods for formulating linear prediction analysis are: autocorrelation formulation, covariance method, and the lattice method. Only the first method will be discussed in this thesis. The autocorrelation formulation can be considered to be made up of two subtasks: the calculation of the autocorrelation function, and the matrix inversion of the autocorrelation matrix. The details of the second task will be discussed in chapter III. The first task is usually accomplished by evaluating a digital sequence using a short time autocorrelation function. However, in this thesis a recursive autocorrelation algorithm will replace the typically used short time autocorrelation function.

The rest of this chapter will be composed of a brief

review of basic linear predictive coding and a discussion of using recursion in linear predictive coding.

LPC

A LPC requires a sampled digital signal. In this case it will be assumed that a continuous speech signal is sampled at a frequency of $1/T$ where T is the sampling period. If the speech signal is $s(t)$ then the sampled version will be $s(nT)$. Henceforth, for notational simplicity, $s(nT)$ will be abbreviated by $s(n)$ as is generally accepted.

A model can be created in which the signal $s(n)$ is the output of a system with an unknown input $u(n)$ such that the following relation holds:

$$s(n) = - \sum_{k=1}^p a_k s(n-k) + G \sum_{l=0}^q b_l u(n-l), \quad b_0 = 1 \quad (2.1)$$

where a_k ; $1 \leq k \leq p$, b_l ; $1 \leq l \leq q$, and G are the parameters of the system. The past outputs are $s(n-k)$ and the past and present inputs are $u(n-k)$. Equation (2.1) describes the current output $s(n)$ as a linear combination of past outputs and past and present inputs. Thus the name linear prediction.

By taking the Z transform of equation (2.1) the model can be specified in the frequency domain where $H(z)$ is the

transfer function given as:

$$H(z) = S(z) / U(z) = G[1 + b_1 z^{-1}] / [1 - \sum_{k=1}^P a_k z^{-k}] \quad (2.2)$$

where

$$S(z) = \sum_{n=-\infty}^{+\infty} s(n) z^n \quad (2.3)$$

is the Z transform of $s(n)$ and $U(z)$ is the Z transform of $u(n)$.

Acoustic theory states that nasal and fricative sounds require both resonances and anti-resonances (poles and zeros). However, reasoning with Atal (Ref 1), the effect of a zero of a transfer function can be achieved by including more poles. This results in an all pole model given by:

$$H(z) = S(z) / U(z) = G / [1 - \sum_{k=1}^P a_k z^{-k}] \quad (2.4)$$

where all the variables are the same as in equation (2.2). This new all pole system has the advantage that the gain parameter, G , and the filter coefficients a_k can be estimated in a straight forward and computationally efficient manner using linear predictive analysis. Using an all pole model simplifies equation (2.1) so that the speech samples $s(n)$ are related to the input by the following

difference equation:

$$s(n) = \sum_{k=1}^p a_k s(n-k) + Gu(n) \quad (2.5)$$

Linear prediction of speech assumes that a sample of speech, $s(n)$, can be approximated by a weighted sum of the preceding p samples of speech given by:

$$\hat{s}(n) = \sum_{k=1}^p \alpha_k s(n-k) \quad (2.6)$$

where $\{\alpha_k\}$ is a set of predictor coefficients. The difference between the actual speech sample and linear predicted speech (equation(2.6)) is the prediction error and is given by:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^p \alpha_k s(n-k) \quad (2.7)$$

where $e(n)$ in equation (2.7) can be thought of as the output of a system whose transfer function is

$$A(z) = 1 - \sum_{k=1}^p \alpha_k z^{-k} \quad (2.8)$$

By comparing equation (2.5) and equation (2.7) it is noted that if the speech signal obeys the model of equation (2.5) exactly, and if $\alpha_k = a_k$, then $e(n) = Gu(n)$. Therefore, the prediction error filter, $A(z)$, will be an inverse filter for

the system, $H(z)$, given by:

$$H(z) = G/A(z) \quad (2.9)$$

Basically, linear predictive analysis determines a set of predictor coefficients $\{\alpha_k\}$ from the sampled speech signal in such a way as to obtain an estimate of the spectral properties of the speech signal using equation (2.9). Short segments of the speech signal are used to estimate the predictor coefficients because of the time-varying nature of the speech signal. Minimum mean-squared error techniques are used to determine the predictor coefficients which are assumed to be the parameters of the system function, $H(z)$, used in the model for speech production. This is a good assumption and this approach leads to a set of linear equations that can be efficiently solved to obtain the predictor parameters.

The short-time average prediction error defined over a segment of speech is:

$$E_n = \sum_m e_n^2(m) \quad (2.10)$$

$$= \sum_m \left[s_n(m) - \sum_{k=1}^p \alpha_k s_n(m-k) \right]^2 \quad (2.11)$$

where $s_n(m)$ is a segment of speech that has been selected in

the vicinity of sample n for example:

$$s_n(m) = s(m+n) \quad (2.12)$$

Note, the limits of the summation will not be specified now but will be later since this is a short-time analysis technique. Additionally the subscript n will be dropped because it is not needed in short-time analysis.

By setting the $\partial E_n / \partial \alpha = 0$, for $i=1,2,\dots,p$ the values of that minimize E_n can be found from the following equations:

$$\sum_m s(m-i)s(m) = \sum_{k=1}^p \alpha_k \sum_m s(m-i)s(m-k) \quad (2.13)$$

for $1 \leq i \leq p$ and where α_k are the values of α_k that minimize E . Henceforth the carat will be dropped and α_k will denote the values that minimize E . Defining $\phi_n(i,k)$ as:

$$\phi_n(i,k) = \sum_m s(m-i)s(m-k) \quad (2.14)$$

equation (2.13) can be rewritten as:

$$\sum_{k=1}^p \alpha_k \phi_n(i,k) = \phi(i,0) \quad (2.15)$$

for $i=1,2,\dots,p$. Now solving this set of p equations and p unknowns for the unknown predictor coefficients $\{\alpha_k\}$ that minimize the average squared prediction error for $s(m)$ we

have:

$$E_n = \sum_m s(m) - \sum_{k=1}^p \alpha_k \sum_m s(m) s(m-k) \quad (2.16)$$

or using equation (2.14) a simpler expression can be used:

$$E_n = \phi_n(0,0) - \sum_{k=1}^p \alpha_k \phi_n(0,k) \quad (2.17)$$

Solving for the optimum predictor coefficients requires computing the quantities $\phi_n(i,k)$ for $1 \leq i \leq p$ and $0 \leq k \leq p$ then solving equation (2.15) for the α_k 's. The details of solving the equations will be discussed using the autocorrelation method.

Autocorrelation method

First the limits on the sums in equation (2.10) and (2.11) must be determined. One commonly used technique is using a window where $s(m)$ is equal to 0 outside the interval $0 \leq m \leq N-1$ (Ref 10). This can be expressed as:

$$s(m) = s(m+n)w(m) \quad (2.18)$$

where $w(m)$ is a finite length window that is zero outside the interval $0 \leq m \leq N-1$. Using these limits $\phi_n(i,k)$ can be

expressed as:

$$\phi_n(i, k) = \sum_{m=0}^{N-1-(i-k)} s(m) s(m+i-k) \quad (2.19)$$

for $1 \leq i \leq p$ and $0 \leq k \leq p$. The short-time autocorrelation function is defined as:

$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w(m)] [x(n+m+k)w(k+m)] \quad (2.20)$$

Using equation (2.18) with (2.19) it is seen that equation (2.19) and (2.20) are the same, thus we have

$$\phi_n(i, k) = R_n(i-k) \quad (2.21)$$

Using the fact that the autocorrelation function is an even function E can be expressed as:

$$E_n = R_n(0) - \sum_{k=1}^p \alpha_k R_n(k) \quad (2.22)$$

This set of equations can be expressed in matrix form creating a $p \times p$ autocorrelation matrix which is a Toeplitz matrix. A Toeplitz matrix has the property that it is symmetric and all the elements along a given diagonal are equal. One solution for finding predictor coefficients from the Toeplitz matrix is to use Durbin's recursion algorithm (Ref 7). The algorithm will be discussed in chapter III as part of the procedure. This thesis does not use the

standard autocorrelation technique just described to find the autocorrelation matrix. Instead the standard technique is replaced with a recursive autocorrelation method.

RLPC

The idea for using a recursive technique to compute the autocorrelation matrix comes from Barnwell (Ref 2). Using the standard autocorrelation technique requires windowing operations and buffering operations in addition to extensive computations (multiplies and adds). However, the recursive technique uses an infinite length window which is also the impulse response of a recursive digital filter. Thus the autocorrelation estimation may be made recursive resulting in reductions in computation for some structures. Also great reductions in the buffer storage requirements and control logic requirements are realized. Finally this method results in a speech quality that is equivalent to the traditional Hamming window realization.

In the standard short-time autocorrelation method described in the last section a 20-30 ms Hamming window is typically used for computing the autocorrelation function. This technique has two problems. First, for good quality speech the window areas must overlap. This causes many speech samples to be used in forming the autocorrelation functions for more than one frame. Second, framing and buffering problems associated with handling overlapping windows cause computational architectures which are complex

and unwieldy.

If the requirement for a finite window is removed a class of windows can be used which, though infinite in length, are very small in magnitude outside of some finite region (for example a 20ms region). Of particular interest to this thesis is a class of windows which can be formed from the impulse response of a second-order digital filter having two real poles. This filter has a z transform given by:

$$H(z) = 1 / (1 - \alpha z^{-1}) (1 - \beta z^{-1}) \quad (2.23)$$

where α and β are the pole locations.

Now define the autocorrelation function for a windowed sequence to be written as:

$$R(k, m) = \sum_{n=-\infty}^{+\infty} s(n) s(n+k) w(m-n) w(m-n-k) \quad (2.24)$$

where $R(k, m)$ is the k th autocorrelation lag for window placement m . Defining:

$$w(n, k) = w(n) w(n-k) \quad (2.25)$$

and

$$s(n, k) = s(n) w(n+k) \quad (2.26)$$

equation (2.24) can be rewritten as:

$$R(k, m) = \sum_{n=-\infty}^{+\infty} s(n, k) w(m-n, k) \quad (2.27)$$

From this it can be seen that the k th autocorrelation lag is the convolution of the function $w(n, k)$ and the sequence $s(n, k)$. Additionally $w(n, k)$ is the product of two window functions, thus the Z transform of $w(n, k)$ is $W_k(z)$, the convolution of the Z transforms of the two window functions $w(n)$ and $w(n-k)$. Using a window of infinite length and a transfer function $H(z)$, the impulse response of a digital filter, $W_k(z)$ is:

$$W_k(z) = (1/2\pi j) \oint H(v) H(z/v) v^{-K-1} dv \quad (2.28)$$

using the window described in equation (2.23), equation (2.27) becomes:

$$W_k(z) = (1/2\pi j) \oint \left\{ v^{K-1} / [(1-\alpha v^{-1})(1-\beta v^{-1})(1-\alpha v/z)(1-\beta v/z)] \right\} dv \quad (2.29)$$

upon evaluating this equation $W_k(z)$ becomes:

$$W_k(z) = [b(0, k) + b(1, k) z^{-1}] / Q \quad (2.30)$$

where

$$Q=1-a(1,k)z^{-1}-a(2,k)z^{-2}-a(3,k)z^{-3} \quad (2.31)$$

$$b(0,k)=(\alpha^{K+1}-\beta^{K+1})/(\alpha-\beta) \quad (2.32)$$

$$b(1,k)=(\beta^{K+1}\alpha^2-\alpha^{K+1}\beta^2)/(\alpha-\beta) \quad (2.33)$$

$$a(1,k)=\alpha^2+\beta^2+\alpha\beta \quad (2.34)$$

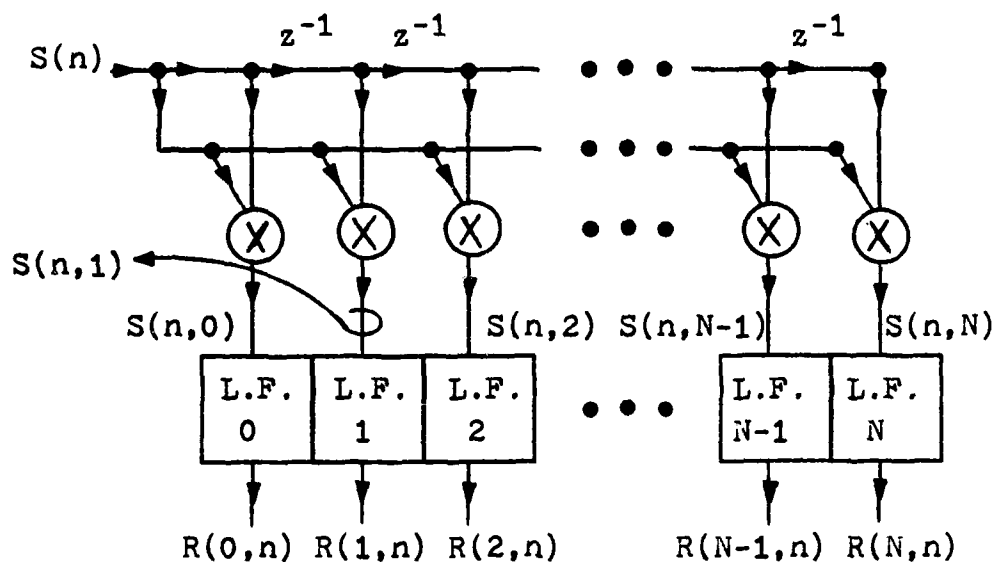
$$a(2,k)=-(\alpha^2\beta^2+\alpha^3\beta+\beta^3\alpha) \quad (2.35)$$

$$a(3,k)=\alpha^3\beta^3 \quad (2.36)$$

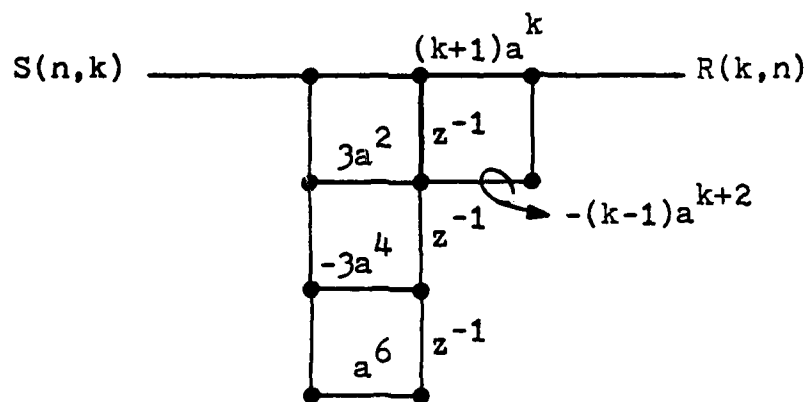
These equations reduce to a simpler form if α is allowed to equal β . The basic recursive structure is shown in figure II-1.

There are many points worth noting about the system in figure II-1. First, it is a point by point system which operates identically on every sample. Therefore additional buffering is not required which is most useful in a hardware system. Second, the parameter α completely controls the window length and the same number of computations are required regardless of window length or frame interval. Finally, only two multiplies are required in the nonrecursive portion of the filter on every frame interval and not on every sample (Ref 2).

In conclusion, RLPC is not different than other autocorrelation techniques because they both obtain a short-time autocorrelation analysis of speech by applying a window function. Thus the recursive technique retains all the important virtues of the autocorrelation technique including: reflection coefficients whose magnitude is always less than one, a stable receiver filter, and a Toeplitz autocorrelation matrix.



(a) Recursive Autocorrelation Structure



(b) Linear Filter (L.F.)

Figure II-1 Recursive Structure

III. The Vocoder Model and Procedure

This chapter describes how the recursive linear predictive coding vocoder used in this thesis is designed and implemented. The vocoder system will be discussed in three major sections; the pitch detector, the speech analyzer, and the speech synthesizer. For a software description see appendix A and for program listings see appendix B.

Pitch Detector

Four basic problems exist in detecting the pitch period of a speech signal. First, the glottal excitation waveform is not a perfect train of periodic pulses (for an excellent introduction in the mechanics of speech production see Ref 10). Although finding the period of a perfectly periodic waveform is straight forward, a speech waveform varies both in period and in the detailed waveform structure causing pitch period determination to be difficult. Secondly, the pitch period must be extracted from an output which is caused by the interaction between the vocal tract and the glottal excitation. In some cases the formants of the vocal tract alter the structure of the glottal waveform so that the actual pitch period is difficult to detect. The third problem comes from the inherent difficulty in defining the exact beginning and end of each pitch period during voiced speech segments. Finally, the fourth difficulty occurs in distinguishing between unvoiced speech and low-level voiced

speech. Often transitions between unvoiced speech segments and low-level voiced speech segments are very subtle, thus making them very hard to pinpoint.

To find a pitch detector that would adequately perform given the problems described above, several pitch detectors were reviewed for use in this LPC vocoder (Ref 4). Two pitch detectors were chosen to be used. The first pitch detector chosen was the center clipping autocorrelation pitch detector (AUTOC) and the second is a modified version of the simplified inverse filter tracking (SIFT) pitch detector.

The center clipping autocorrelation pitch detector (AUTOC) was chosen because its performance characteristics were equal to or better than the other pitch detection algorithms. Specifically, for the fine pitch error category (fewer than 10 continuous errors) AUTOC performed equal to the cepstrum (CEP) method, the simplified inverse filtering technique (SIFT), and the linear predictive coding (LPC) using pattern recognition and spectral equalization. In the category of gross pitch errors (more than 10 continuous errors) AUTOC had average performance. The second pitch detector was chosen because the code has been published (Ref 9) and only a few changes were required to have a functioning alternate pitch detector.

The first pitch detector to be discussed is the AUTOC pitch detector. Figure III-1 is a block diagram of the AUTOC system. The input is a data file of speech that has

been sampled at 8000 Hz with a range in magnitude from -2048 to 2047 (in 2's compliment representation). The original speech was input in an active laboratory environment with a maximum possible input level of -5.0 to +5.0 volts. The input speech data is passed through a low pass filter with a 900Hz cut off frequency.

The speech is divided into 10ms segments called sets where three sets make up one frame. Frames are 30ms long and each frame overlaps the previous frame by 20ms (two sets). The average energy is calculated for each frame using the following formula:

$$E = \sum_{m=-\infty}^{+\infty} [x(m)w(n-m)]^2 \quad (3.1)$$

A 120 point rectangular window, $w(n)$, was chosen for windowing the speech. This compromise in size was chosen to be responsive to rapid amplitude changes (best with a short window) yet provide sufficient averaging to produce a smooth energy function (best with a long window). The energy is summed at each point in the frame and if any of these sums exceed the voiced/unvoiced threshold the frame is considered voiced and energy processing on the current frame stops. Next the frame is processed in a peak detector algorithm.

The peak detector algorithm determines the largest absolute value of speech in the first and third set of each frame and compares them. The smaller of the two is multiplied by a constant (in the range of .6 to .8) and the

result is called the clipping level C_1 . The clipping level was chosen in this manner to avoid over clipping when the speech varies greatly in intensity from the beginning of the frame to the end. The clipping level will be used in the center and infinite peak clipper. When a large clipping level is chosen, fewer peaks will exceed the clipping level, thus fewer will appear in the autocorrelation function. A smaller clipping level allows more peaks to pass through thus the autocorrelation function is more complex (more peaks). So a higher clipping level is preferred to minimize the autocorrelation function's complexity which in turn simplifies the process of finding the fundamental autocorrelation peak. Therefore, by checking at the beginning and the end of the frame a reasonable high clipping level can be chosen which produces a smoother autocorrelation function.

Next the speech is passed through the clipper which uses the previously determined clipping level. The clipper does center and infinite peak clipping. If a speech value exceeds C_1 or $-C_1$ it is assigned a value of 1.0 or -1.0 respectively and anything in between is assigned a value of 0.0.

A short-time autocorrelation operation is performed on the clipped speech using:

$$R(k) = \sum_{m=0}^{N-1} x(n+m)x(n+m+k)$$

where x_1 and x_2 have been windowed by rectangular windows 80 and 240 samples long respectively. Also, n is the frame edge, m is the location in the frame being autocorrelated and k is the current autocorrelation lag. From this it can be seen that lags as large as 160 samples can occur. With speech being sampled at 8000 Hz pitch periods as large as 20ms (or equivalently 50Hz) or as small as 2ms (or equivalently 500Hz) can be found. This range is sufficient to determine the fundamental excitation period. The pitch period of a frame is determined by finding the first autocorrelation value (beyond 2ms) that exceeds 80% of $R(0)$.

The second pitch detector to be used is the simplified inverse filtering technique (SIFT). Since the basic code and detailed description of SIFT has been published in Ref 9, a less detailed description than the one for the previous pitch detector will be given. Figure III-2 is a block diagram of the SIFT pitch detector. A block of 400 speech samples (sampled at 8KHz) is low pass filtered to a bandwidth of 900Hz and then decimated by a 5 to 1 ratio. The autocorrelation method of LPC analysis is used to find the coefficients of a 4th-order inverse filter. The 1600Hz speech signal (due to decimation) is passed through the inverse filter to produce a spectrally flattened signal which is then autocorrelated. By using parabolic interpolation in the region of the peak of the autocorrelation function the pitch period is found. The voiced/unvoiced decision is made on the basis of the

amplitude of the peak of the autocorrelation function. In addition to this a silence detector is used to determine if speech or silence is present. The silence detector computes the energy in a frame and if it exceeds the silence threshold then it is speech otherwise it is considered silence.

As a final output, both pitch detectors provide the following output information:

- 1) Whether the frame is silent or not;
- 2) voiced or unvoiced;
- 3) if voiced, what the pitch period is.

Speech Analysis

The speech analysis technique used in this thesis is the standard autocorrelation method where the short-time autocorrelation computation was replaced with a recursive autocorrelation computation. Figure III-3 is a block diagram of the analysis system. Basically the system requires three operations: first the autocorrelation is performed on some segment of speech, second the LPC predictor coefficients are found using Durbin's recursion (Ref 7), and finally the gain is determined.

In addition to the system described above a module was created to preprocess speech before the autocorrelator. This module is a FIR pre-emphasis filter of the following

form:

$$y(n) = x(n) - .9x(n-1) \quad (3.2)$$

where $x(n)$ is the input to the filter and $y(n)$ is the output. The pre-emphasis filter is used to make the spectrum as flat as possible which has been shown to minimize the effect of roundoff errors causing ill-conditioned matrices (Ref 9).

The autocorrelation method used in this thesis has been discussed extensively in chapter II and the system design is shown in figure II-1. However, the choice of the order (p) of the predictor polynomial has not been discussed. It is well known that p must be large enough to account for both the vocal tract and glottal pulse effects. However, as p increases the computational burden increases. Determining what value of p to use is resolved by observing the normalized rms prediction error versus the predictor order p for sections of voiced and unvoiced speech (Ref 10). For p on the order of 13-14 the error essentially flattens out showing only small decreases as p is increased further. Also it was found that the choice of p depends primarily on the sampling rate (independent of the LPC method used). In this thesis an 8khz sampling rate is used which requires approximately 8 poles to represent the vocal tract. Additionally 2-4 poles are required to adequately represent the source excitation spectrum and the radiation load. Thus

p was chosen to be of order 10.

As discussed in chapter II for the autocorrelation method the matrix equation for solving for the predictor coefficients is of the form:

$$\sum_{k=1}^p \alpha_k R_n(i-k) = R_n(i) \quad 1 \leq i \leq p \quad (3.3)$$

The most efficient method known for solving this set of equations exploits the Toeplitz nature of the matrix of coefficients. This method is Durbin's recursive procedure (Ref 7) which is given in equations 3.4 to 3.8:

$$E^{(0)} = R(0) \quad (3.4)$$

$$k_i = [R(i) - \sum_{k=1}^i \alpha^{(i-1)}_k R(i-k)] / E^{(i-1)} \quad 1 \leq i \leq p \quad (3.5)$$

$$\alpha^{(i)}_k = k_i \quad (3.6)$$

$$\alpha^{(i)}_j = \alpha^{(i-1)}_j - k_i \alpha^{(i-1)}_i \quad 1 \leq j \leq i-1 \quad (3.7)$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \quad (3.8)$$

Equations 3.5 to 3.9 are solved recursively for $i=1,2,\dots,p$ and the final solution is:

$$\alpha_j = \alpha^{(p)}_j \quad 1 \leq j \leq p \quad (3.9)$$

The last part of the analysis system determines the system gain for the current frame. This is found using the following relation:

$$G^2 = R(0) - \sum_{k=1}^P a_k R(k) \quad (3.10)$$

where G is the gain.

As a final output the analysis system provides the following information for each frame:

- 1) A set of predictor coefficients;
- 2) the gain.

Speech Synthesis

The speech synthesizer is composed of five basic parts including a pulse generator, noise generator, switch, receiver filter, and a D/A converter. These elements are shown in figure III-4. The receiver filter can be thought of as a vocal tract that is excited by the pulse generator or noise generator for voiced or unvoiced sounds respectively.

If the current speech to be produced is unvoiced (determined earlier by the pitch detector) the switch causes the noise generator to be the input to the receiver. Conversely for voiced speech the switch connects the pulse generator to the receiver input filter.

For voiced speech the pulse generator can be thought of

as a three part system consisting of an impulse train generator, a glottal pulse model, and an amplitude control where the system's input is a pitch period and the output is a glottal pulse shape. The system is shown in figure III-5. Current pitch information will come from the pitch detector causing the glottal pulse model to be impulsed at the pitch period. The output pulse will then be multiplied by the current gain value received from the speech analysis system.

In experiments, Rosenberg (Ref 11), discovered that good quality synthetic speech can be obtained using excitation functions which can be specified by trigonometric or polynomial functions uniformly throughout the vowel portions of an utterance. The results of his experimentation showed that the most preferred pulse shapes have only one single slope discontinuity which is at the closing end of the pulse. One of the waveshapes rated highest in Rosenberg's tests and the waveshape used in this thesis is:

$$\begin{aligned} g(n) &= (1/2) [1 - \cos(\pi n/T_p)] & 0 < n < T_p & \quad (3.11) \\ &= \cos(\pi(n - T_p)/2T_N) & T_p < n < T_p + T_N \\ &= 0 & \text{elsewhere} \end{aligned}$$

where T_p and T_N are the portions of the waveform with a positive and negative slope respectively. The tests also demonstrated that there is no particular best value for T_p and T_N . However, combinations having very small opening or closing times, or opening times less than or approximately

equal to closing time were not ranked well in listening tests. The values of T_p and T_N chosen for this thesis are discussed in chapter IV.

In contrast to the pulse generator which produces a pulse the noise generator produces impulses of varying magnitude. For discrete-time models (such as this), the random number generator provides a source of flat-spectrum noise. The probability distribution of the noise was chosen to be normal although there seems (Ref 10) to be no distribution which is considered the best.

The algorithm for the normal generator required a uniformly distributed random generator as an input. A prime modulus multiplicative linear congruential generator (PMMLCG) was chosen as the uniform random number generator. This decision was made purely because Ref 5:227 had available a Fortran listing of a proven uniform generator for 16 bit machines such as the Data General Eclipse. The problem with many uniform random number generators is that they have undesirable statistical properties (although this may not be a problem in speech). Rather than going into the details of PMMLCG systems which is not pertinent to speech production the reader can see Ref 5 for more information.

The algorithm for the normal generator was chosen as the best of the current normal generating algorithms available using the criteria of efficiency (speed and storage requirements) and quality or accuracy of the output (Ref 6). The prime benefit of this algorithm is its speed with

average storage requirements compared to other algorithms.

Referring back to figure III-4 the excitation to the synthesis filter varies with time as would be the case in a true glottus. In a similar manner the synthesis filter changes its filter coefficients with time similar to the vocal tract changing shape. This is only an analogy and the particular values of the filter coefficients are a function of what type of digital filter is chosen and how the parameters were created (for example, autocorrelation, covariance etc.).

Generally filter parameters are estimated at regular intervals in regions of voiced speech and filter coefficients are updated at the beginning of each pitch period. For unvoiced speech the coefficients are updated once at the beginning of each frame. This thesis requires a slightly different approach since filter coefficients are available for each speech sample. One interesting question is: how often is the useful (new) information available to update the synthesis filter? This imposes the requirement that the synthesis filter must be able to be updated at any time. As stated earlier, typical systems usually update their parameters each pitch period (called pitch synchronous synthesis) which has been found to be more effective than updating parameters once per frame (called asynchronous synthesis). Unfortunately, the pitch synchronous synthesis technique has the problem that it requires filter coefficients that must be found by the interpolation of the

predictor coefficients, which are not always stable. This problem, although common and easily solved, does not exist in this synthesis system because of the recursive nature of the analysis system makes filter coefficient information available at any synthesized speech sample.

The speech analysis system provides filter coefficients configured in the direct form. Using some simple transformations other filter forms could be used for the synthesis filter. But since the thrust of this thesis isn't affected by the choice of synthesis filters, the direct form filter was chosen because it is easier to implement.

Since a pre-emphasis filter was used in the analysis portion of the system, a de-emphasis filter must be used in the synthesis portion. The synthesized speech is passed through a filter of the following form:

$$y(n) = x(n) + .9y(n-1)$$

where $x(n)$ is the filter input and $y(n)$ is the filter output.

As a final output of the synthesis system the following information is provided:

- 1) An integer file of speech.

In summary, the vocoder is composed of a pitch detector, a speech analyzer, and speech synthesizer. The pitch detector determines if a speech segment is silence, voiced

or unvoiced, and if voiced a pitch period. Then the speech analyzer uses recursive autocorrelation with Durbin's recursion algorithm to compute LPC predictor coefficients. Speech is then synthesized by the speech synthesizer using information provided by the pitch detector and speech analyzer.

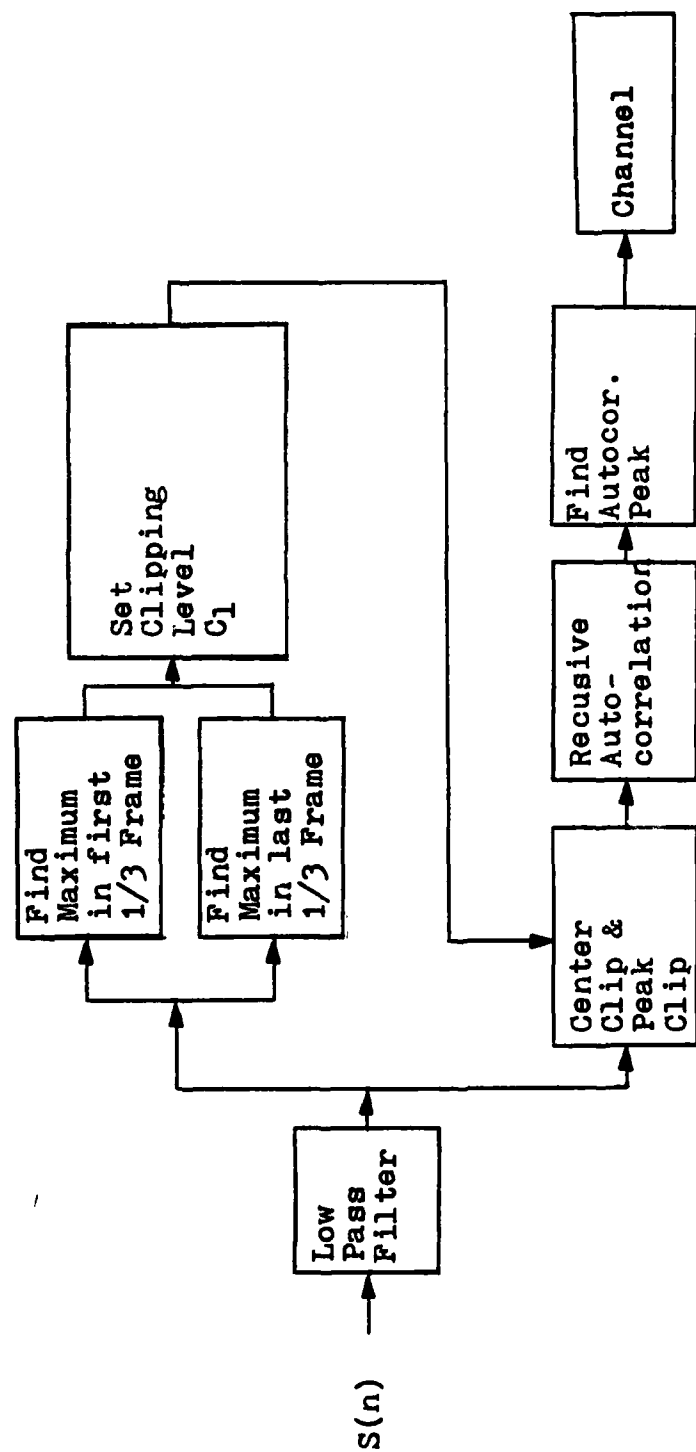


Figure III-1 Block Diagram of the AUTOC Pitch Detector

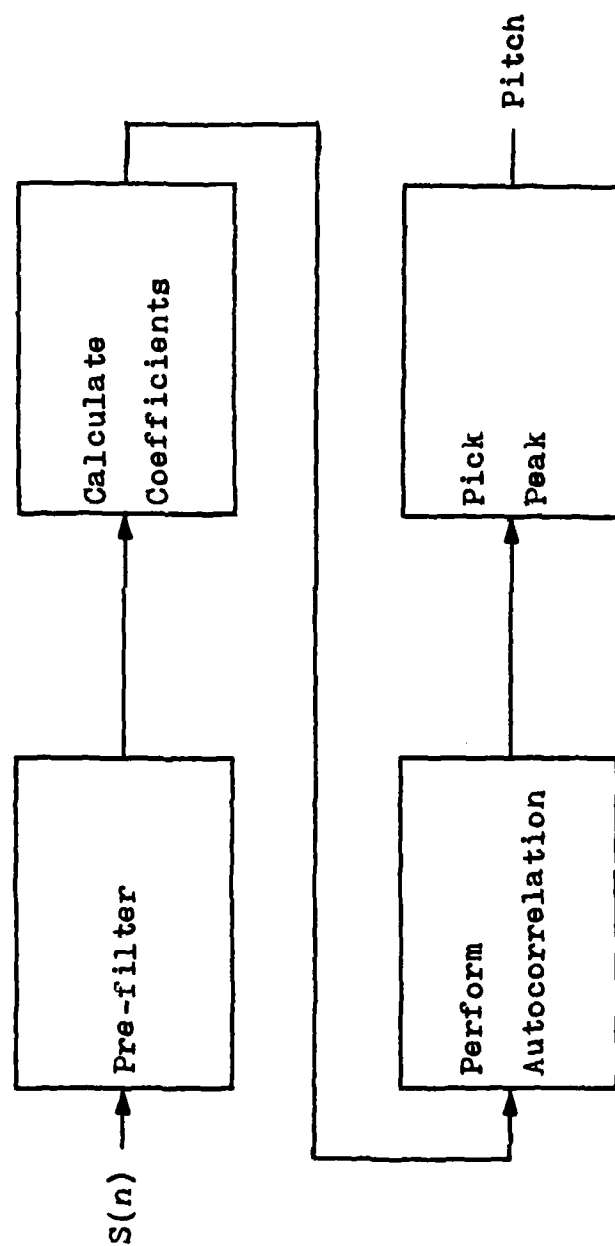


Figure III-2 Block Diagram of the SIFT Pitch Detector

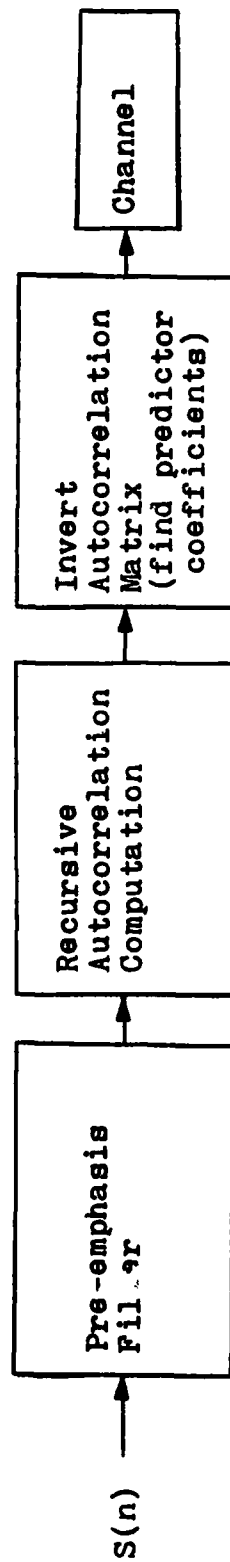


Figure III-3 Block Diagram of Speech Analyzer

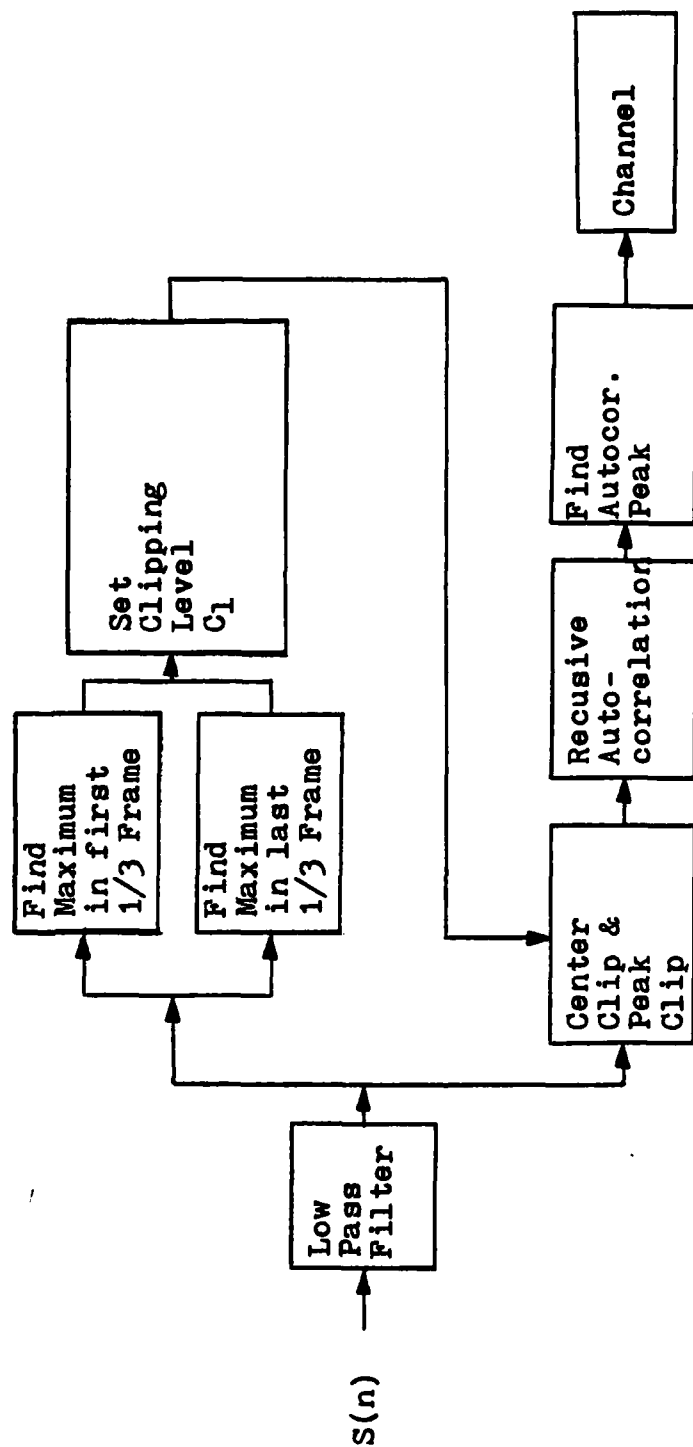


Figure III-1 Block Diagram of the AUTCC Pitch Detector

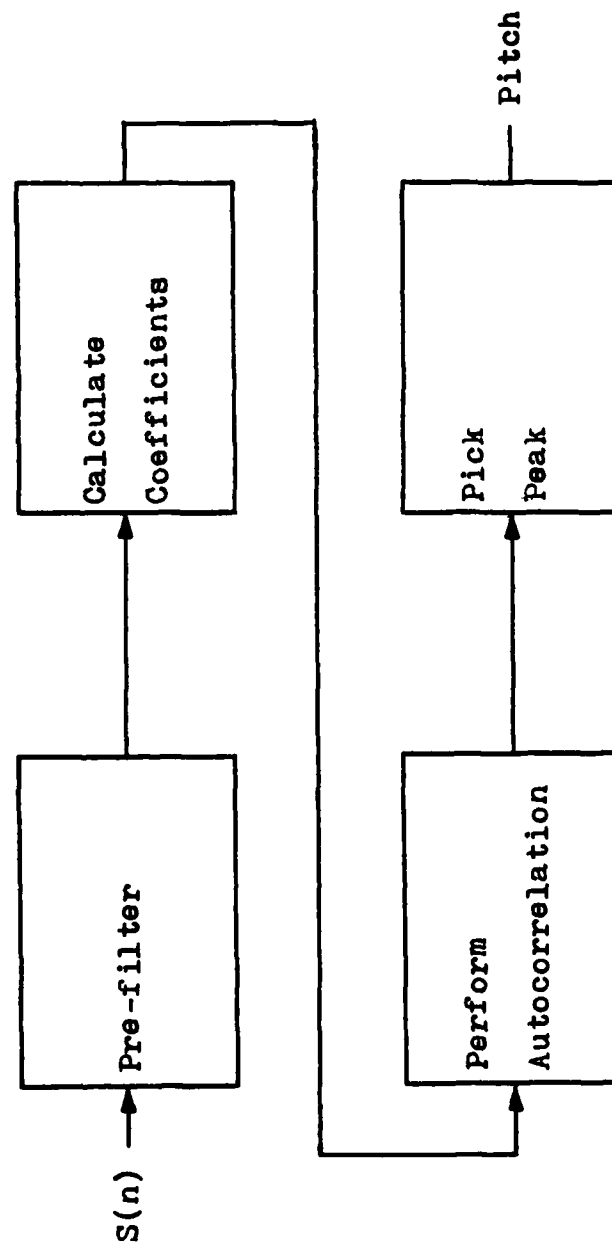


Figure III-2 Block Diagram of the SIFT Pitch Detector

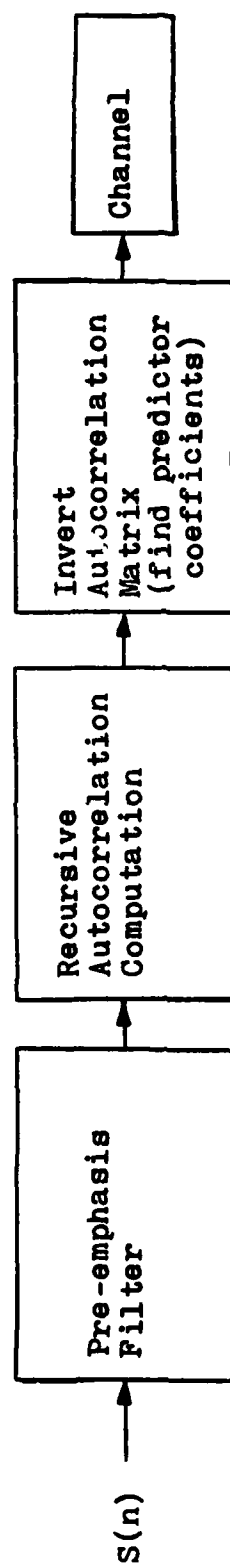


Figure III-3 Block Diagram of Speech Analyzer

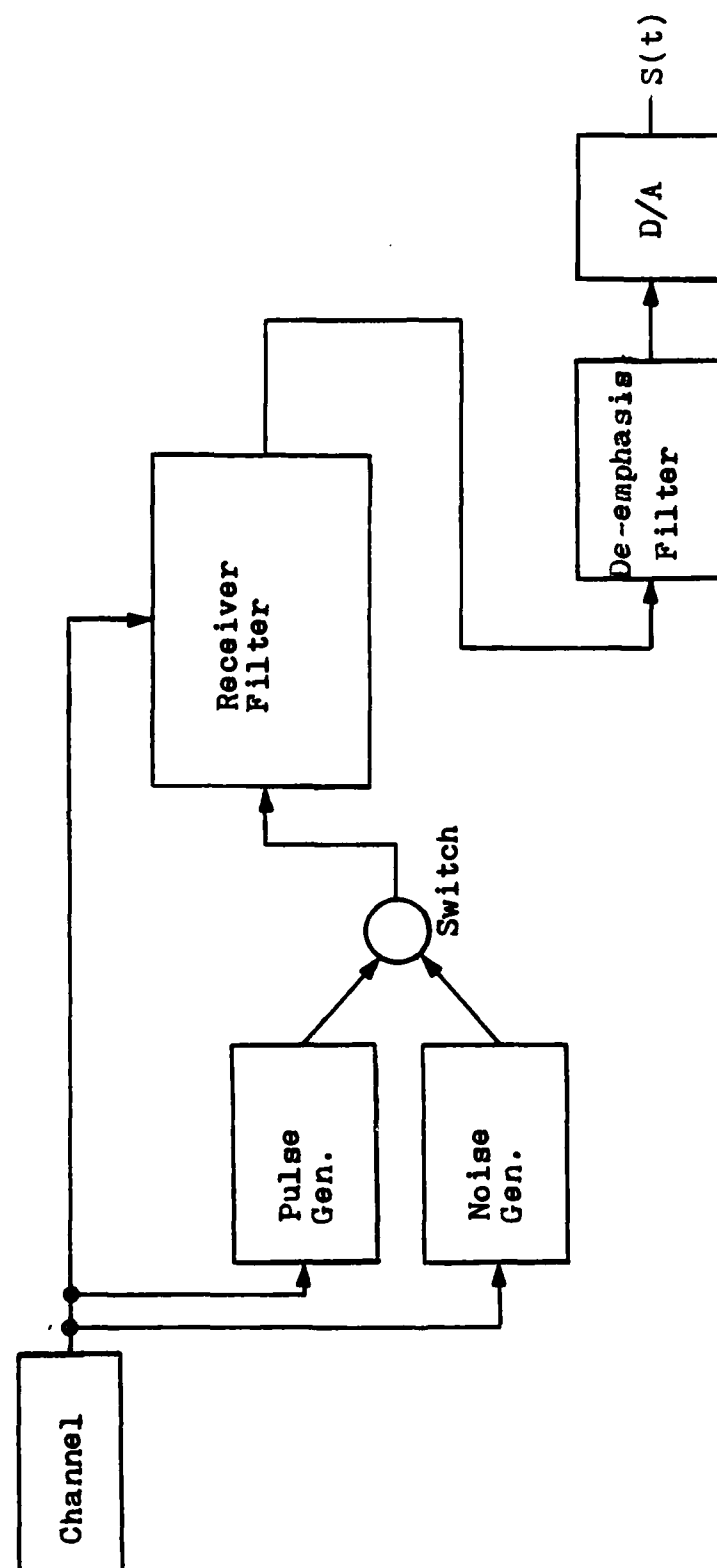


Figure III-4 Block Diagram of Speech Synthesizer

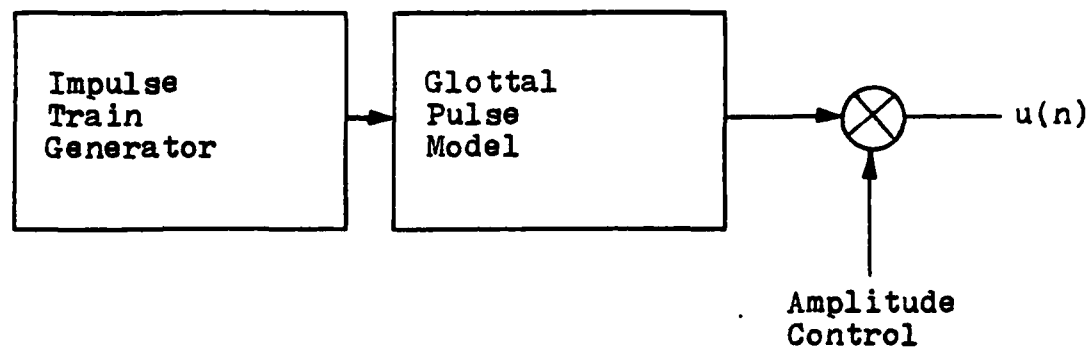


Figure III-5 Block Diagram of Pulse Generator

IV. Results and Conclusions

This chapter will be divided into five sections. The first of these will discuss tests that were performed to find the best quality of speech by varying parameters such as silence thresholds, glottal pulse shape, etc. The second section describes some of the vocoder outputs using time domain plots and 3-dimensional spectral plots (log magnitude). The third section will look at the data rate of the vocoder and the effects of using recursion. The fourth section looks at some of the effects noise has on the RLPC system. The final section will be the conclusion section.

Tests

The goal of the tests described in this section was to find the combination of parameters that caused the vocoder to produce the best "quality of speech." Unfortunately a measure of the "quality of speech" is not easy. For the purposes of this thesis informal listening tests were used. Each of the parameters that were varied are essentially independent, thereby allowing optimization tests to be performed on one parameter at a time.

The tests were performed using the same utterance as an input to the vocoder for each parameter change. The range of the parameters were changed in such a way that the whole range of reasonable values were checked. For example, in the case of the silence threshold one value was chosen large enough to clip speech information and the other extreme was

made small enough to be obvious that excessive noise was input. By using this method of choosing extremes, optimum parameter values would not be outside of the range of testing and thus would not be missed. By varying parameters between the extremes an informal listening group was able to listen to the vocoder output and choose the best parameter.

Six speech files (S1-S6) were available in the Data General Nova-Eclipse system. The files are two's complement integer files with an absolute maximum value of 32,768. Out of the six possible files two were chosen, one was spoken by a man and the other was a woman. There was no reason to choose a particular file but it seemed reasonable to choose one male and one female. The two utterances chosen were S1 and S2 (number 1 and 2, below, respectively). The utterance in each file is listed below:

- 1) S1-"The pipe began to rust while new"-female speaker
- 2) S2-"Thieves who rob friends deserve jail"-male speaker
- 3) S3-"Add the sum to the product of these two"-female speaker
- 4) S4-"Open the crate but don't break the glass"-male speaker
- 5) S5-"Oak is strong and also gives shade"-male speaker
- 6) S6-"Cats and dogs each hate the other"-male speaker

Also three additional speech files (the speaker was Major Kizer) were used and are listed below:

- 1) file "ONE"-containing"one...two...three...four"
- 2) file "FIVE"-containing"five...six...
seven...eight"
- 3) file "ZERO"-containing"zero...nine...ten"

The first test to be performed was the silence threshold test. The goal of this test was to determine the silence threshold that eliminated unwanted noise (during intervals without speech) but did not eliminate any speech. The silence detection is accomplished in the pitch detector. Two pitch detectors were used in this thesis and because of processing differences different silence thresholds are needed for each pitch detector. Using the SIFT pitch detector the best silence threshold was found to be .1 (which equates to .1% of the rms energy in a sine wave with the maximum integer value of 32,768). Using the AUTOC pitch detector the best silence threshold was found to 5.0 .

For the AUTOC pitch detector two additional tests had to be performed. These tests found the optimum clipping level and autocorrelation threshold both of which were discussed in chapter III. These tests were the only tests in which informal listening tests were not used exclusively. Instead the AUTOC pitch detector output was plotted for the speech files S1-S6. These plots were compared with a pitch period plot of the original speech file. Unfortunately, the time scale information was not available for the original pitch period plot. However, general pitch trends could be observed to determine if the pitch detector was close. See figure IV-1 (pitch plot of S1) for an example of this. Of course this really is only a qualitative test and an actual listening test is required to verify the pitch detectors quality. The best choice found for the clipping level was

0.6 and 0.3 for the autocorrelation threshold. These parameter choices were then verified using informal listening tests.

The next test compared the quality of speech produced using each pitch detector. See figure IV-1 for a pitch period plot of S1 using each pitch detector and a plot of the original "handpainted" pitch period plot (note: the hand painted plot is not on the same time scale as the other two). The best speech was produced using the SIFT pitch detector in the vocoder. The synthesized speech using the AUTO C pitch detector tended to waver (even after trying linear smoothing, median smoothing, and the two combined). Since the primary purpose of this thesis is to create a system which allows the user to experiment with recursive LPC the SIFT pitch detector was chosen to be used in the remainder of the tests. In choosing the SIFT pitch detector, higher quality speech will be produced. Therefore subtle LPC effects will not be masked by a lower quality pitch detector.

The next test found a ratio between unvoiced and voiced impulse values. In a qualitative way this allowed the energy in voiced and unvoiced speech to be modified. To understand why this ratio is necessary consider the digital filter synthesizer. For unvoiced speech, energy is being continually input to the digital filter (where the actual energy is determined by the random process used). However, in the case of voiced speech the digital filter is only

impulsed at the beginning of each pitch period. To account for this difference in energies, informal listening tests were used to find the best unvoiced/voiced multiplier value. Additionally this scaling of the random generator could be thought of as decreasing the variance of the random process. This number turned out to be smaller than 1.0 as would be expected since unvoiced speech has less energy than voiced speech. The actual value found was 0.1 . For larger values the synthesized speech tended to be scratchy sounding and smaller values produced speech without good quality fricatives.

The next test used each of the six speech files described earlier as an input to the vocoder. Basically, the quality of the processed speech was understandable with the speech produced by males being generally of higher quality than those produced by female voices. The final observation is that the reconstructed speech did not seem to have the tonal quality (or timbre) present in the original speech.

Whether to use a glottal impulse or a glottal pulse shape was the subject of the next test. As stated in chapter III Rosenberg (Ref 11) found that various glottal pulse wave shapes provide good quality speech. The vocoder created for this thesis allows either a trigonometric wave form or an impulse to be used for the glottal pulse. One of the better waveforms used in Rosenberg's tests occupied approximately forty percent of the pitch period. When a

glottal pulse of this duration was used the speech quality was very poor. Only when the glottal pulse occupied ten percent or less of a pitch period did the quality of the speech become reasonable. However, when an impulse was used the quality was best. The impulse seemed to produce crisper, cleaner speech; whereas the glottal pulse shaped wave form tended to be muffled sounding. Although other wave shapes could have been chosen from Rosenberg's tests (the trigonometric was considered one of the best in the tests) this author believes that the shapes of the waveforms were similar enough that another waveform choice would not have changed the results.

The final test varied the recursive autocorrelation window shape. See figure IV-2 (a) for a plot of the window shape with $\alpha=.97$ and figure IV-2 (b) for $\alpha=.98$. Tests by Makhoul and Cosell (Ref 8) of speech quality for various window shapes α (see chapter II equation (2.23) to see where α is used) was varied from .97467 to .97979 (actually they used the square of α but this author found the square root for easier comparisons). However, in Barnwell's testing α was varied from .97900 to .98500. In this test α was varied from .97750 to .98500. The quality of the speech over the range of the α s did not vary much. The value .98000 was the best choice for α but it wasn't a great deal better than the extremes of α .

VOCODER OUTPUT

This section has two purposes where the first is to demonstrate possible outputs available from the vocoder system and second show some of the effects and benefits of the recursive autocorrelation technique.

As discussed in the last section initially six speech files were used (S1-S6). Figure IV-2.1 to IV-2.8 is a plot of the original speech file S1 ("The pipe began to rust while new"). These and all other plotted vocoder outputs, unless otherwise specified, were created using the SIFT pitch detector and the following vocoder parameters:

1) window shape-	.98000
2) silence threshold-	.10000
3) unvoiced to voiced gain-	.10000
4) frame separation-	10
5) pre/de-emphasis-	used
6) glottal pulse-	impulse used

Before getting into the details of using the speech (S1) some basic information on how the plots of the speech files are structured will be given. First, the figures were ordered so that the figure number represents one complete speech file. The number following the decimal point simply identifies sequential speech plots. In addition to this, plots on a given page are also time sequential (for example, the second plot from the top follows the first etc.) because the speech is continuous. The vertical scale of each plot is the integer value of speech at a given sample time (one two byte two's complimentary value in the speech file which was scaled for plotting). The horizontal scale represents time (in samples at a sampled rate of 8000HZ) or more

conveniently each point represents one data point in the speech file. For the rest of the thesis unless otherwise specified the horizontal axis will be described as a sequence of data points. From the plot it can be seen that there are 512 data points per graph which is conveniently 2 blocks of data. Typically speech files are 88 blocks long and from figure IV-4 the it can be seen that only 80 blocks are shown. There was no reason to include plots of silence from the begining of the speech file so they were not included. One final note: a particular graph (a plot of 2 blocks of data) of a speech file will be identified by a figure number, a plot number (after decimal point) and the number of a particular graph on one page will follow a second decimal point where they are numbered from top to bottom. For example, the third graph from the top of figure IV-4.3 would be identified as figure IV-4.3.3. For details on creating these plots see appendix C.

Looking at an isolated speech plot does not provide one with much information. For example, looking at figure IV-4.1.5 only tells one that part of the speech is voiced. Most people would not recognize this figure as the word "The". Additionally, it is difficult to separate the beginning and end of two connected phonemes.

To make the speech plots more usable the various letters of an utterance have been included under the plot. A slash "/" indicates the approximate beginning, ending or seperation point between parts of the utterance. For

example the "/" in figure IV-4.6.5 separates the "wh" and "i" sound in "while". Also the utterance is printed at the bottom of the figure with the portion of the utterance underlined that is contained on the page of plots.

A short discussion of the types of phonemes and some example phonemes will be presented to aide in understanding some of the things that are occuring in the speech plots.

The vowels are produced by quasi-periodic glottal excitation. The manner in which the cross-sectional area varies along the vocal tract determines the resonant frequencies (formats) and therefore the sound (Ref 10). The vowels are /IY/, (beet); /I/, (bit); /E/, (bet); /AE/, (bat); /UH/ (but); /A/, (hot); /OW/, (bought); /U/, (foot); /OO/, (boot); and /ER/, (bird). An example of a vowel is the "OO" in "TO" in figure IV-4.5.1.

The "th" in "the" figure IV-4.1.5 is a voiced fricative. The voiced fricatives are /v/, /th/, /z/ and /zh/. A voiced fricative is produced by vocal cords that vibrate (one excitement source at glottis) and a vocal tract constriction at some point forward of the glottis. Thus producing turbulence in the neighborhood of the constriction. Unvoiced fricatives are essentially the same as voiced fricatives except the glottal excitation isn't present. The unvoiced fricatives include /f/, /θ/, /s/ and /sh/. The "s" in "rust" is an unvoiced fricative, however, it was not very pronounced in the original speech file. For a better example of an unvoiced fricative see the "s" in "six" in

figure IV-17.4 .

The nasals are produced with glottal excitation and the vocal tract completely constricted at some point along the oral passage. With a lowered velum, air flows through the nasal tract with sound being radiated at the nostrils. An example of the nasal sounds (including /m/, /n/, /ŋ/) can be found in figure IV-4.7.4 with "n" in the word "new".

The first semivowels sound vowel-like and are characterized by a gliding transition in vocal tract area function between adjacent phonemes. Therefore, the acoustic characteristics of these sounds are influenced by the context in which they occur. The semivowels include /w/, /l/, /r/, and /y/. Although the "r" in "to rust" is only directly connected to the word "rust" in figure IV-4.5.3 it is clear from the plot that the "r" performs a transition.

Diphthongs are generally considered to be a gliding monosyllabic speech that starts at the articulatory position for one vowel and then moves toward the other vowels position. Diphthongs include /eI/, (bay); /oU/, (boat); /aI/, (buy); /aU/, (how); /oI/, (boy); and /ju/, (you).

Voiced stop consonants /b/, /d/, and /g/ are produced by building up pressure behind a total constriction somewhere in the oral tract, then suddenly releasing the pressure. These transient, noncontinuant sounds are dynamic in nature and have characteristics that are highly influenced by the vowel which follows the stop constant. For an example of a voiced stop see figure IV-4.3.2 containing "b" in the word

"began".

Unvoiced stop consonants are similar to their voiced counter parts except that the vocal cords don't vibrate. The unvoiced stop consonants include /p/, /t/, and /k/. An example is the "p" in "pipe" in figure IV-4.2.2 .

The final phonemes to be mentioned are the affricates. The affricates (including /t / and /j/) can be modelled as the concatenation of the unvoiced stop /t/ and the fricative /ʃ/.

Figure IV-5 is a plot of synthesized speech from the RLPC vocoder. In general terms the best way to determine the quality of speech output from the vocoder is to listen to it. Unfortunately, as stated before, the quality of good speech is difficult to determine. From the informal listening tests it was determined that the speech in figure IV-5 sounded very much like the speech in figure IV-4. Just a cursory comparison between the figures will show some similarities and differences. The first similarity is in the pitch period for voiced speech. As would be expected if the pitch detector works, voiced speech will consist of damped sinusoids at the proper pitch period. An example of this can be seen in figures IV-4.4.1 and IV-5.4.1. There are more similarities but what makes the speech different is probably more interesting.

Probably the most apparent difference is the gain adjustments. For example, in figures IV-4.3.2-300 and IV-5.3.2-300 the maximum magnitude in the original speech

varies slowly whereas the synthesized speech is constant, then increases and then decreases. Also related to the gain is how the relative magnitude varies over the whole speech file. Some words seem to receive more emphasis than others. One final difference is lack of finer details on the synthesized speech waveform. However this is to be expected since only ten poles are used in the synthesis system.

In addition to the plots of the original and synthesized speech a series of 3-dimensional (3-D) plots are shown in figure IV-7. Figure IV-6 is a set of labeled axis that are representative of all the axis in figure IV-7 and other 3-D plots. The log magnitude response versus frequency is found from a set of filter (predictor) coefficients. A series (time sequential) of these plots are combined to produce a log magnitude versus frequency versus time plot. The vertical axis (height) is the log magnitude and although scale information isn't available viewing the general trend as time varies is the primary function of the plots. The horizontal axis (width) is the frequency (0-4KHZ) and the diagonal axis (depth) is the time axis representing a time span of 512 samples or 6.4 msec unless otherwise stated. A table cross referencing the speech plots and formant plots is given in Table IV-1 (see page IV-24) For details on creating these plots see appendix C.

The orientation of the axis varies for different plots because this author did not have control over axis orientation (though the correspondence to frequency etc. has

not changed). To aid in comparing the log magnitude plots with the synthesized speech the same numbering convention used on the speech plots will be used even though each log magnitude figure will only have one figure per page. For example, figure IV-7.3.3 is a log magnitude plot of synthesized speech in figure IV-4.3.3. In addition to the figure number a decimal point then a letter "a", "b" or "c" will be appended. The letter will indicate the time interval between subsequent log magnitude versus frequency plots. A letter "a" indicates a separation of 10 samples or 1.25 msec. Letter "b" is used for an 80 sample separation or 10 msec. Finally, "c" is for a 160 sample separation or 20 msec and only occurs in one figure. In a few figures, for example (figure IV-7.3.2.a-300) a dash with a number will follow. The "-number" is the point at which the 3-D plot begins. This was done to provide a clearer plot of vocal information in cases where a considerable portion of the speech plot is silence. Note: most frames of silence do not have a corresponding 3-D plot therefore some figure sequences will be missing.

One potential problem that occurs with the 3-D plots will be discussed. If one looks at a formant as it varies with time it will be noted that it often seems to periodically increase and decrease. For an example of this see figure IV-20.6 then look at the lowest frequency formant. Note how as time passes it almost looks like a low frequency sinusoid. From peak to peak it is seven formant

plots apart where each formant plot is separated by 10 sample points. Thus each peak is about 70 samples apart. By looking at the pitch plot in figure IV-25 (a) it can be seen that the pitch period is roughly 70. By observing other plots a similar correspondence can be noted. One possible explanation is that when the pitch impulse occurs more energy is in the system and this causes the log magnitude of the formant to be larger. There may be other explanations of what causes this, but the problem this causes is the same whatever the origin. The 3-D formant plots are useful for comparative purposes. However, if the time between formant plots (log magnitude versus frequency) is not small compared to the pitch period subsequent formant plots may be in peaks and valleys causing a jagged looking plot. See figure IV-21 for an example of this jagged look then look at figure IV-20 which is also a 3-D plot of "FIVE". Unfortunately figure IV-20 requires more plots but there is no question whether the jaggedness is inherent in the information or if it is just a matter of the location formant plots that were picked.

It is worthwhile to look at some of the various phonemes and observe what happens in the 3-D plots. First an example of a vowel will be examined at. Figure IV-7.5.1 and IV-7.5.2 are an example of the vowel /oo/ in the word "to". From the 3-D plot it is clear that three distinct formant exist and they vary little through the whole utterance which is what is expected of vowels.

The next phoneme is the semivowel which as stated earlier are vowel-like with a gliding transition between adjacent phonemes. A nice example of the semivowel /r/ is shown in figures IV-7.5.3 through IV-7.5.5. From these figures the vowel like formant structure can be seen. Also the formant structure shifts from figure to figure hence the "glide".

Another interesting phoneme in figure IV-7.1.5 is the voiced fricative /th/. Three main formants exist and they vary in intensity through the word "the" which is not surprising if one listens to the word as it is said.

The additional phonemes will not be discussed because phoneme analysis is not the purpose of this thesis. However, enough examples have been provided to demonstrate possible uses of the output plots.

Some additional speech and 3-D plots of the numbers zero through ten are included in this thesis because they provide more examples of this systems potential outputs. The numbers were chosen because of their frequent use.

The speech file of numbers was created as three separate files. The first file includes utterances of the numbers "one", "two", "three" and "four". The second speech file contains the numbers "five", "six", "seven" and "eight". The third speech file contains the numbers "zero", "nine" and "ten". A plot of the first speech file is shown in figure IV-11 and the vocoder output is shown in figure IV-12. A plot of the pitch is shown in figure IV-17 (a).

Also, 3-D plots of "one" thru "four" are shown in figures IV-13 thru IV-16 respectively. Figure IV-18 is the original plot of speech file containing "five" thru "nine" and figure IV-19 is plot of the vocoder output for this file. Figure IV-20 is the 3-D plot of the number "five" where the plots are log magnitude versus frequency seperated by 10 data points or 1.25 msec. The format of the supplementary figure numbers in figure IV-8 is the same as described earlier.

Figures IV-21 thru IV-23 are formant versus time plots of the numbers "five" thru "eight" respectively. Figure IV-25 (a) is a plot of the pitch period for the numbers "five" thru "eight". Figure IV-26 is the original speech plot of "zero...nine...ten" and figure IV-27 is the vocoder output plot. Figures IV-28 thru IV-30 are the formant versus time plots respectively. The pitch period plot of this last file is shown in figure IV-31. Figure IV-32.3 are formant versus time plot of a portion of the word "five". What makes these plots different is that the formant plots (log magnitude versus frequency) are only one sample point apart or .125 msec. The three plots are time sequential with the first plot starting at point 138 in figure IV-19.1.2 . Also figure IV-32 can be compared with figure IV-20.2 where the first figure starts roughly 14 formant plots in. Probably the most pertinent observation is that there isn't a great deal of additional information to be gained (in this case) by observing formant plots each sample.

DATA RATE

Although the vocoder system created for this thesis is not a real-time system it is interesting to determine what the data rate of this system is and what it could be if it were real-time.

The primary reason for using an LPC vocoding system is to take advantage of the low bit rate for transmitting speech information. The vocoder system consists of a transmitter, channel, and receiver. If an error free channel is assumed channel encoding is not required. The transmitter performs RLPC analysis and pitch detection then this data is sent across the channel. The receiver uses the channel data to synthesize speech.

For this system, integers require 16 bits and floating point numbers require 32 bits. To determine the data rate for this system the worst case will be assumed. This occurs when pitch information and predictor coefficients must be sent. Pitch information is transmitted once every 80 samples where 16 bits are used for pitch and 16 bits are used as a voiced/unvoiced switch. For a system of order 10 this is 320 bits each time predictor coefficients are sent. Also 32 bits are used for the gain. If the predictor coefficients are updated every 1.25 msec this is a data rate of 284,800 bits/sec. Updating every 10 msec results in a data rate of 38,400 bits/sec. Of course this system would never be implemented transferring this much data (primarily because most of the data doesn't contain useful data). What

follows is a description of the data rate of a RLPC vocoder that could be implemented.

For an order p RLPC analysis system a set of p predictor coefficients, a gain parameter, a voiced/unvoiced parameter, and the pitch period must be sent across the channel. The voiced/unvoiced switch requires 1 bit; 6 bit quantization of the pitch period is adequate; and the gain uses about 5 bits if distributed on a logarithmic scale (Ref 1). This is a total of 12 bits each time pitch information is transmitted which for this system is once every 10 msec.

If one considers direct quantization of the predictor coefficients then 8-10 bits are required per coefficient to ensure stability of the predictor polynomial. This is the case because small changes in the predictor coefficients can lead to relatively large changes in the pole positions.

The obvious question is what appropriate parameter set can be used for coding and transmission. Two possible parameter sets include using predictor polynomial roots or a set of reflection coefficients. Predictor polynomial roots can be easily quantized in such a way that the resulting polynomial is stable. This stability is assured because the predictor polynomial roots will be within the unit circle. With this approach 5 bits per root are adequate to preserve the quality of the synthesized speech (Ref 10). Unfortunately a processing penalty of time is incurred solving for the roots. For details of the method using a set of reflection coefficients see Ref 12.

Assuming the pole method is being used and the predictor polynomial is order 10 (which is what this vocoder used), 62 bits of information will be sent across the channel each 10 msec. This corresponds to 6200 bits/sec if the data is transmitted each 10 msec. This data rate of 6200 bits/sec is really a maximum data rate for sending information each 10 msec. First it must be realized that this data rate assumes voiced speech. If the speech is unvoiced pitch information isn't required which produces a data rate of 5600 bits/sec. If the speech is silence, data doesn't have to be sent (except for a 1 bit silence switch). Since speech is a combination of silence, voiced, and unvoiced segments the average data rate is lower than 6200 bits/sec.

There are other factors that can affect the bit rate. First, the bit rate can be lowered by only transmitting new information. For example, if the predictor coefficients don't change much there is no reason to re-transmit the data. In a similar manner the data rate can be increased by sending more information in places where the formant structure is changing quickly. So for a given data rate the quality of speech may be improved by being selective about which information is sent and when.

Noise

As in other sections this also will not be an in depth study on the effects of noise on the RLPC vocoding system. Basically three effects were observed. First, noise

(uniform density) was added to an original speech file. Secondly, the pitch period was plotted for some speech files with noise and thirdly formant plots were generated using the noise files. In this section when noise is mentioned it refers only to artificially added noise and not the noise in the speech file caused by the original recording.

No attempt was made to measure the effect of various noise levels on the vocoder output (such as various SNR's in db). This is left for another thesis. However, enough noise was added to the input speech file to degrade the quality of the vocoder output. The output speech had a "buzz" sound to it. The SNR was found for the numbers "zero" thru "ten" by finding the ratio of the speech signal squared without noise to the noise squared. This comes from the basic definition of signal to noise ratio. A plot of the speech file "five" plus noise is shown in figure IV-33 and it corresponds directly in time with figure IV-18 which is the original speech plot of file "five" without noise. The vocoder output of the noisy speech is shown in figure IV-34. See table IV-1 for figure numbers of the other numbers with noise.

The signal to noise ratios for all the numbers follows:

1) "zero"-	12.72db
2) "one"-	9.23db
3) "two"-	13.62db
4) "three"-	10.27db
5) "four"-	13.69db
6) "five"-	10.48db

7)	"six"-	9.31db
8)	"seven"-	7.72db
9)	"eight"-	7.39db
10)	"nine"-	8.34db
11)	"ten"-	8.47db

To determine if the choice of starting and stopping positions (for the SNR computation) was critical the beginning and end position for the word "one" was varied. First the beginning and end position was decreased by 10% of the original interval used for the SNR computation. The SNR was found to be 10.13 which is a .9db increase over the original. This process was repeated for a 20% decrease at the beginning and end. The resulting SNR was 1.71db greater than the original or 10.94db. These results indicate that the choice of starting and ending positions in the SNR computation do not produce drastic changes but that comparing signal to noise ratios for different utterance should be used only for rough comparisons.

In comparing the vocoder output in figure IV-34 with the noisy speech in figure IV-33 the most obvious difference seen is that the vocoder output is much smoother than the noisy speech. This is not surprising since LPC systems are only an approximation and with only an order of 10 it is harder to pick up the higher frequency components. Another difference to be noted between the plots is that occasionally the vocoder misses being implused since it "thinks" that the speech segment is unvoiced such as in figure IV-34.4 . Comparing the pitch period plots with and without noise (figures IV-25 (b) and IV-25 (a) respectively)

it can be seen that there are more unvoiced regions with noise than when noise isn't present. It is interesting to observe the differences between the vocoder outputs for the cases with and without noise as inputs. Figure IV-19 without input noise and figure IV-34 with input noise have subtle differences (excluding the voiced/unvoiced problem) which are caused by the formant structure.

Pitch period plots have been provided for the speech files containing the numbers. Where figure IV-17, IV-25, and IV-31 are pitch plots of the speech files "one...two...three...four", "five...six...seven...eight", and "zero...nine...ten" respectively. As stated earlier, without noise more of the pitch plot is voiced. The noise makes the pitch detector believe that the speech isn't voiced. This isn't too surprising since a synthesizer simulates unvoiced speech with noise.

The final item to be discussed is the effect of noise on the formant plots. Figures IV-35 thru IV-38 are 3-D formant plots of the utterances "one", "two", "three", and "four" respectively. The separation between formant plots is 10 data points or 10 msec. Figure IV-39 is a series of 3-D formant plots of the word "five" where the number after the decimal point refers to the corresponding speech plot in figure IV-20. The figure IV-35 thru IV-38 are interesting and it is clear that the noise adds some high frequency formants. Unfortunately due to the problem described earlier where the formant magnitude varies slowly it is

difficult to determine if the jaggedness of the plots is due to noise or picking highs and lows for a formant. Because of this figure IV-39 is more useful. By comparing this figure to figure IV-20 differences due to noise are much easier to evaluate. The differences do not appear to be drastic but generally the shape of the formant appear smoother and more high frequency formants exist in the case with noise (figure IV-39).

CONCLUSION

The recursive system is configured in such a manner that the predictor coefficients can be available as often as needed. As would be expected, updating the synthesizer more often improves the quality of speech. The greatest improvement in quality is noticed when the synthesizer filter is updated every 10 msec instead of 20 msec. Improvement is also noted when updating every 1.25 msec instead of 10 msec but the amount of improvement wasn't as large as the change from 20 msec to 10 msec.

The data rate of the RLPC vocoder is comparable to typical LPC systems. Judicious use of additional information available in the RLPC system may improve the quality of speech without a great impact on the data rate.

The RLPC system is affected by noise in both the pitch detector and the speech analyzer. The pitch detector often mistakes voiced speech for unvoiced speech and the shape of higher frequency portion of the formant plots change.

Table IV-1

This table contains a description, list, and cross reference of speech and formant plots. The formant plots are produced using the predictor coefficients found in the speech analysis. Therefore, only cross references between synthesized speech plots and formant plots exist. The number in parenthesis is the time separation in msec between formant plots. For a more detailed list see the List of Figures.

<u>Speech Figure</u>	<u>Utterance</u>	<u>Formant Figure</u>
IV-4	Original-"The pipe began to rust while new"	None
IV-5	Vocoded- "The pipe began to rust while new"	IV-7 (1.25, 10, or 20)
IV-8	Original-"Thieves who rob friends deserve jail"	None
IV-9	Vocoded- "Thieves who rob friends deserve jail"	None
IV-11	Original-"One...two...three...four"	None
IV-12	Vocoder- "One...two...three...four"	see below
IV-12.1	Vocoder- "One"	IV-13 (10)
IV-12.4	Vocoder- "Two"	IV-14 (10)
IV-12.6	Vocoder- "Three"	IV-15 (10)
IV-12.8	Vocoder- "Four"	IV-16 (10)
IV-18	Original-"Five...six...seven... eight"	None
IV-19	Vocoder- "Five...six...seven... eight"	see below
IV-19.1	Vocoder- "Five"	IV-20 (1.25)
IV-19.1	Vocoder- "Five"	IV-21 (10)
IV-19.1	Vocoder- "Five"	IV-32 (.125)
IV-19.3	Vocoder- "Six"	IV-22 (10)

Table IV-1 (cont'd)

IV-19.5	Vocoder- "Seven"	IV-23 (10)
IV-19.7	Vocoder- "Eight"	IV-24 (10)
IV-26	Original-"Zero...nine...ten"	None
IV-27	Vocoder- "Zero...nine...ten"	see below
IV-27.1	Vocoder- "Zero"	IV-28 (10)
IV-27.4	Vocoder- "Nine"	IV-29 (10)
IV-27.6	Vocoder- "Ten"	IV-30 (10)
IV-33	Original-"Five...six...seven... eight" plus noise	None
IV-34	Vocoder- "Five...six...seven... eight" plus noise	None
None	Vocoder- "One" plus noise	IV-35 (10)
None	Vocoder- "Two" plus noise	IV-36 (10)
None	Vocoder- "Three" plus noise	IV-37 (10)
None	Vocoder- "Four" plus noise	IV-38 (10)
IV-19	Vocoder- "Five" plus noise	IV-39 (1.25)

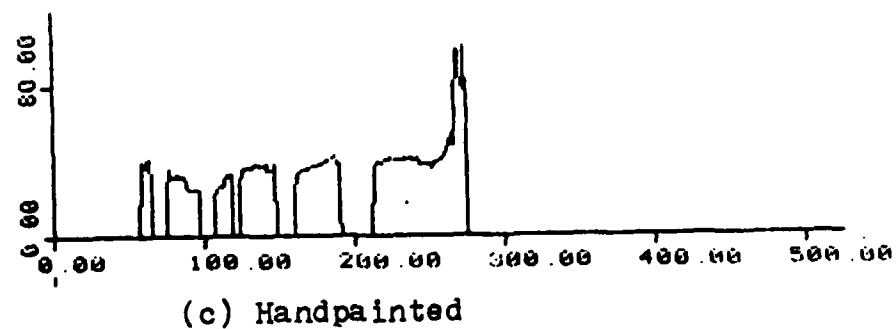
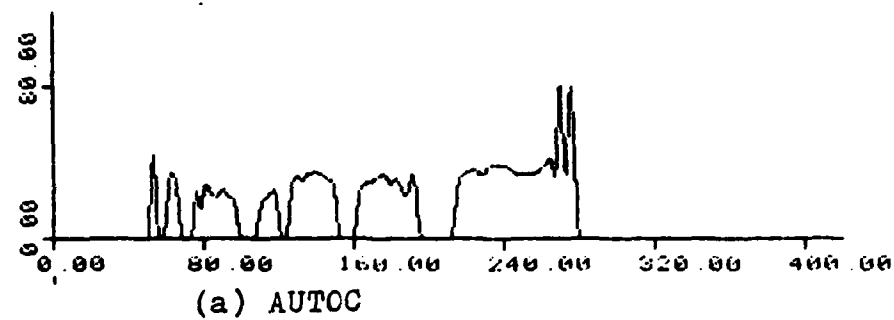


Figure IV-1 Pitch Period Plots of S1

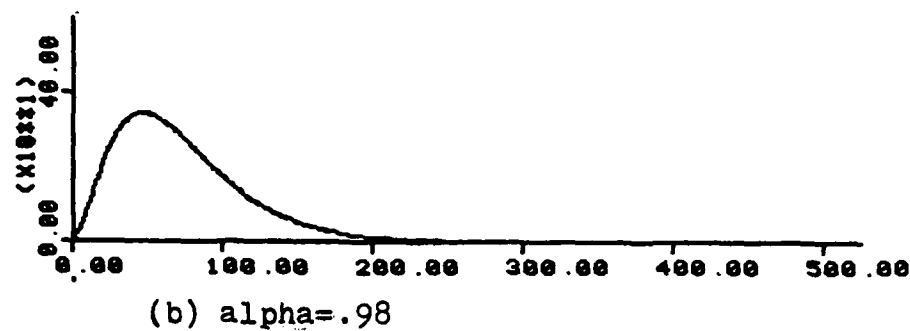
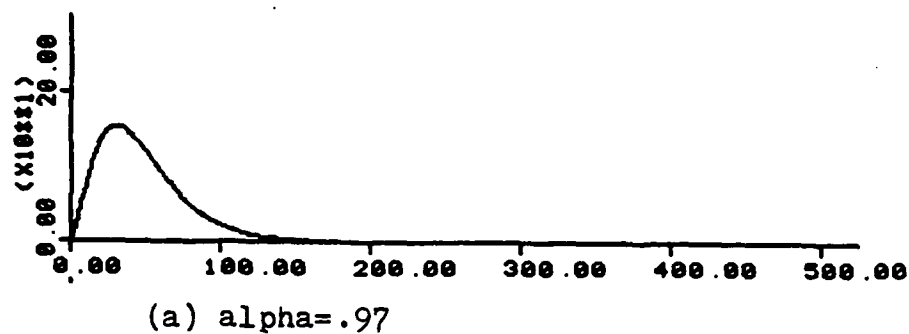
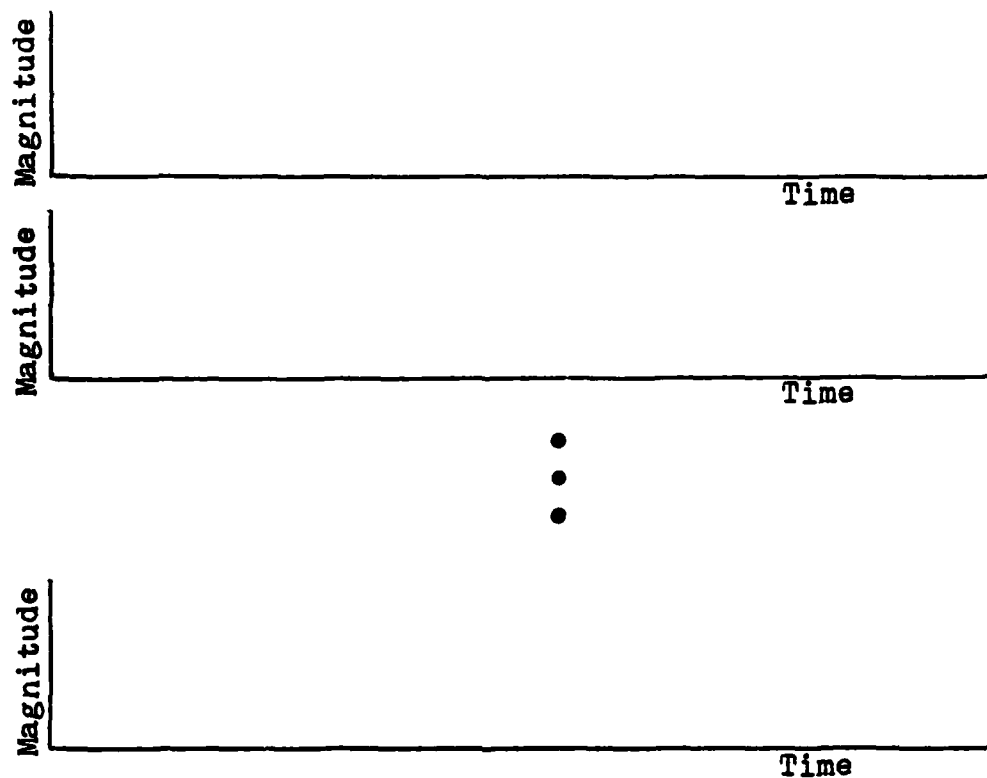


Figure IV-2 Impulse Response of Window



- Note: 1) Vertical axis- Magnitude of speech at sampled point
- 2) Horizontal axes- 512 sampled points (10msec)
- 3) Speech is plotted time sequentially from left to right then top to bottom

Figure IV-3 Skeleton Axis for Speech Plots

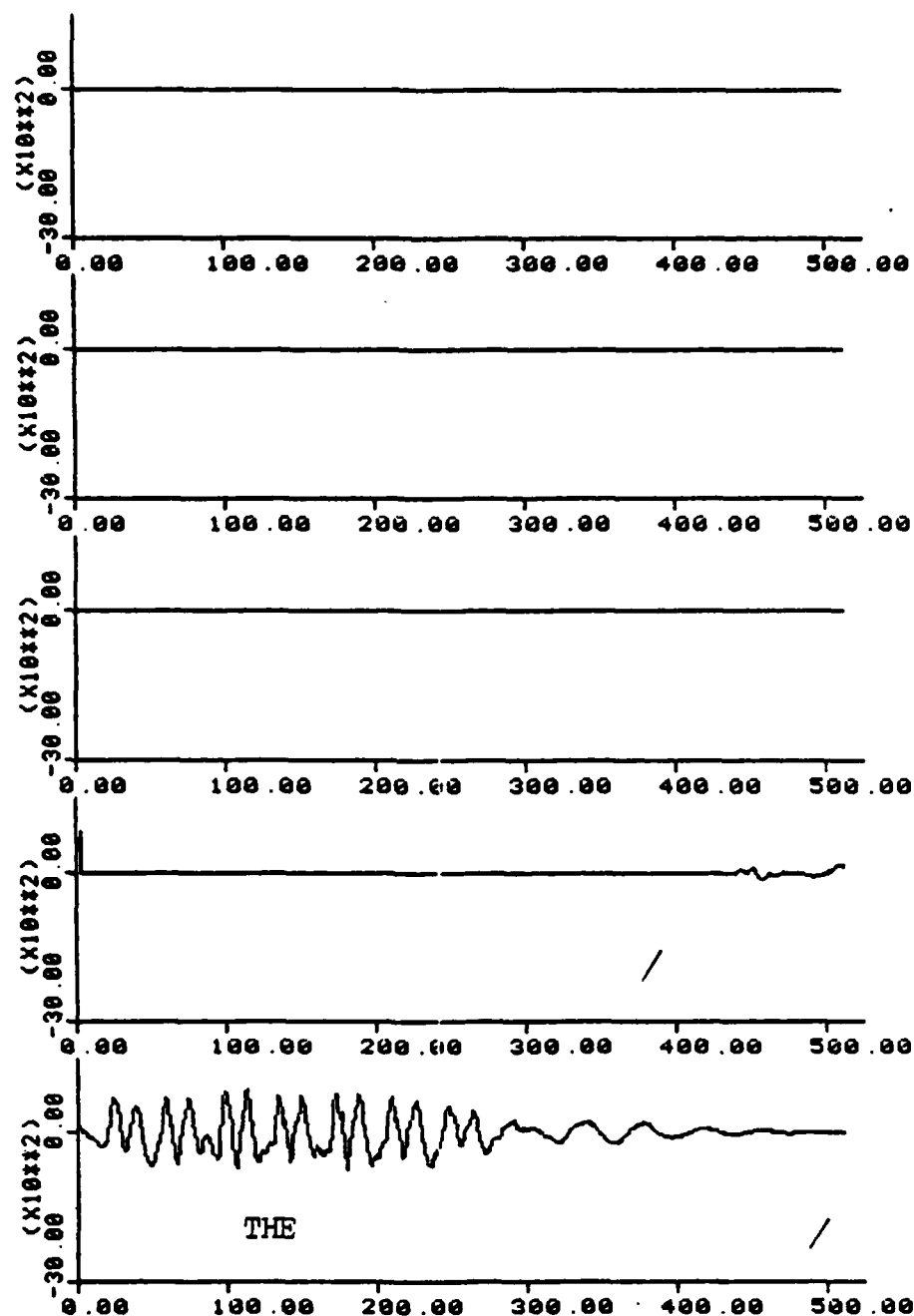


Figure IV-4.1 Original Speech File S1
"The pipe began to rust while new"

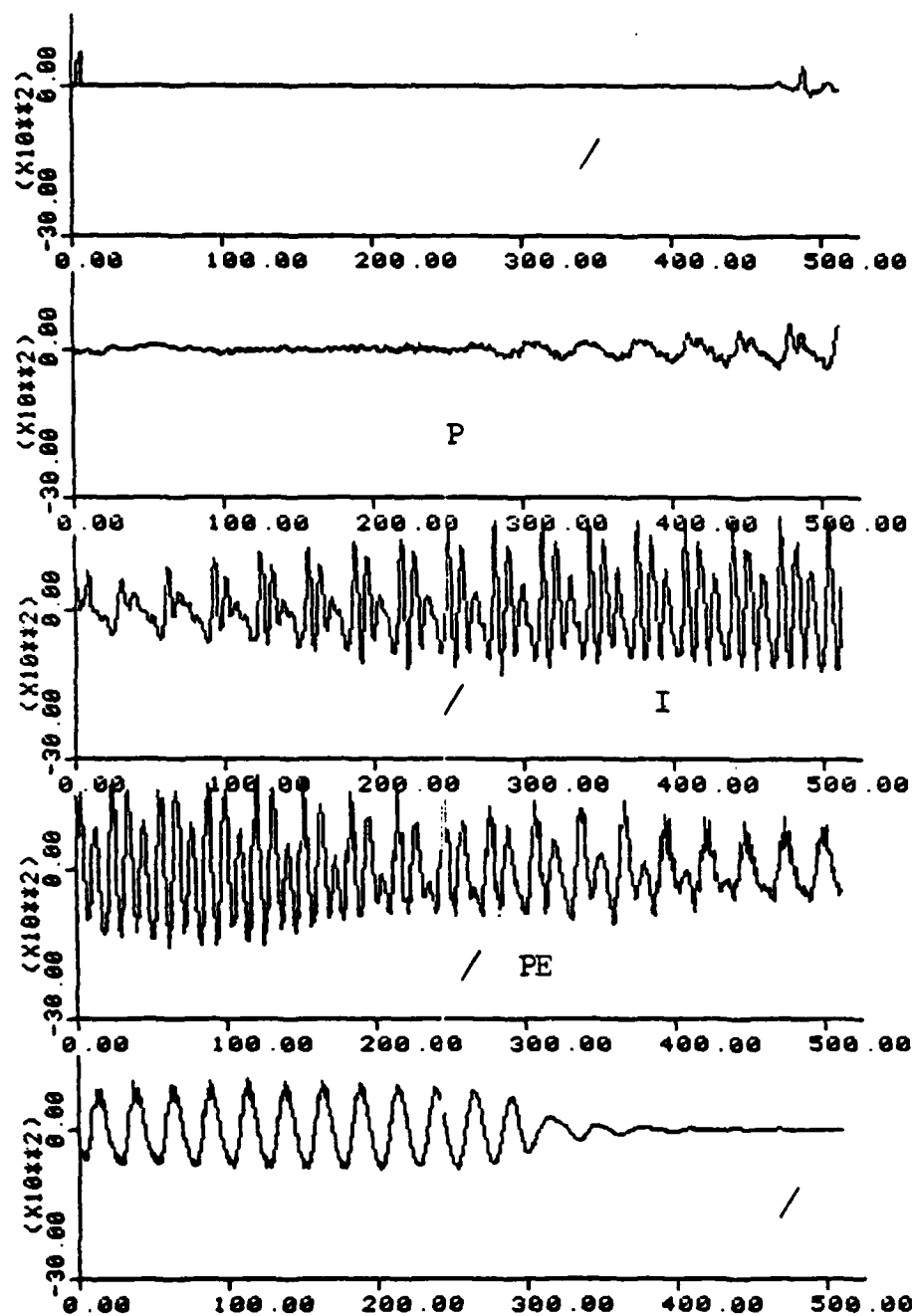


Figure IV-4.2 "The pipe began to rust while new"

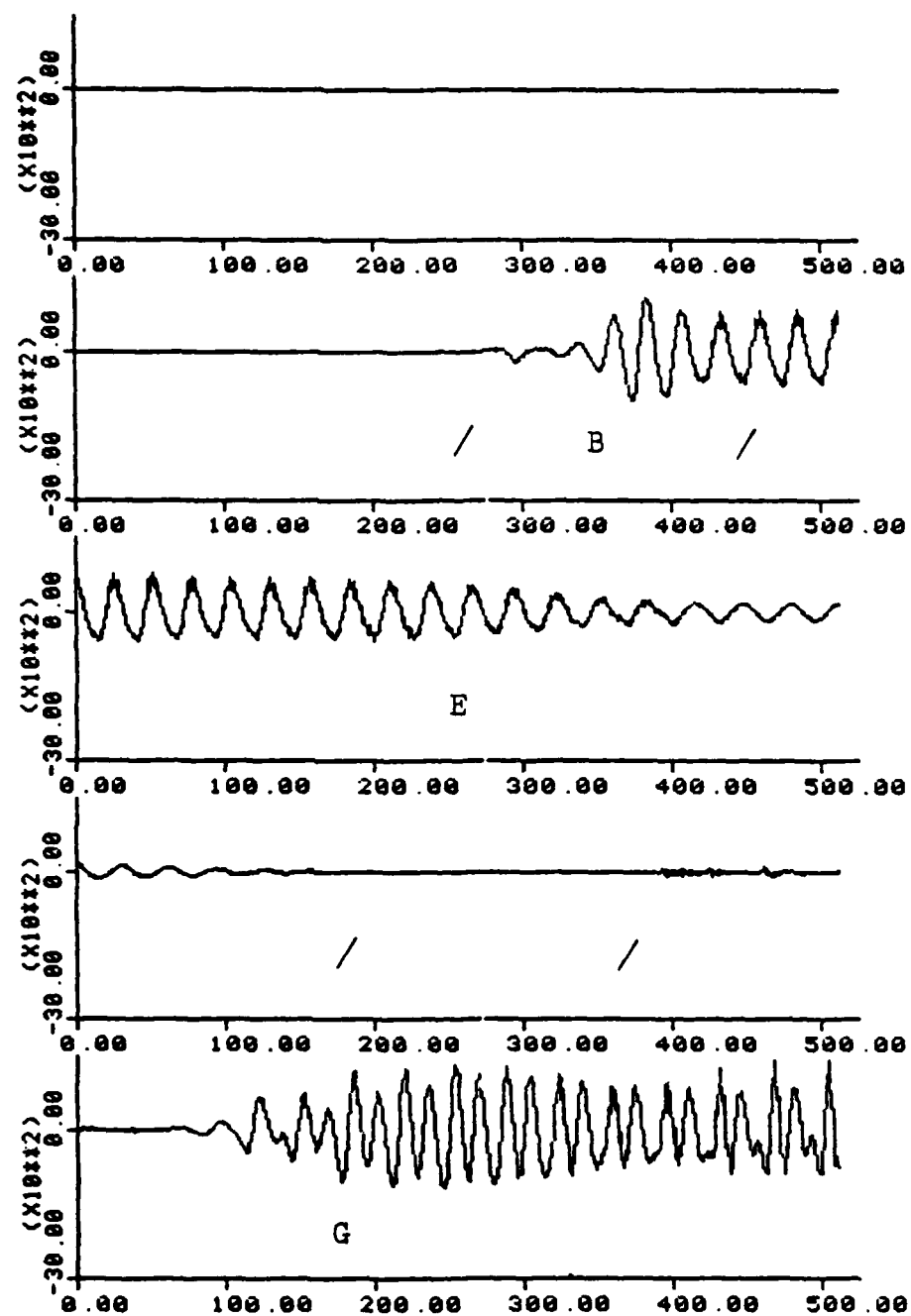


Figura IV-4.3 "The pipe began to rust while new"

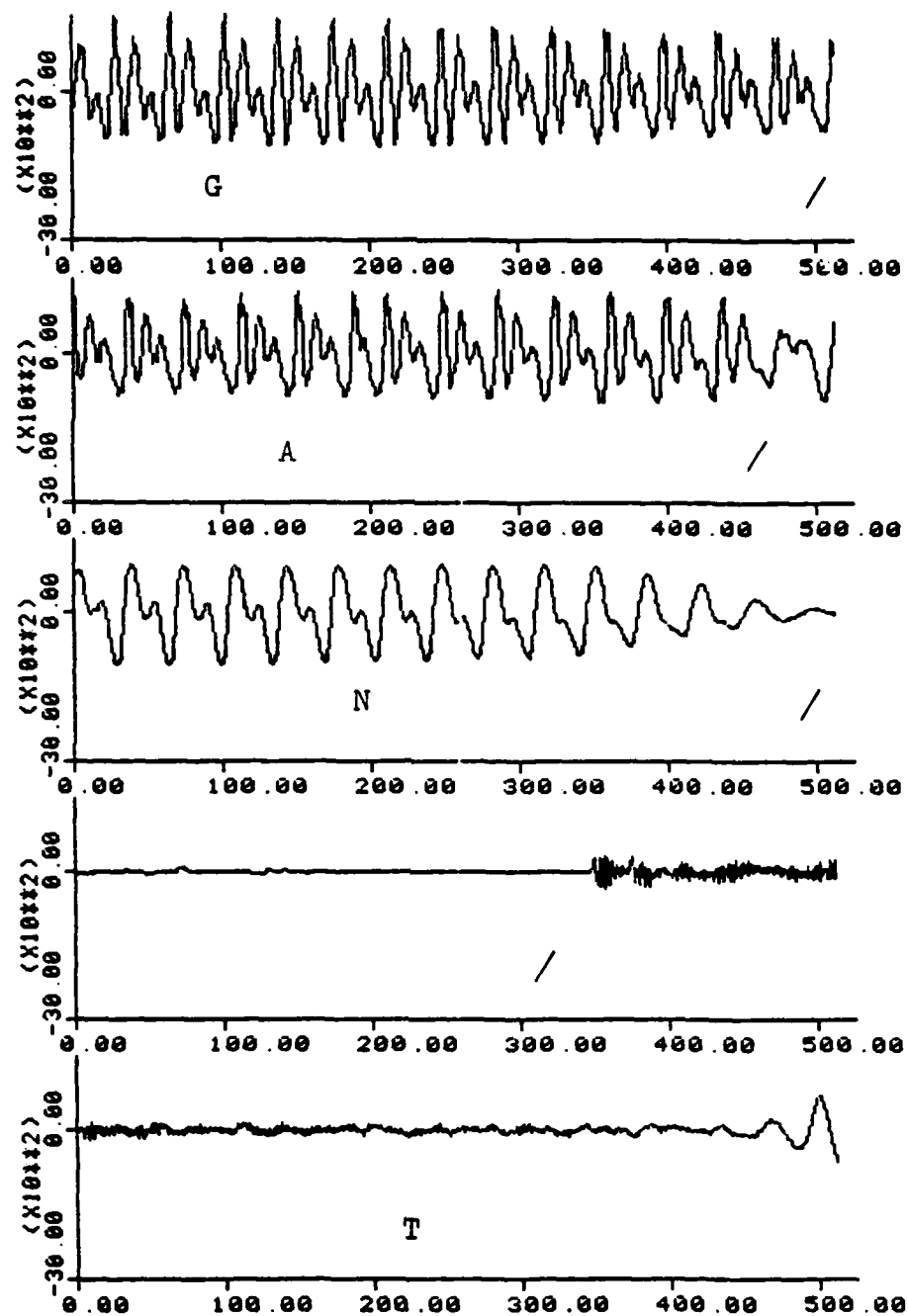


Figure IV-4.4 "The pipe began to rust while new"

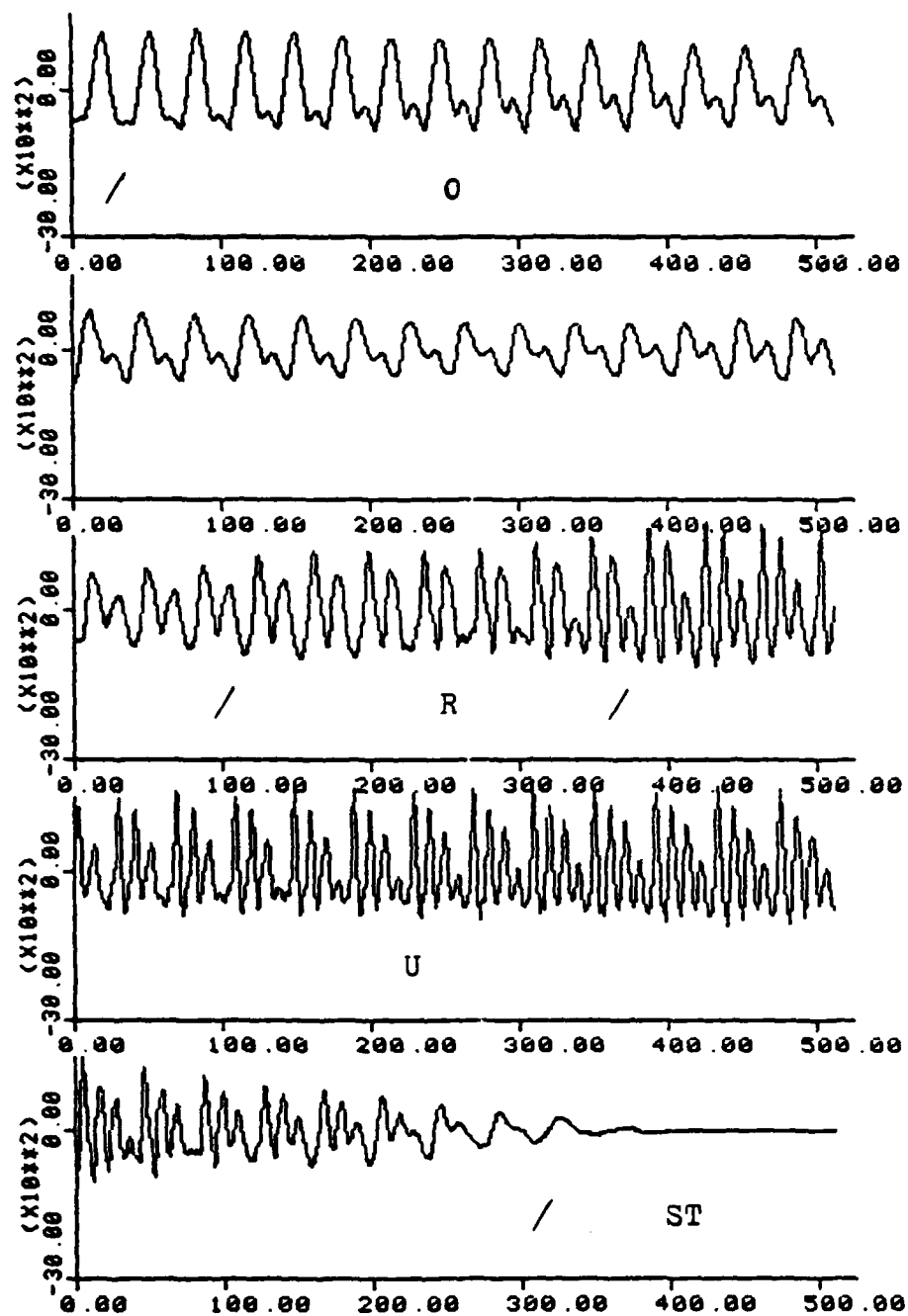


Figure IV-4.5 "The pipe began to rust while new"

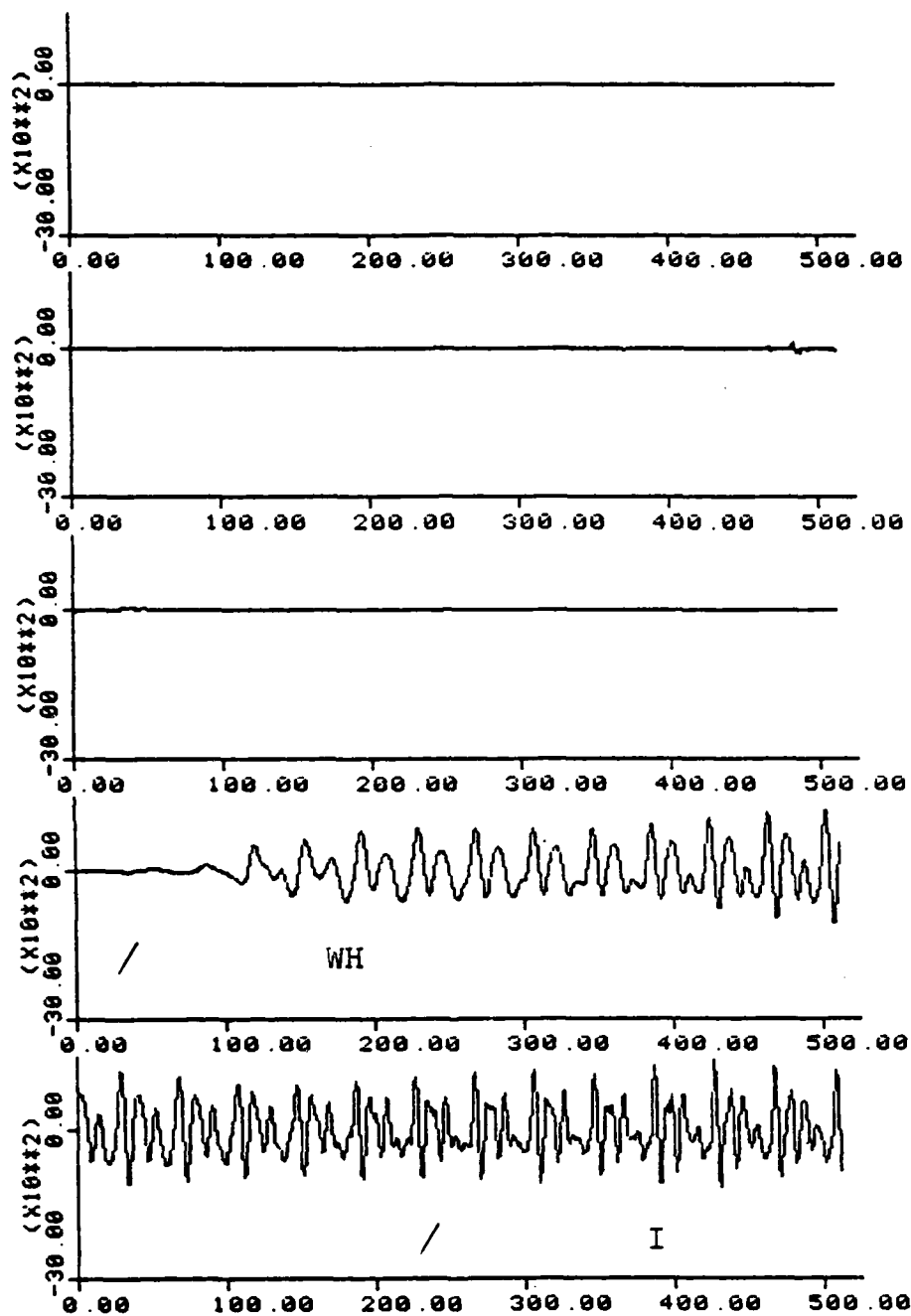


Figure IV-4.6 "The pipe began to rust while new"

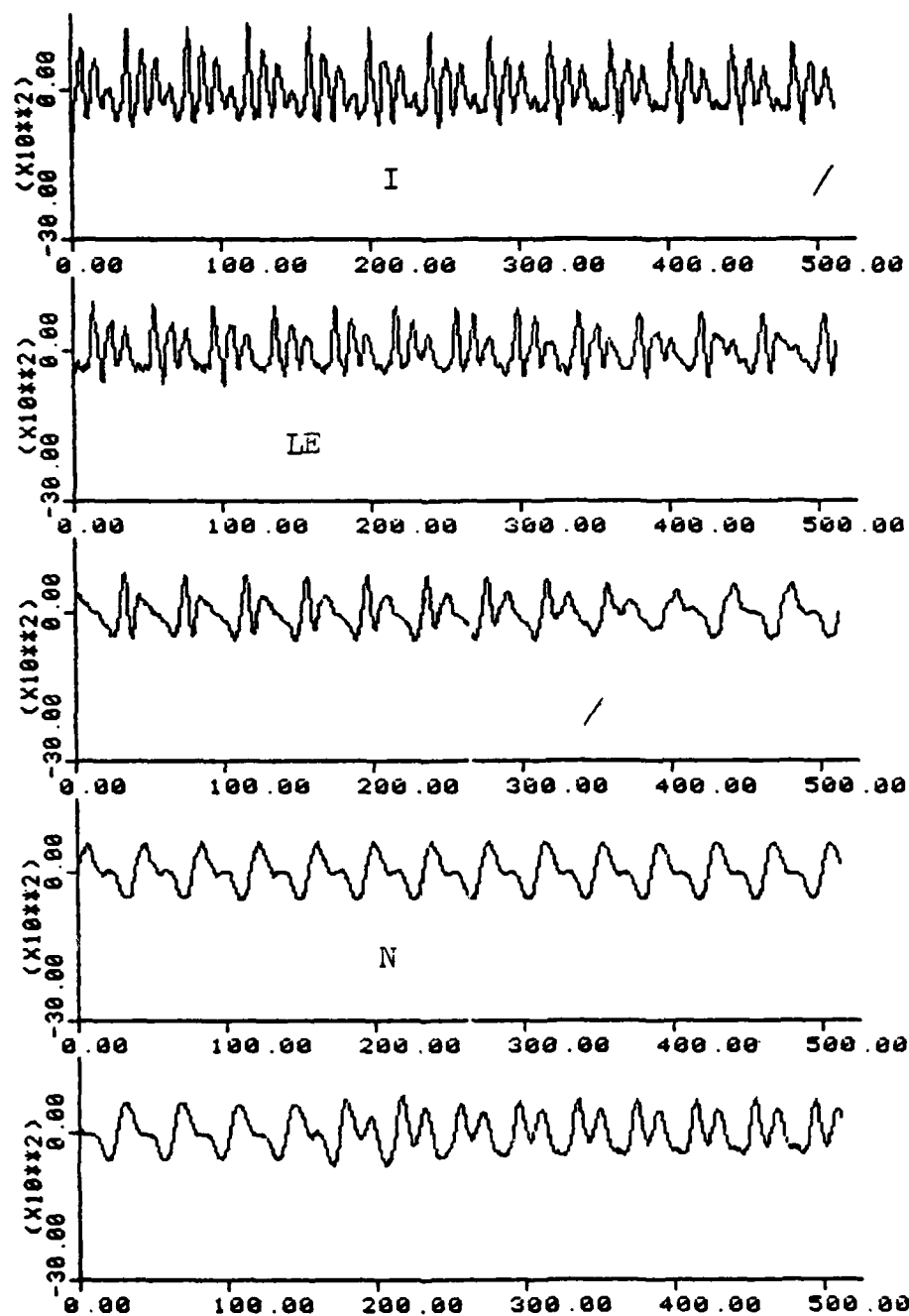


Figure IV-4.7 "The pipe began to rust while new".

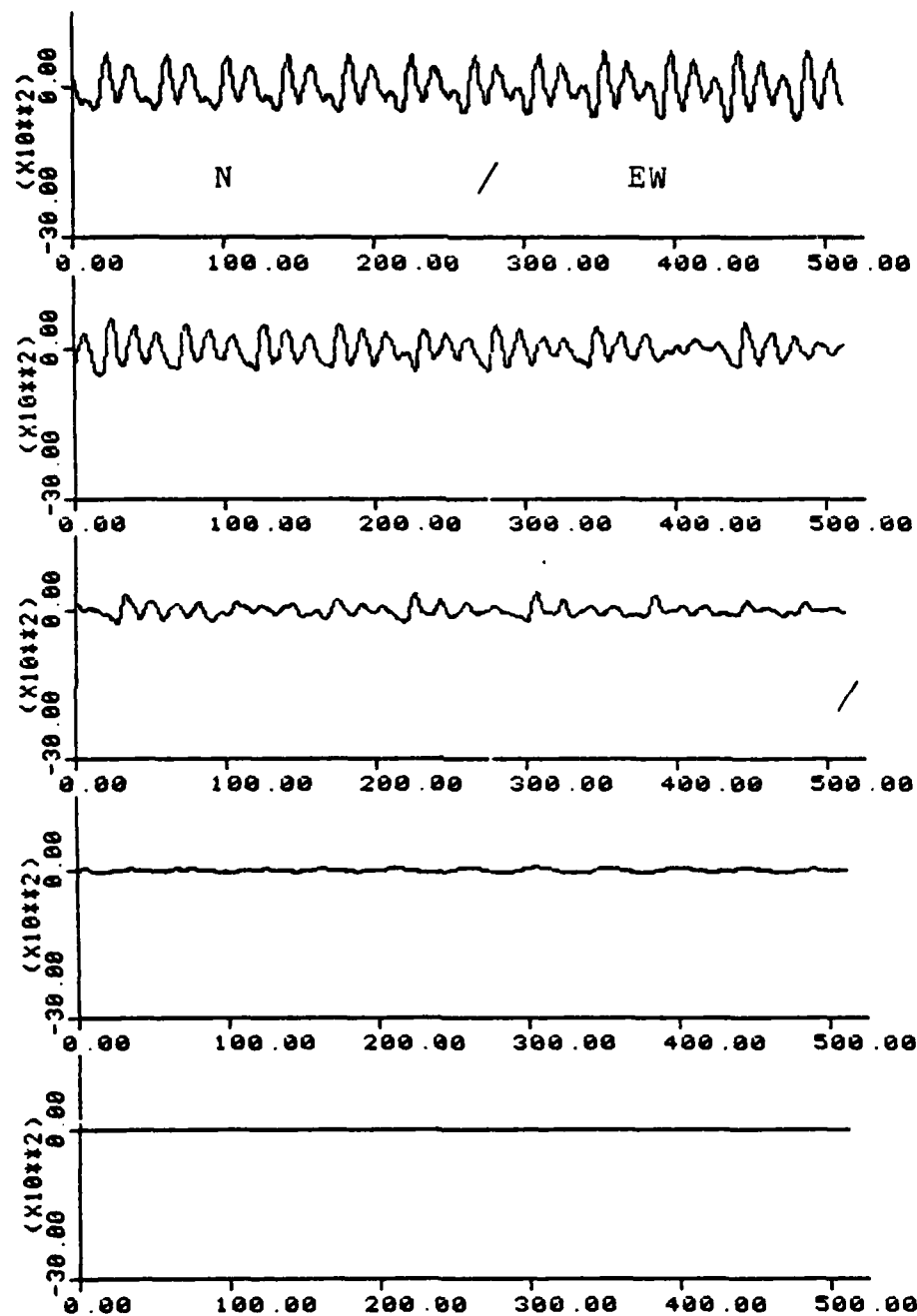


Figure IV-4.8 "The pipe began to rust while new"

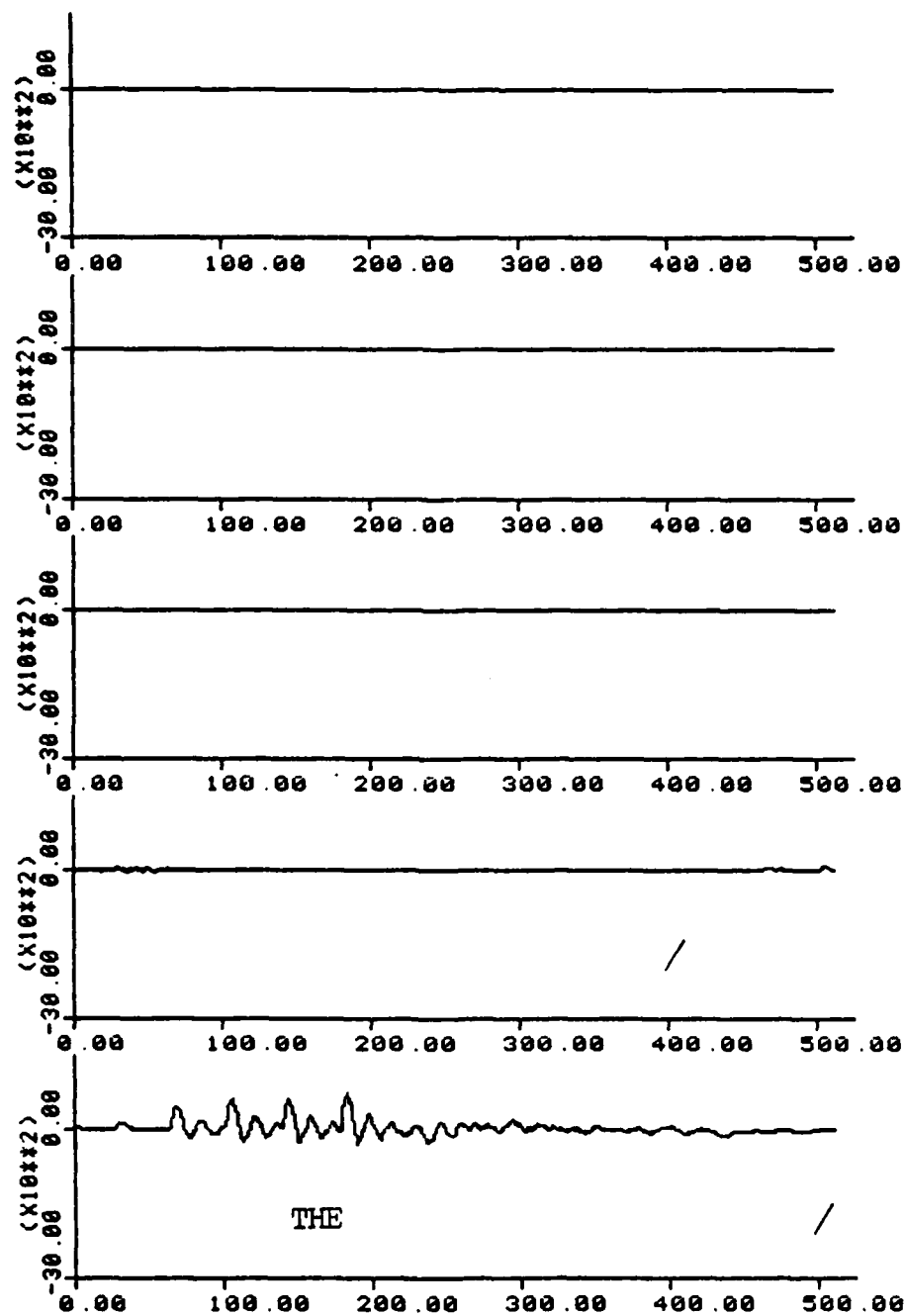


Figure IV-5.1 The Vocoder Output
"The pipe began to rust while new"

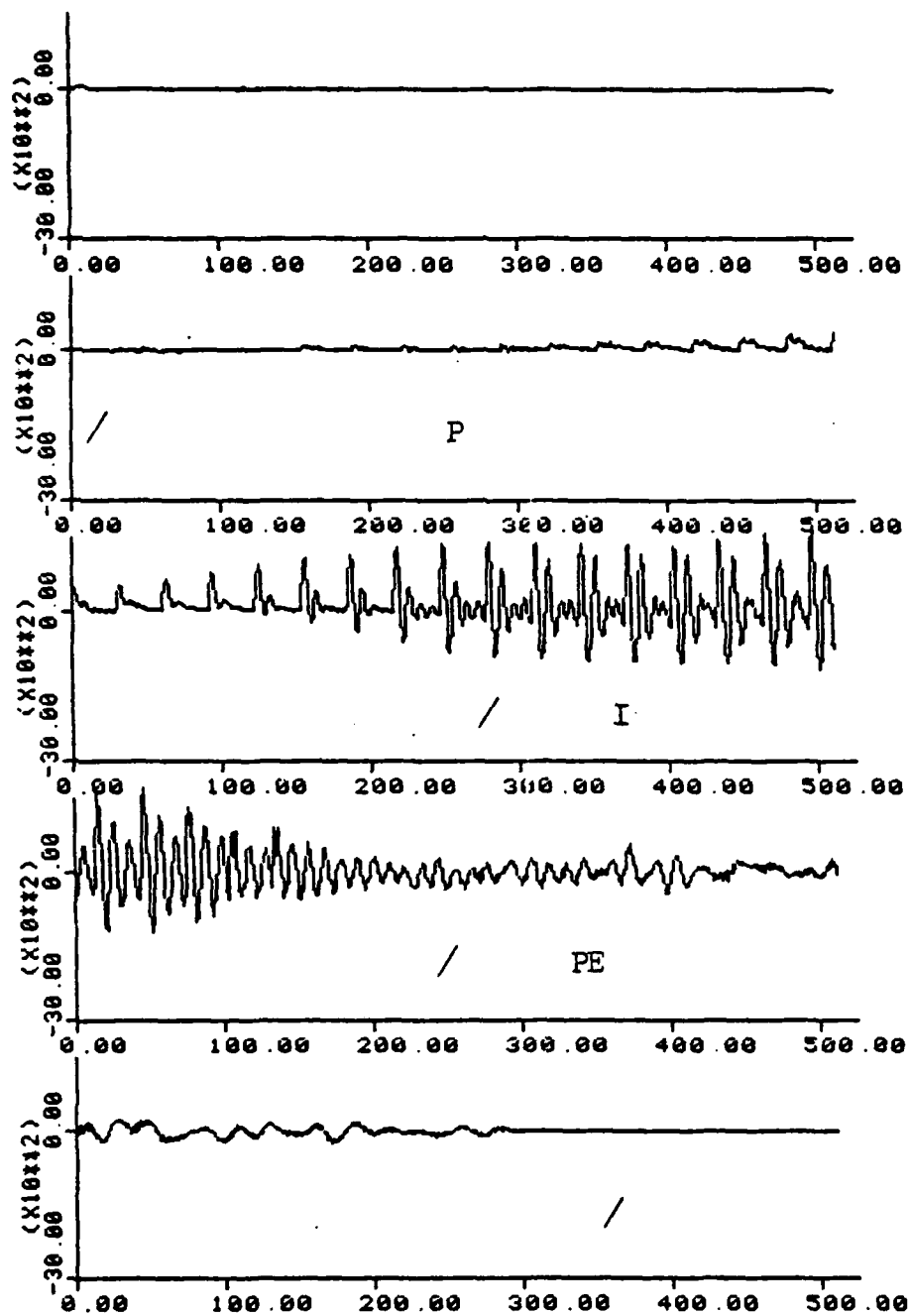


Figure IV-5.2 "The pipe began to rust while new"

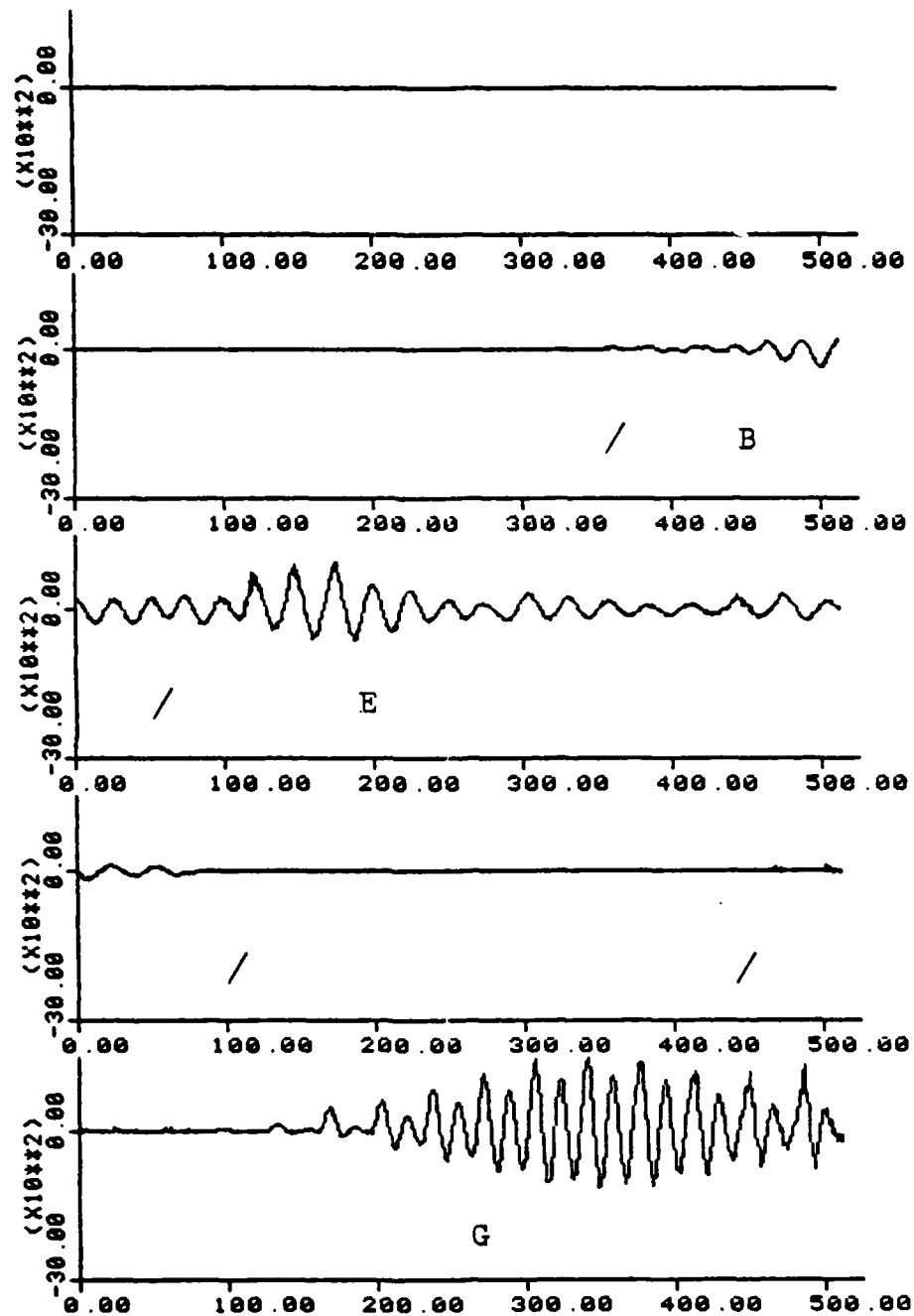


Figure IV-5.3 "The pipe began to rust while new"

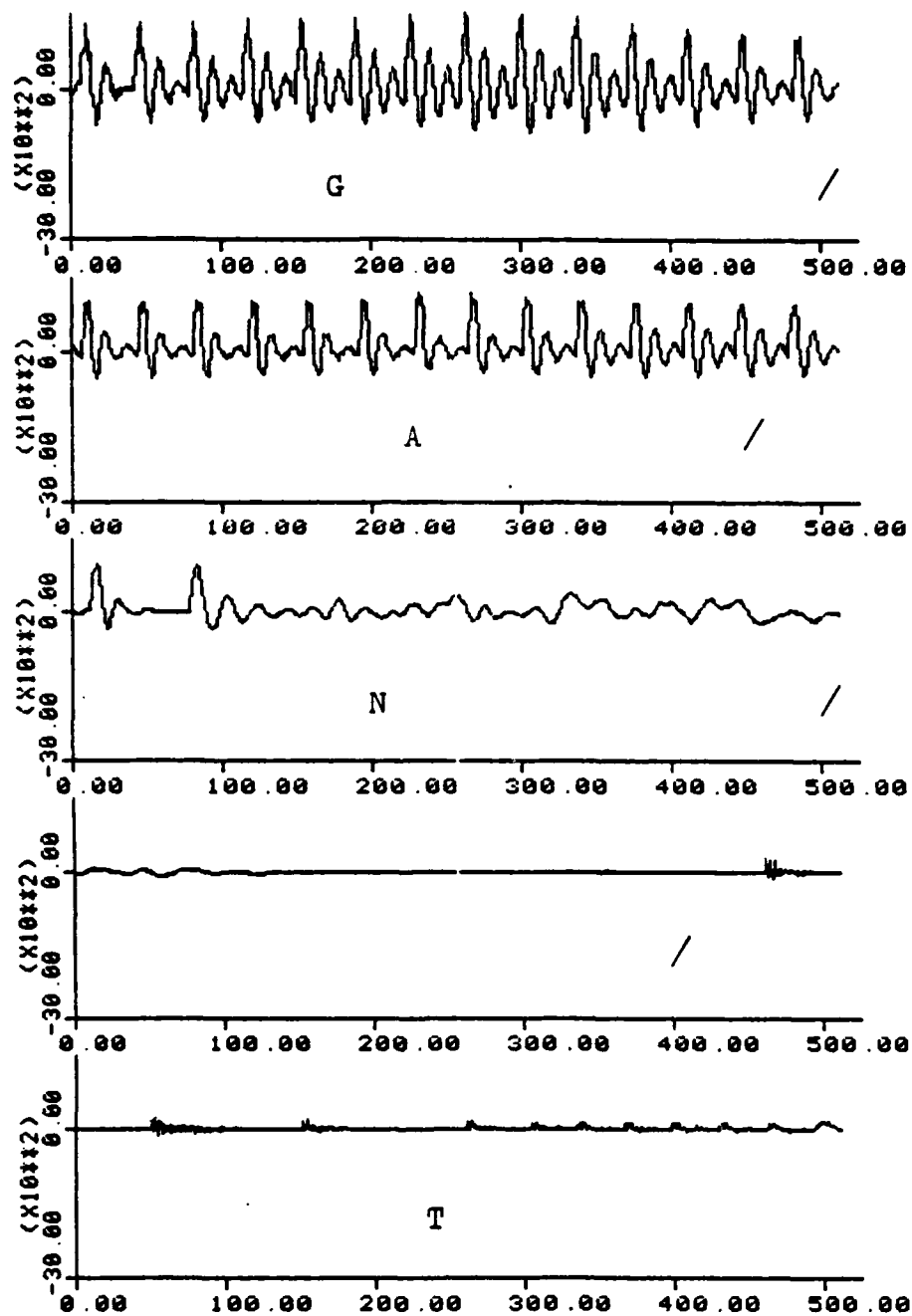


Figure IV-5.4 "The pipe began to rust while new"

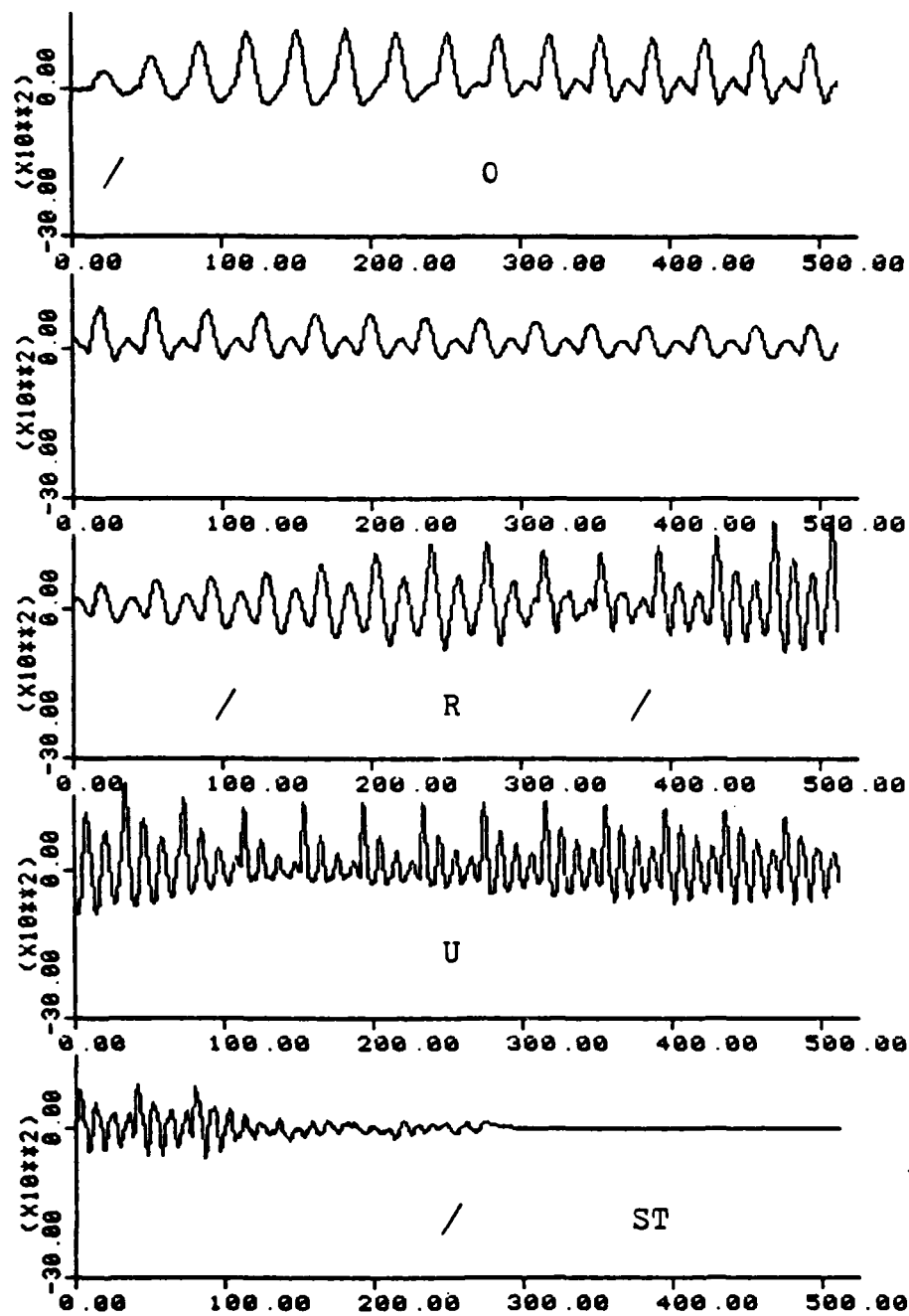


Figure IV-5.5 "The pipe began to rust while new"

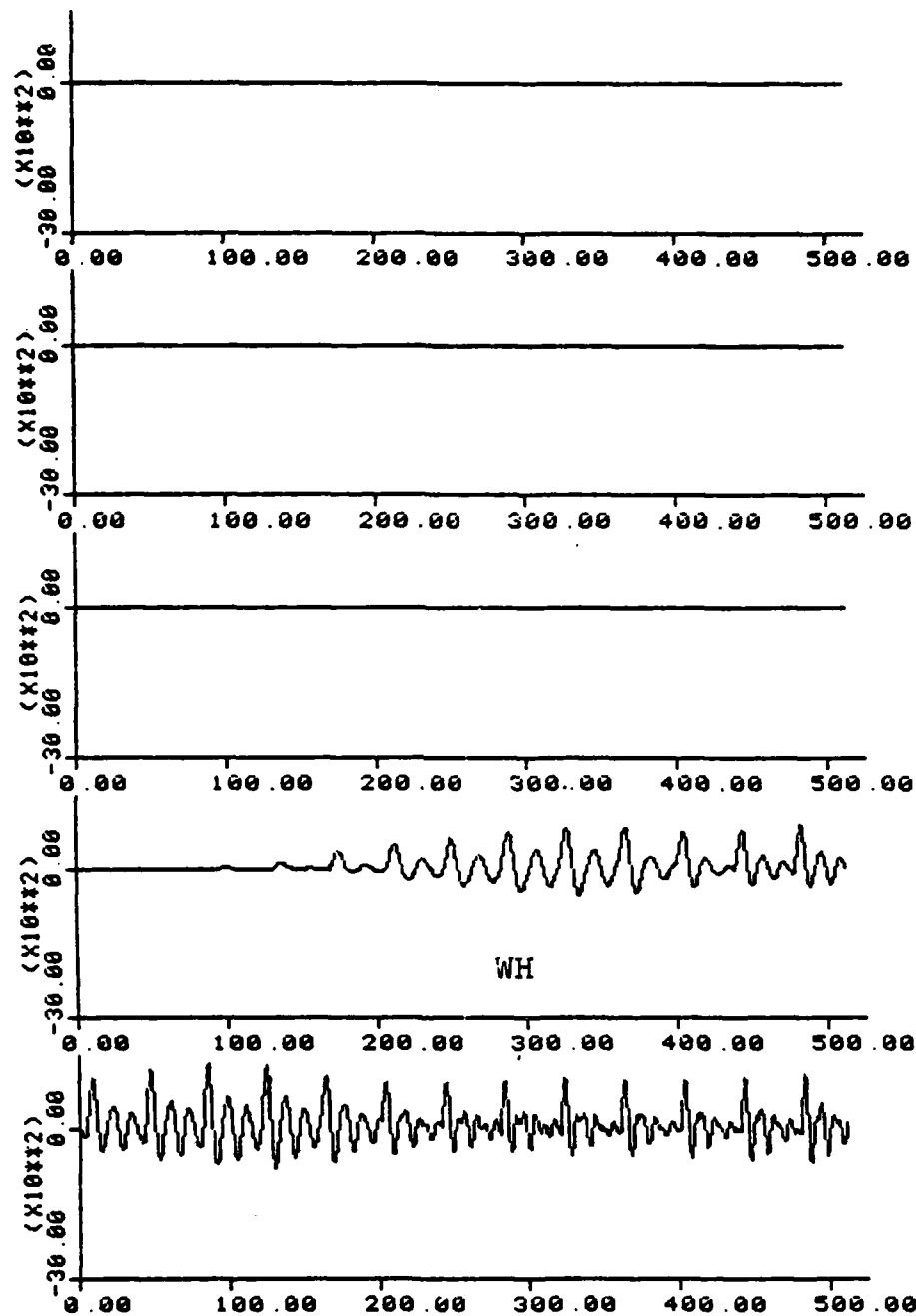


Figure IV-5.6 "The pipe began to rust while new"

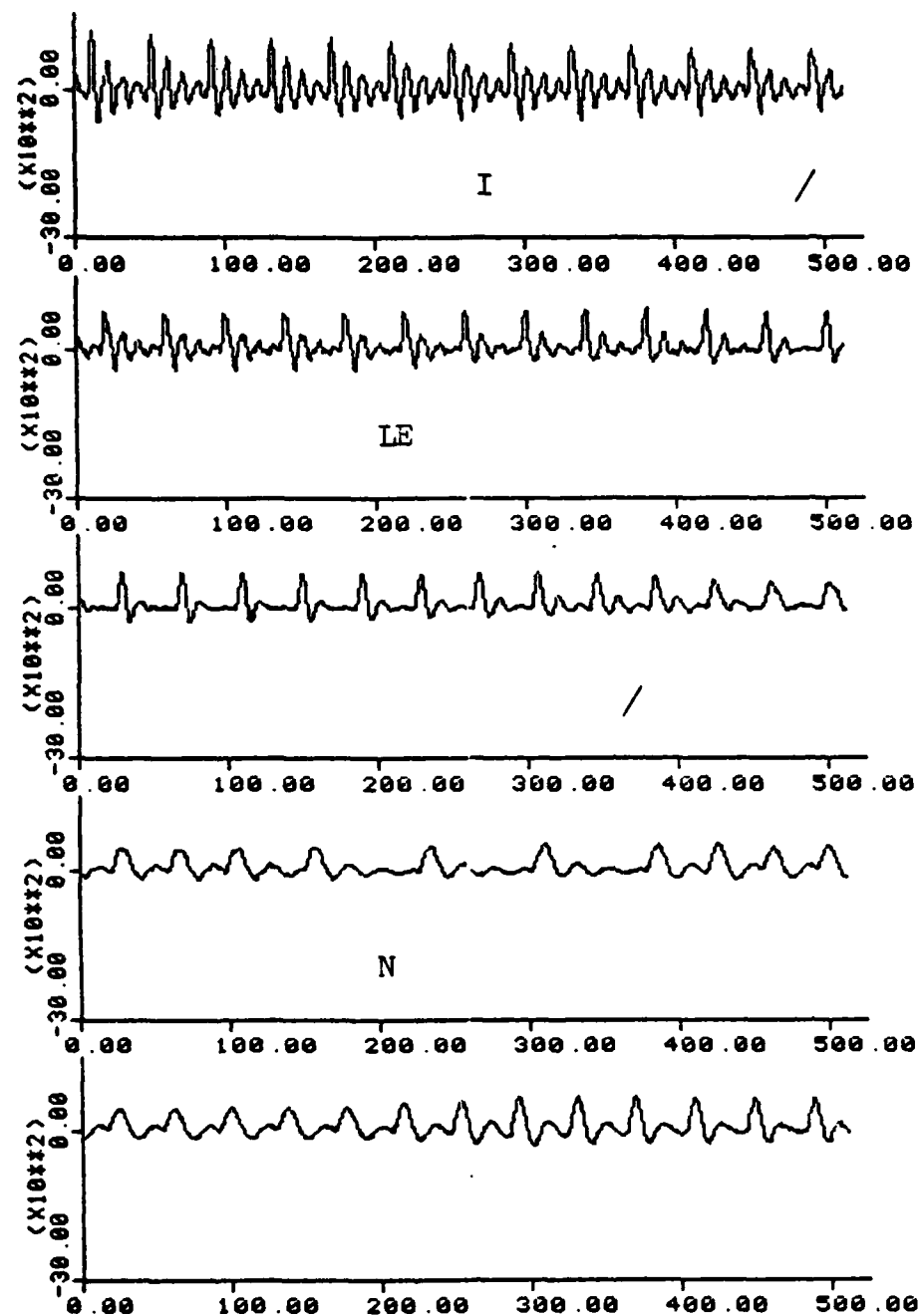


Figure IV-5.7 "The pipe began to rust while new"

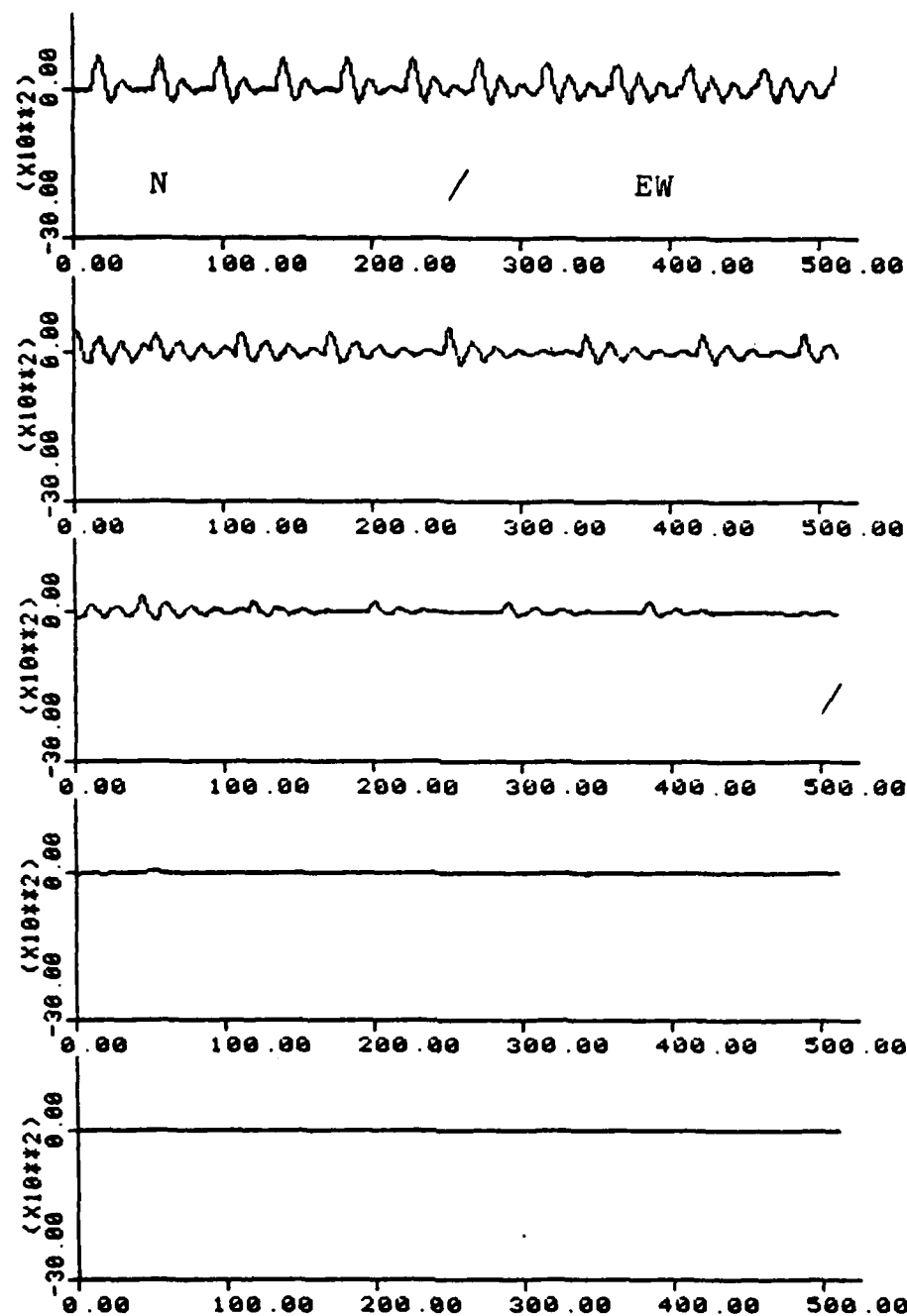
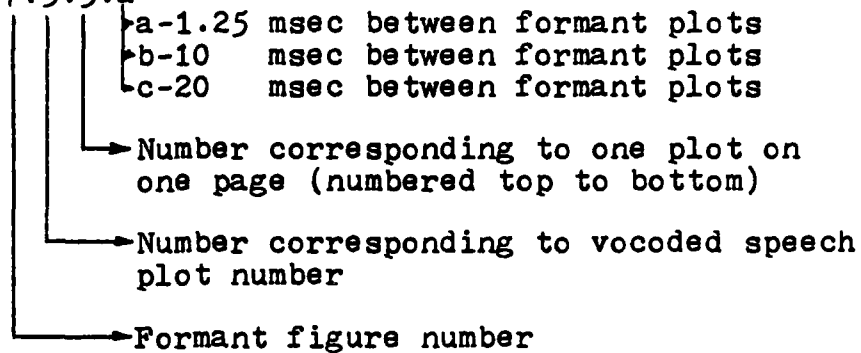


Figure IV-5.8 "The pipe began to rust while new"

Figure IV-7.3.3.a



Note: This example could correspond to the vocoder speech plot figure IV-5.3 the third plot down from the top of the page.

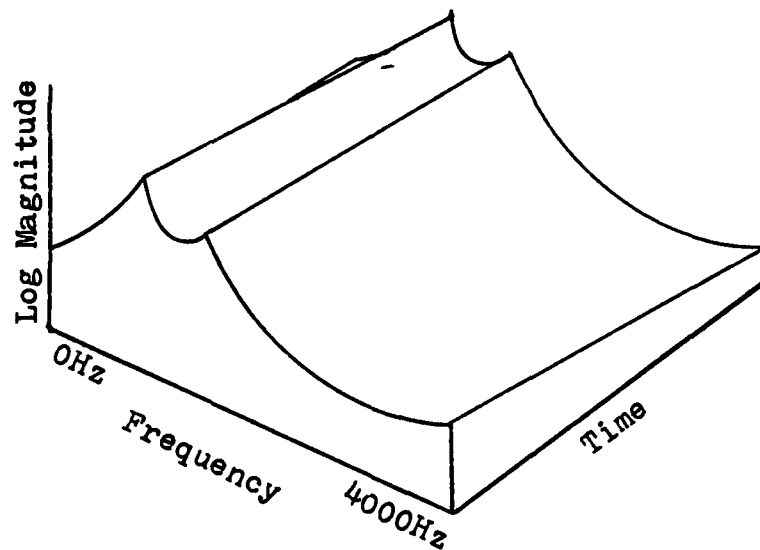


Figure IV-6 Skeleton Axis for Formant Plots

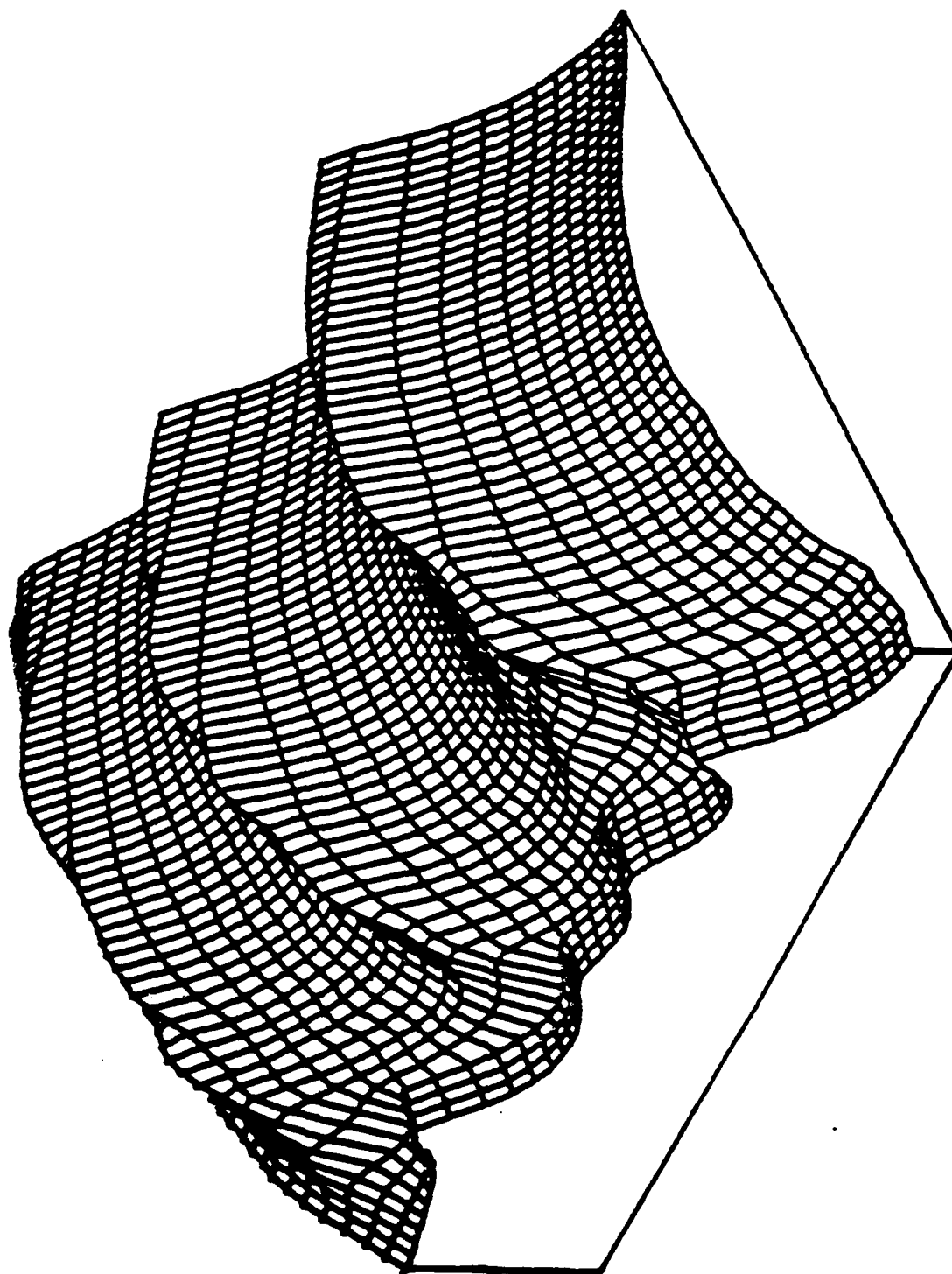


Figure IV-7.1.5.a "THE"

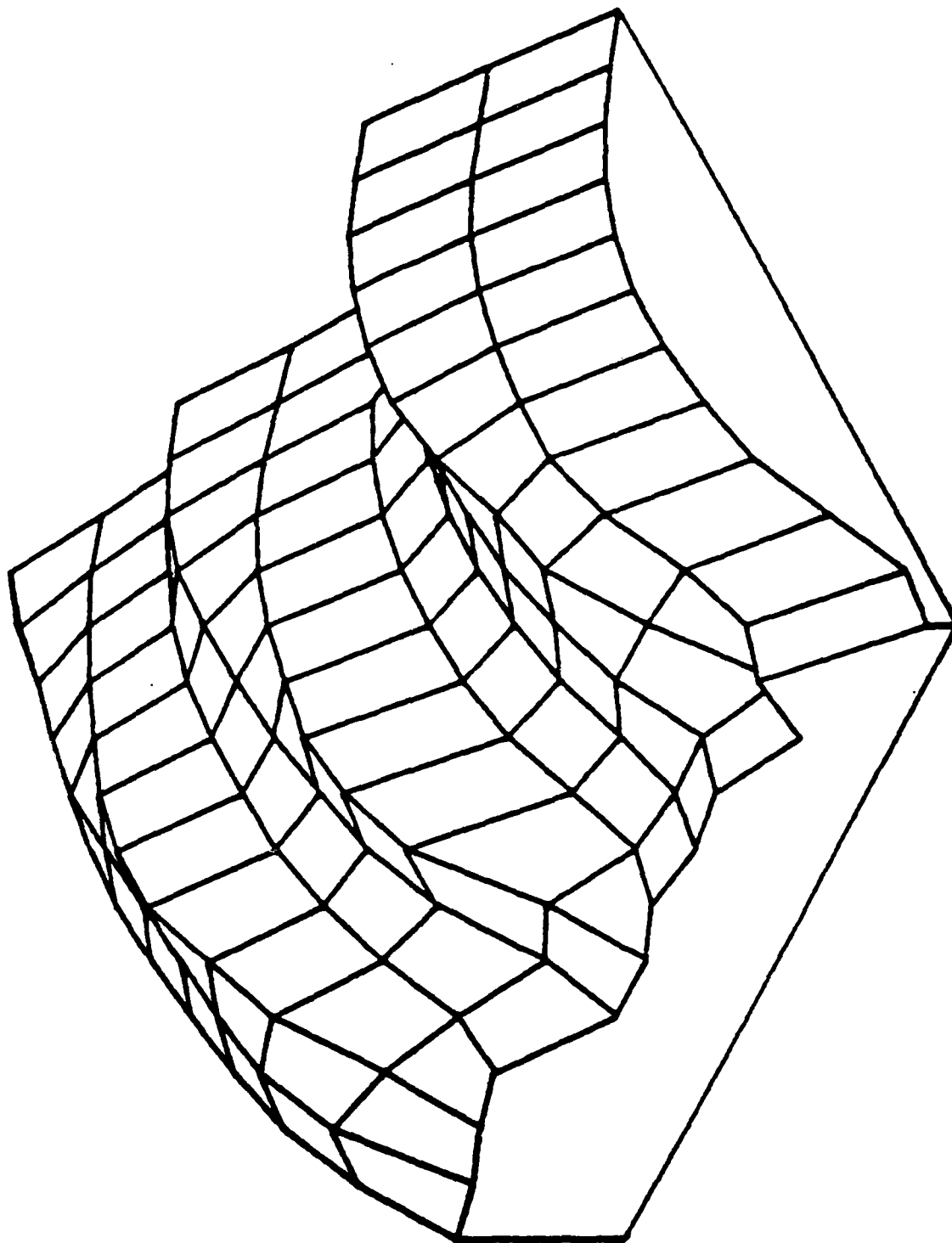


Figure IV-7.1.5.b "THE"

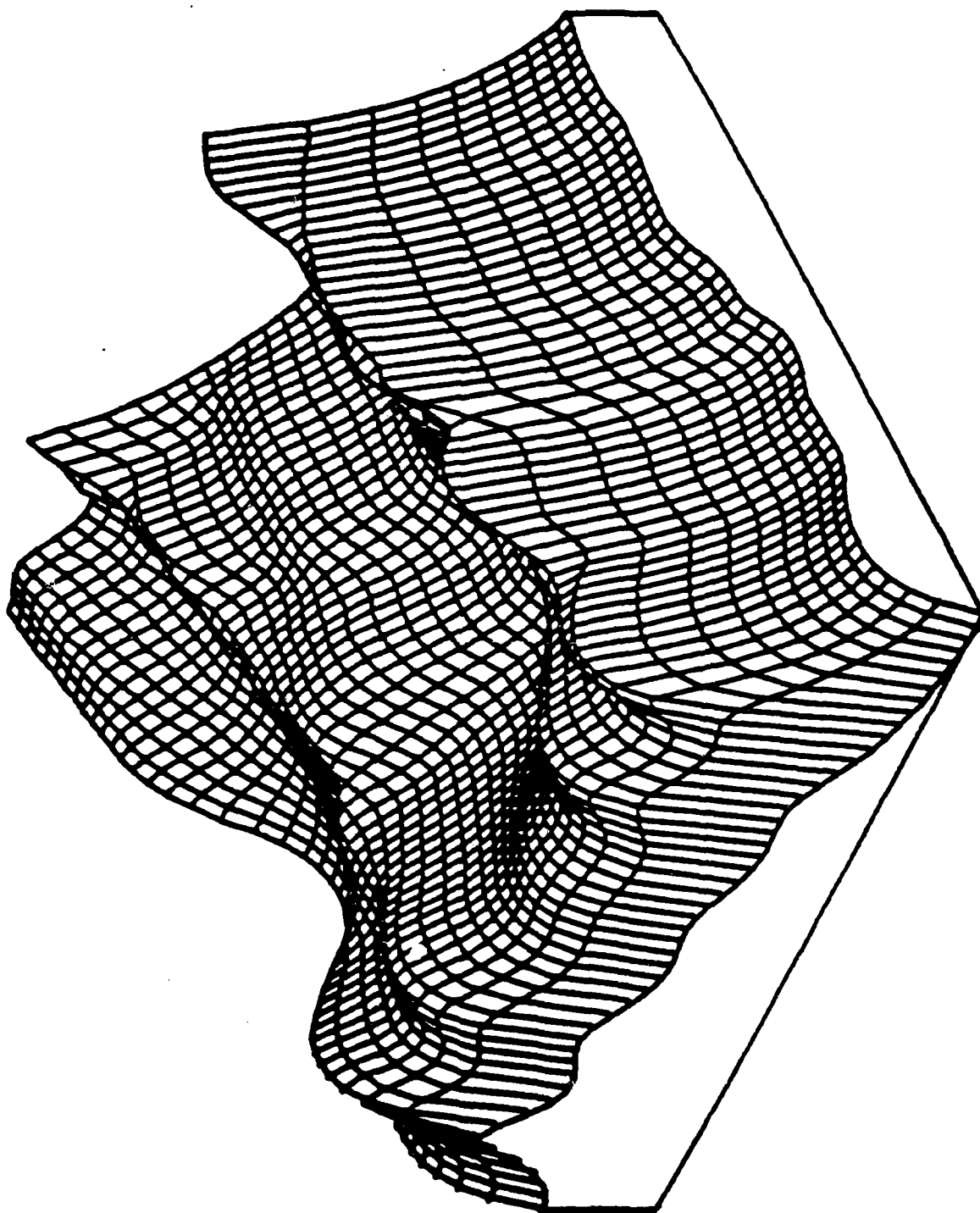


Figure IV-7.2.2.a

AD-A138 008

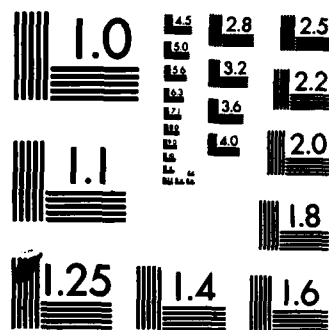
A RECURSIVE LINEAR PREDICTIVE VOCODER(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
W A JANSSEN DEC 83 AFIT/GE/EE/83D-33

2/4

UNCLASSIFIED

F/G 17/2

NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

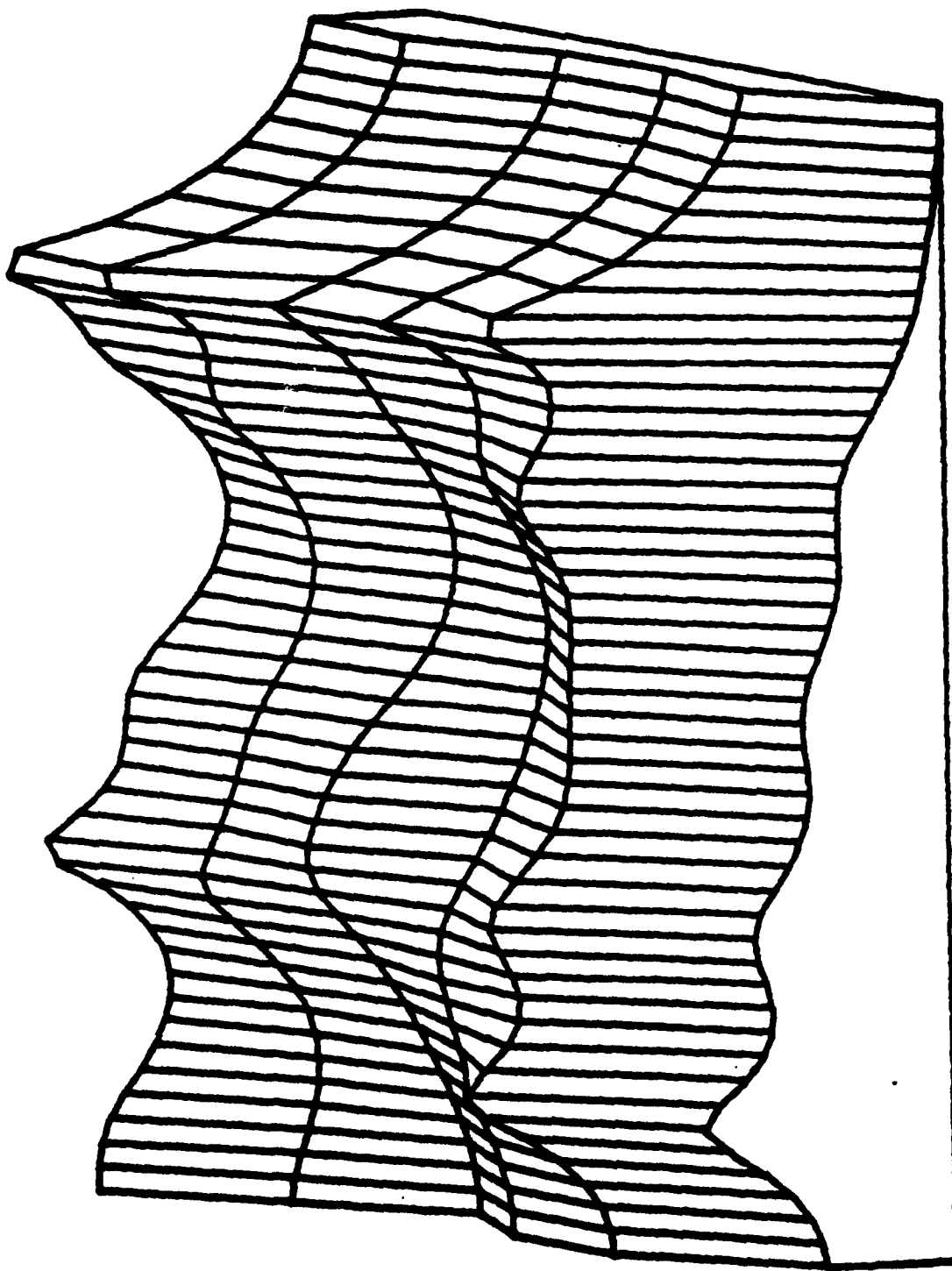


Figure IV-7.2.2.b

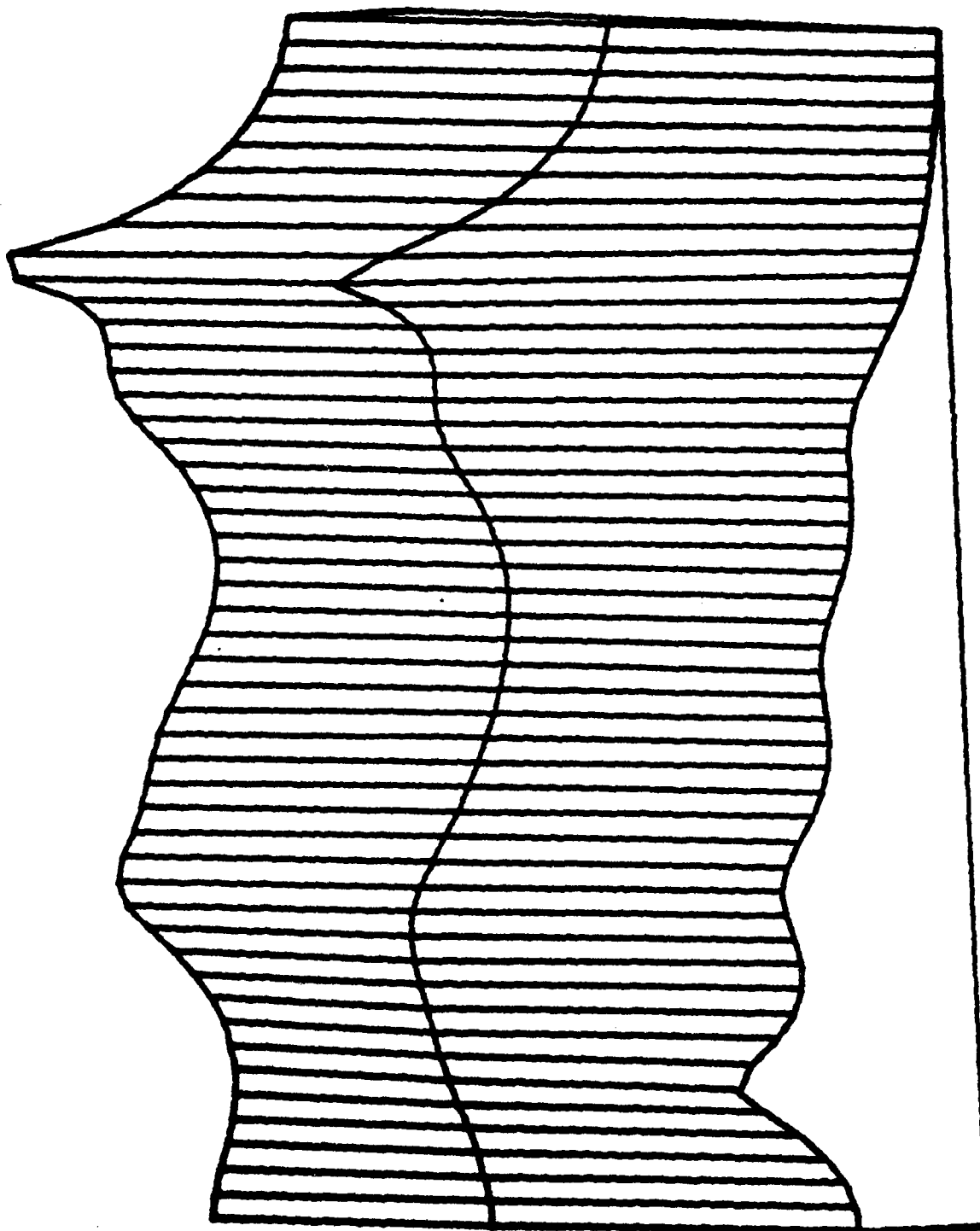


Figure IV-7.2.2.c

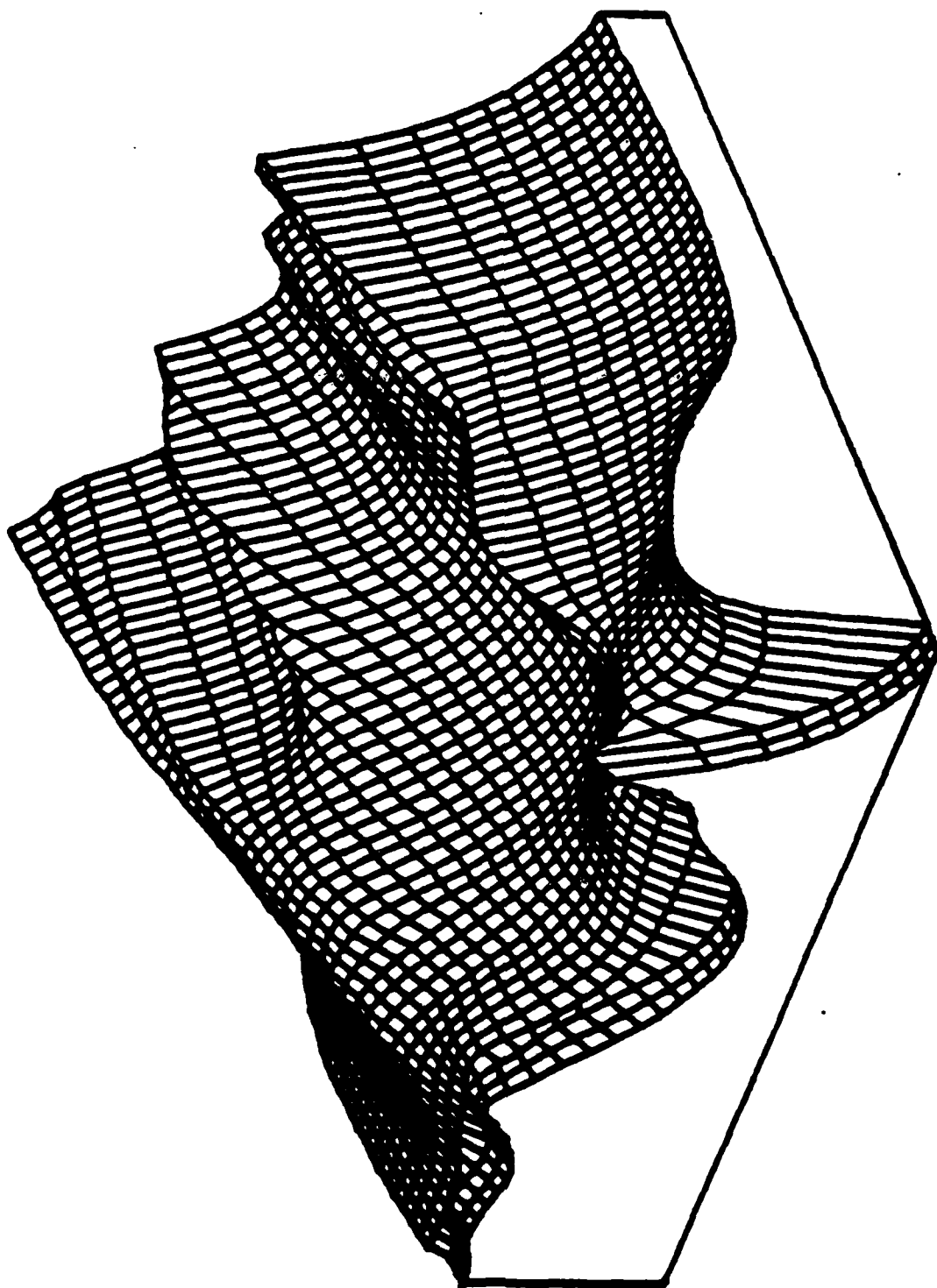


Figure IV-7.2.3.a

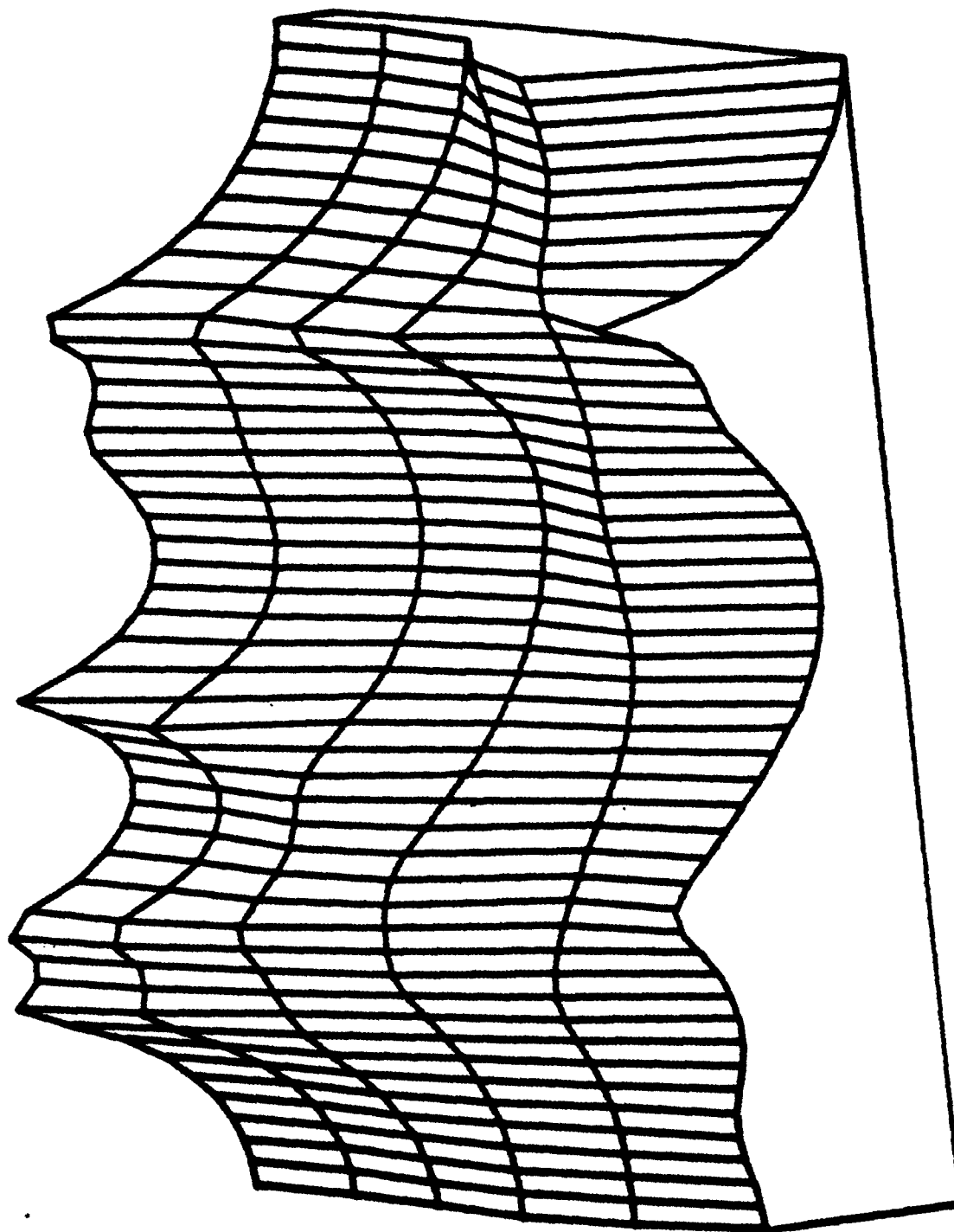


Figure IV-7.2.3.b

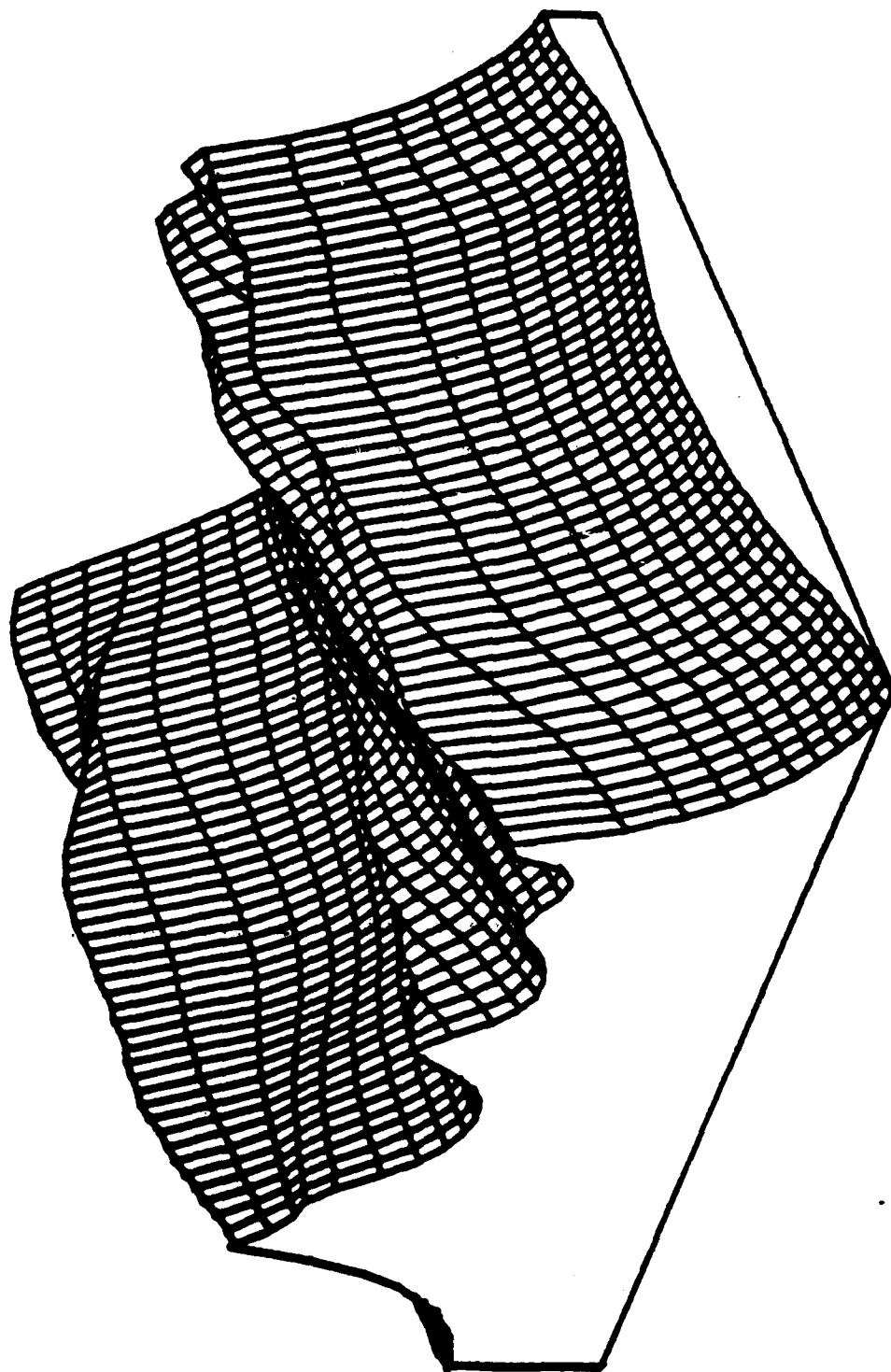


Figure IV-7.2.4.a

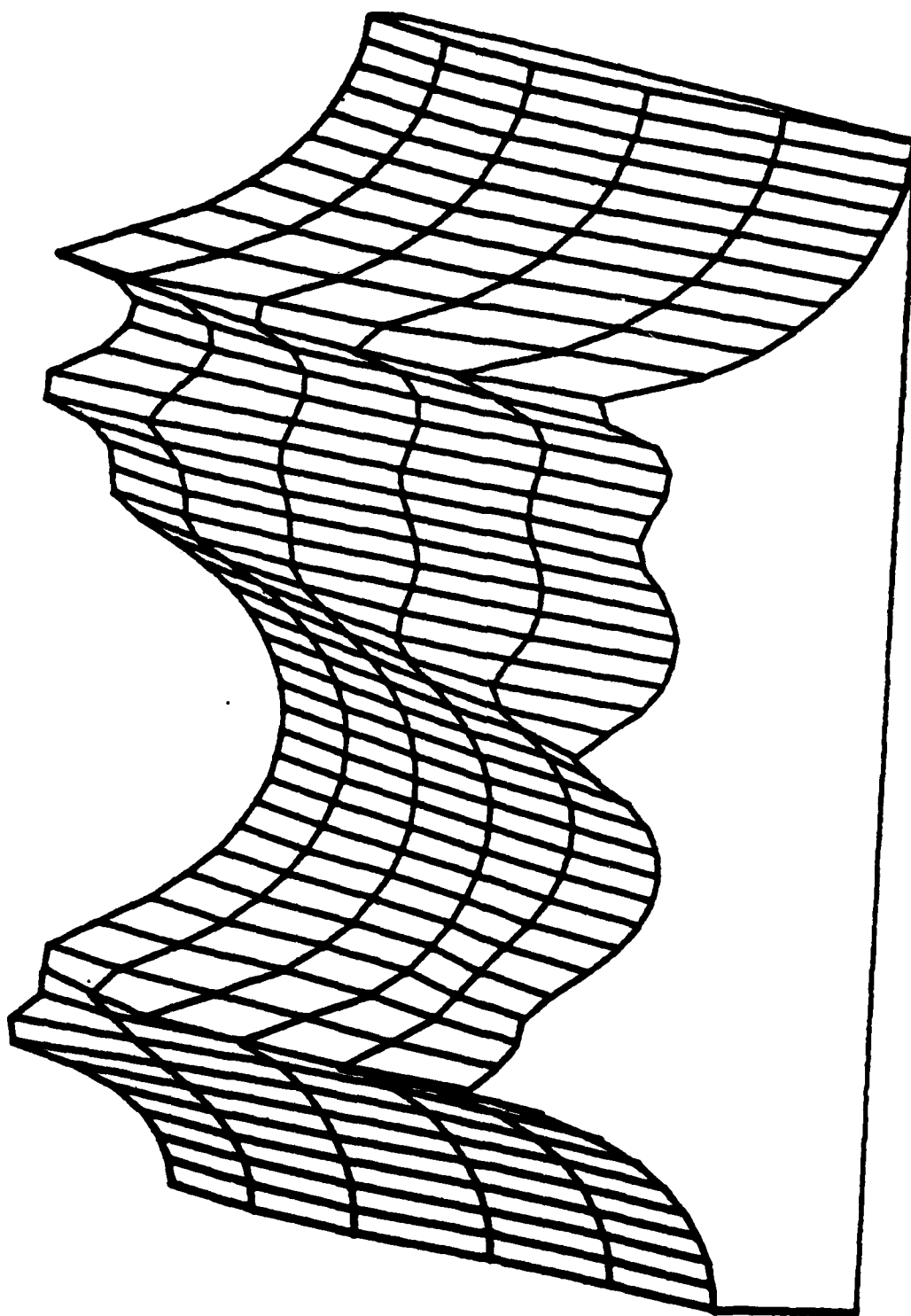


Figure IV-7.2.4.b

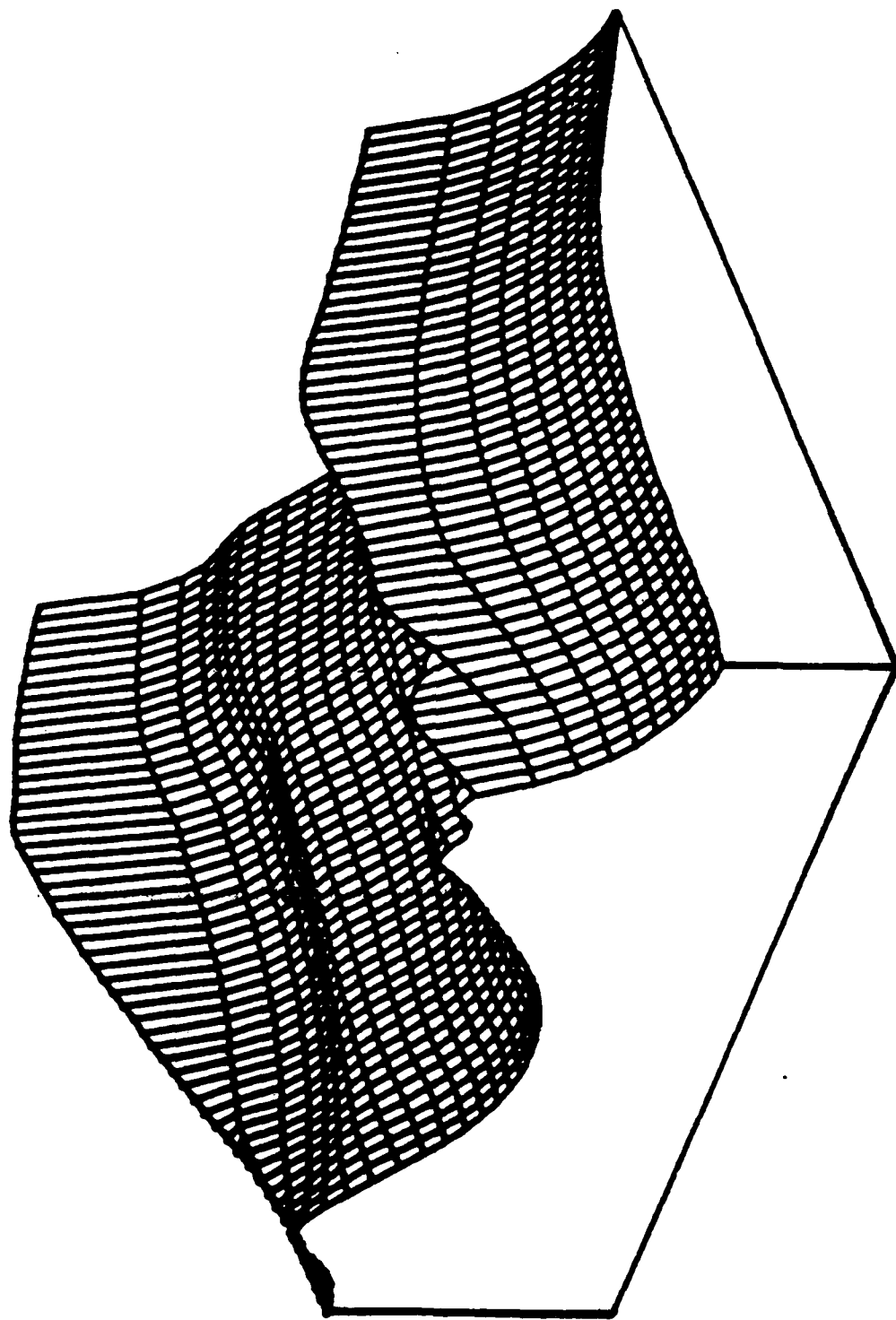


Figure IV-7.2.5.a

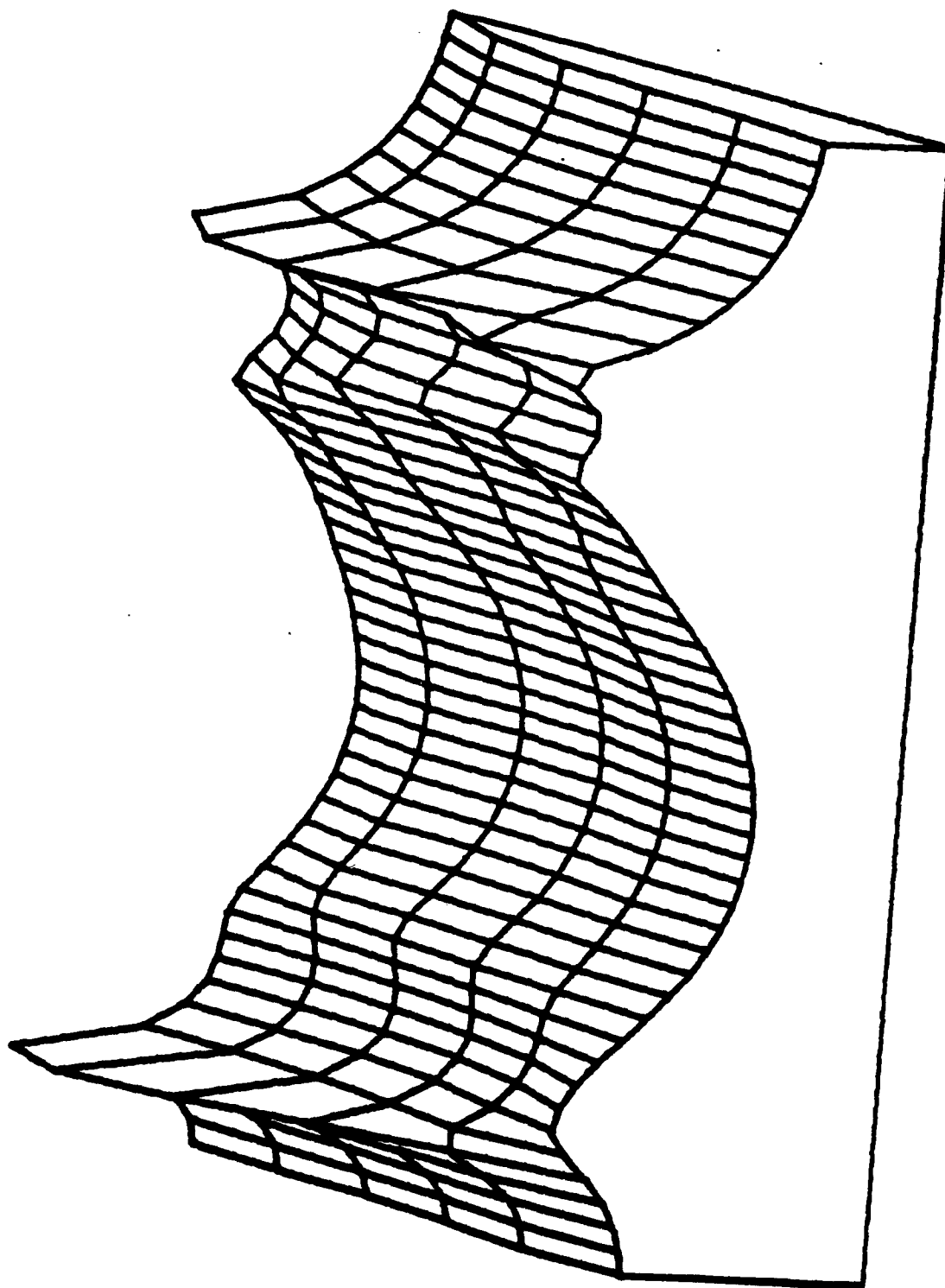


Figure IV-7.2.5.b

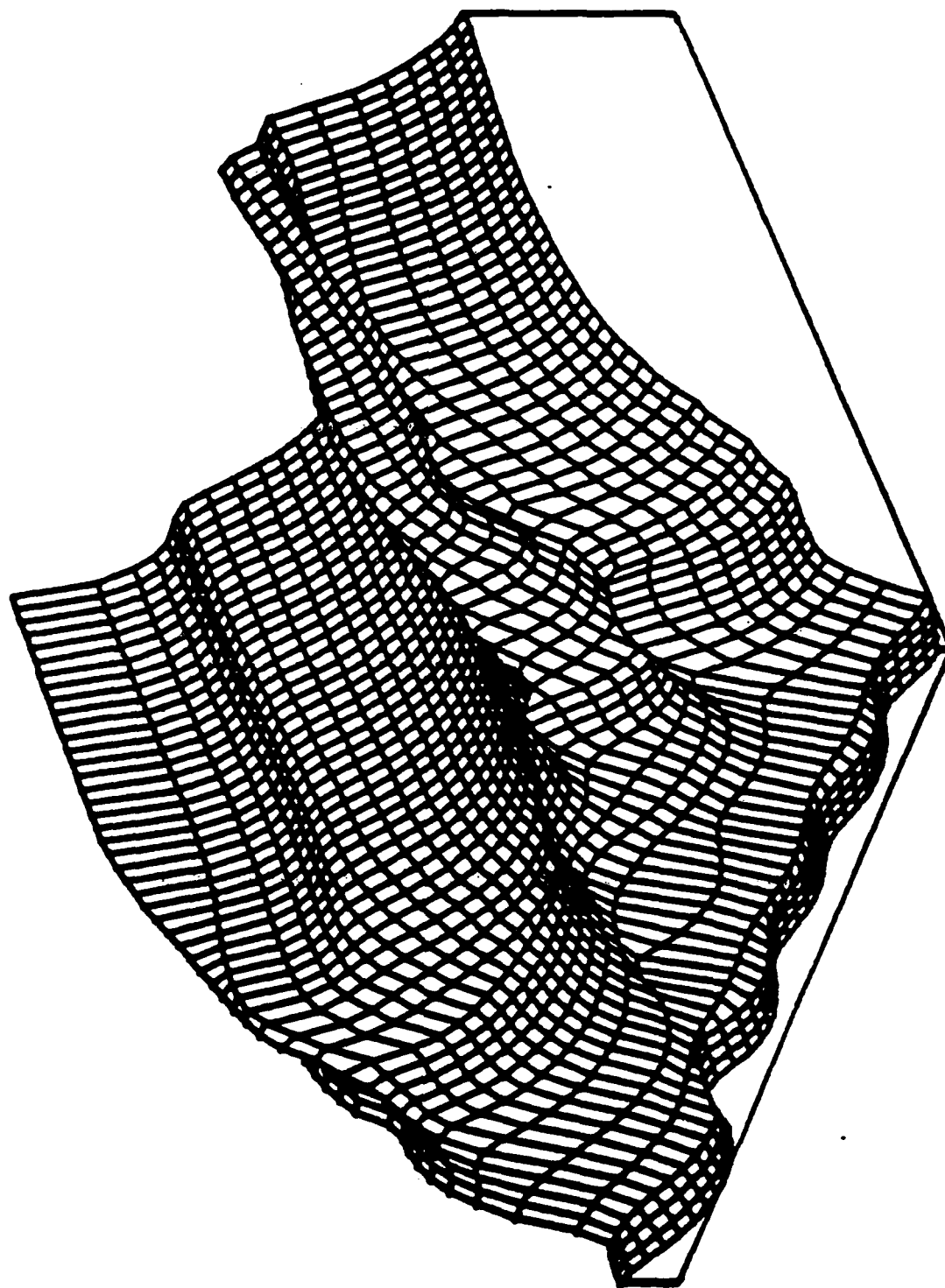


Figure IV-7.3.2.a-Point 300

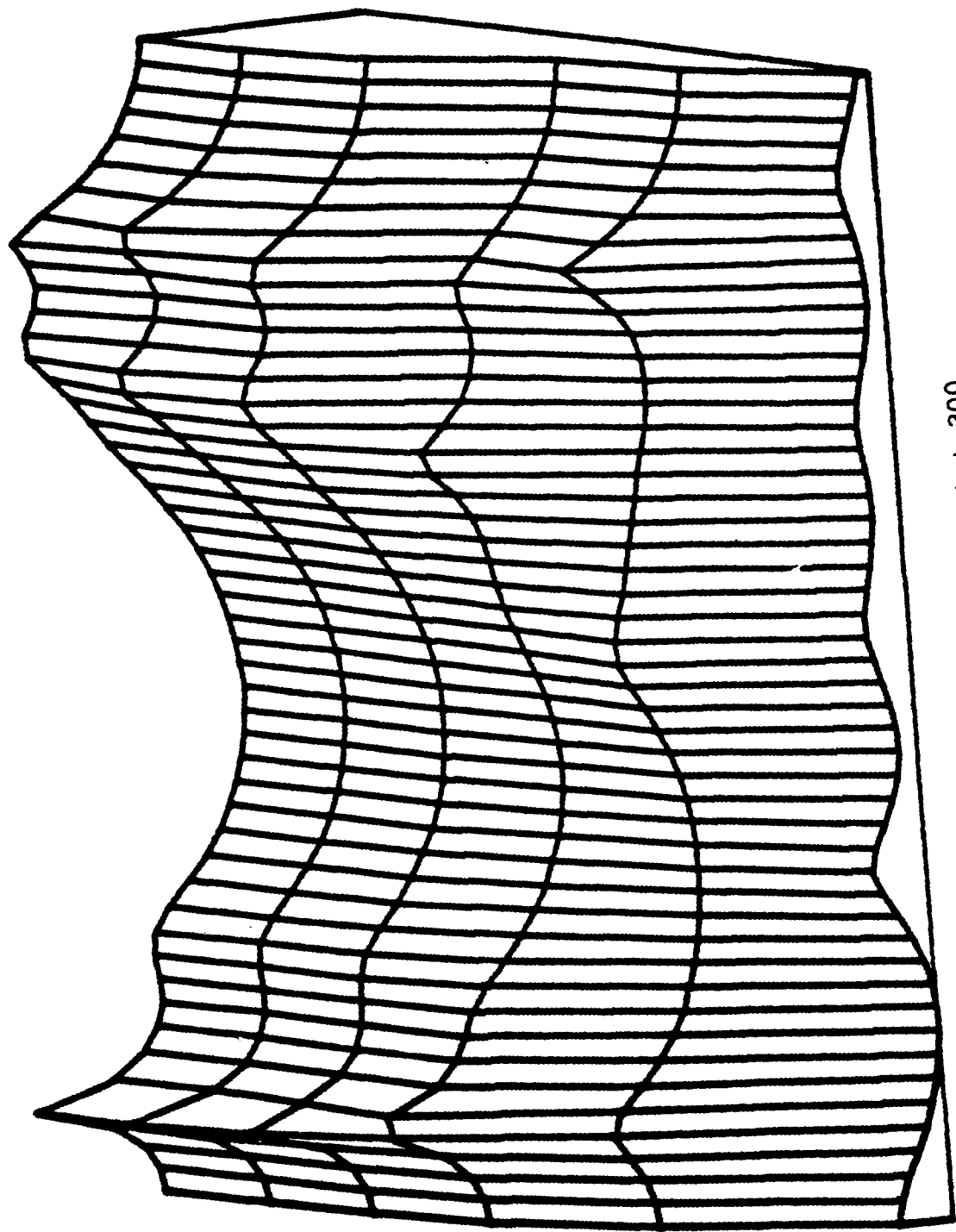


Figure IV-7.3.2.b-Point 300

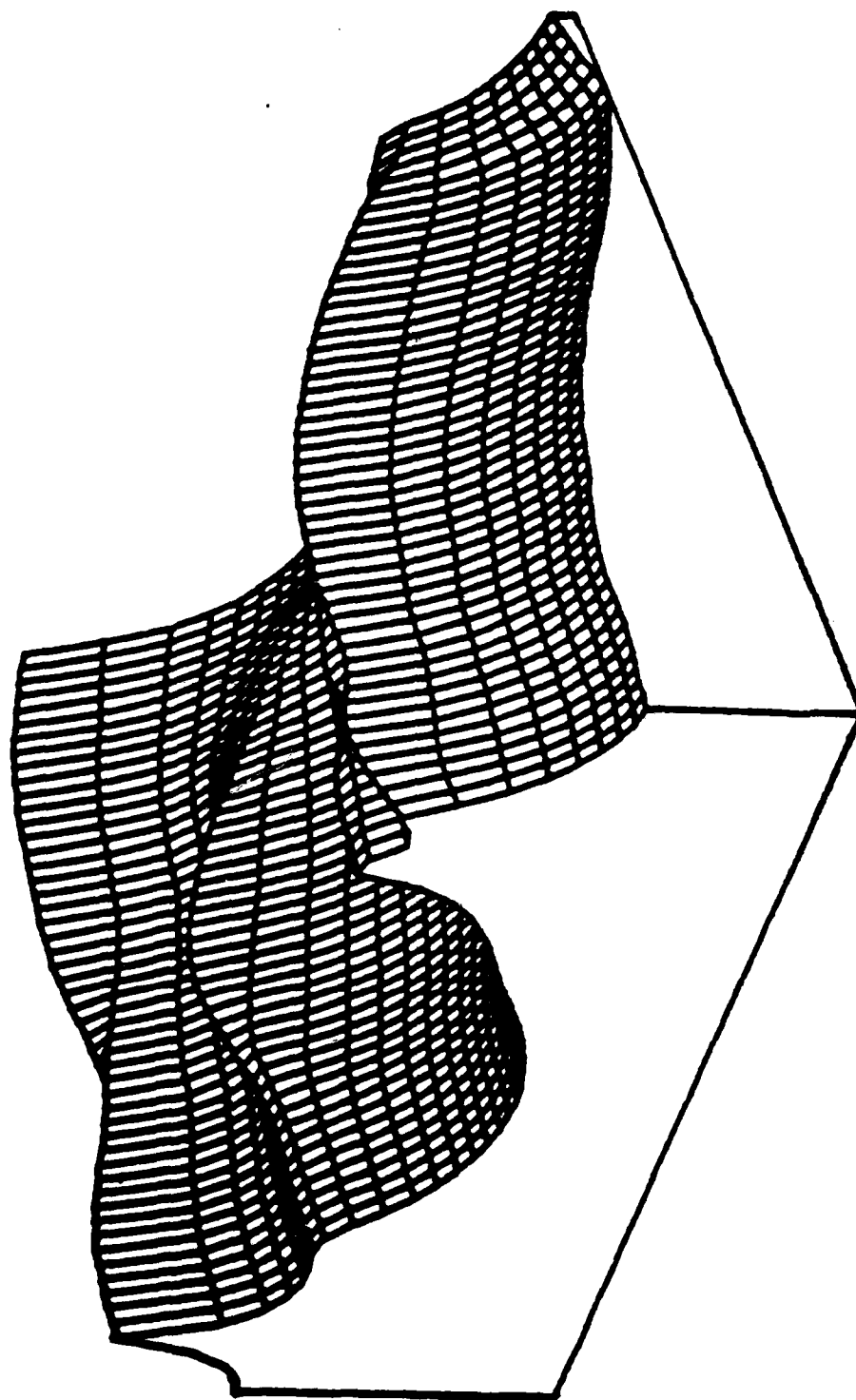


Figure IV-7.3.3.a-Point 300

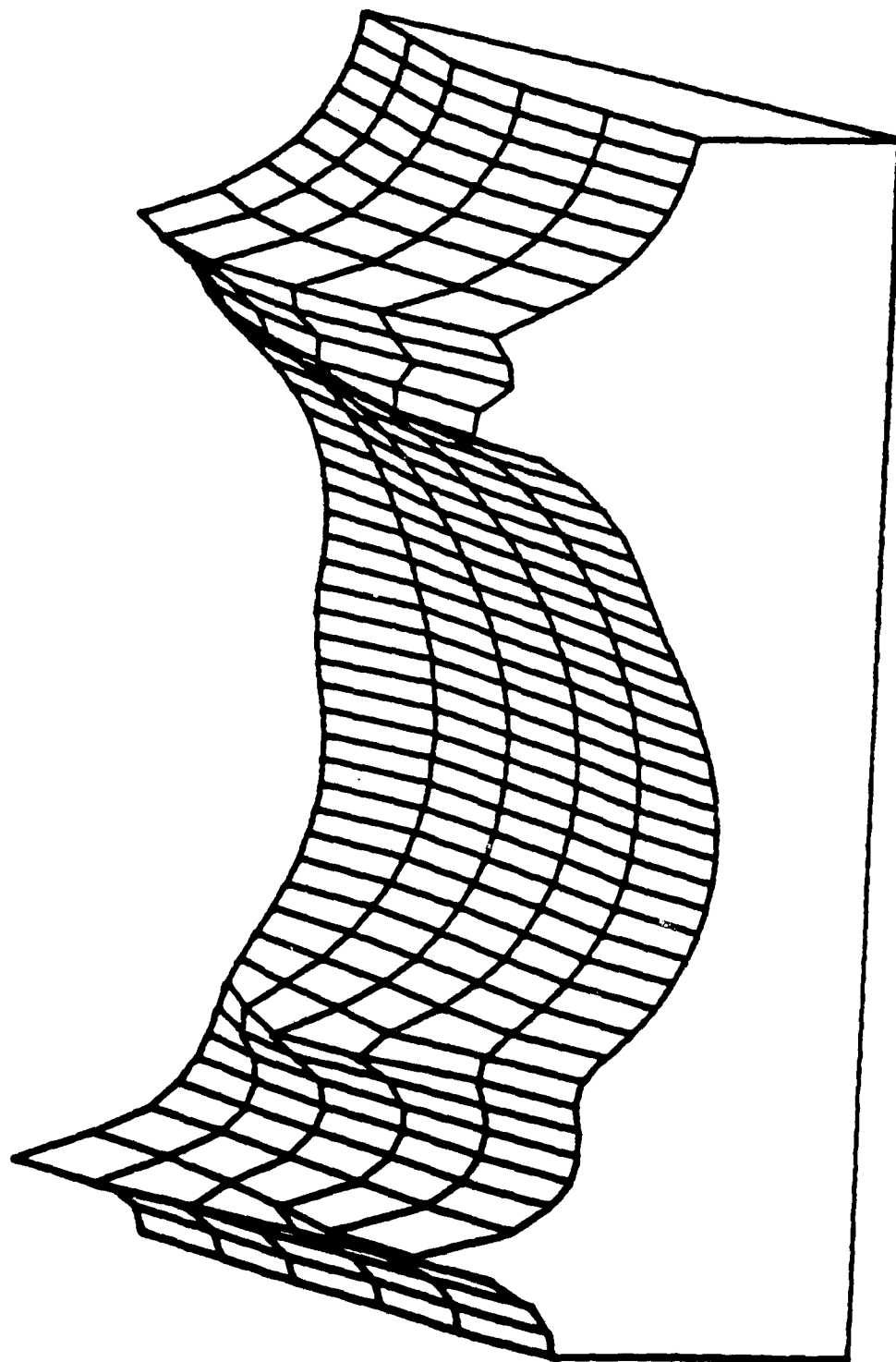


Figure IV-7.3.3.b-Point 300

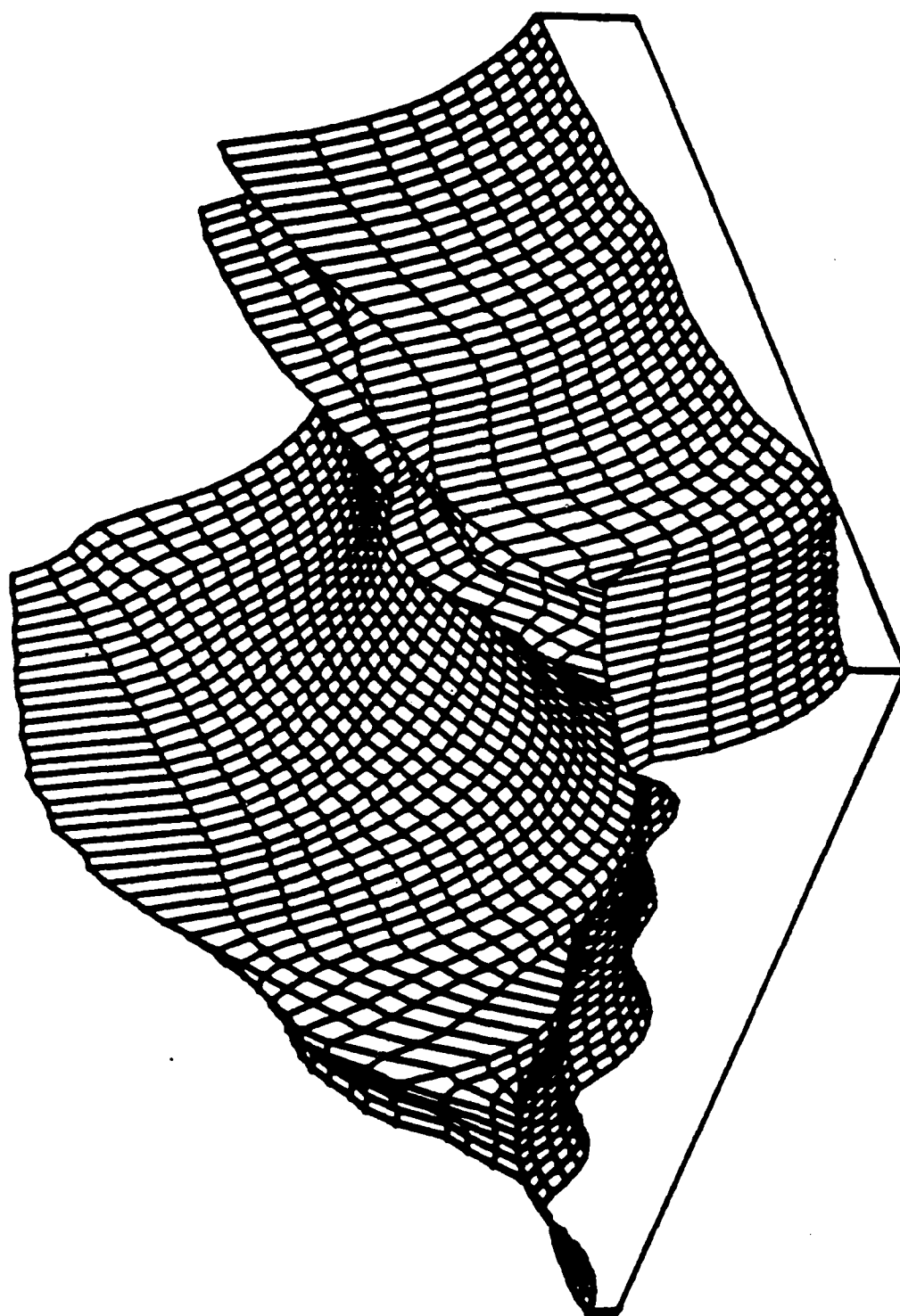


Figure IV-7.3.5.a

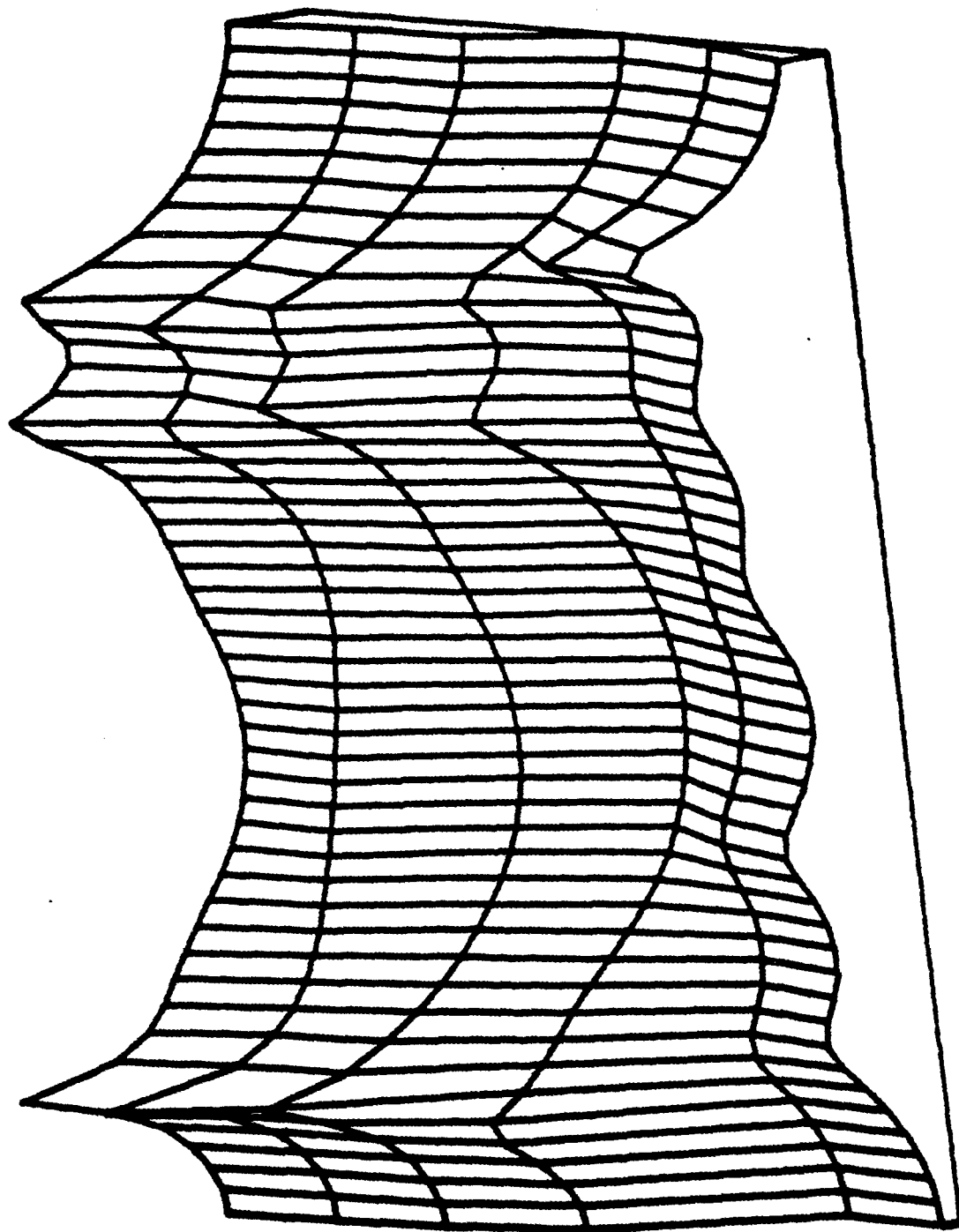


Figure IV-7.3.5.b

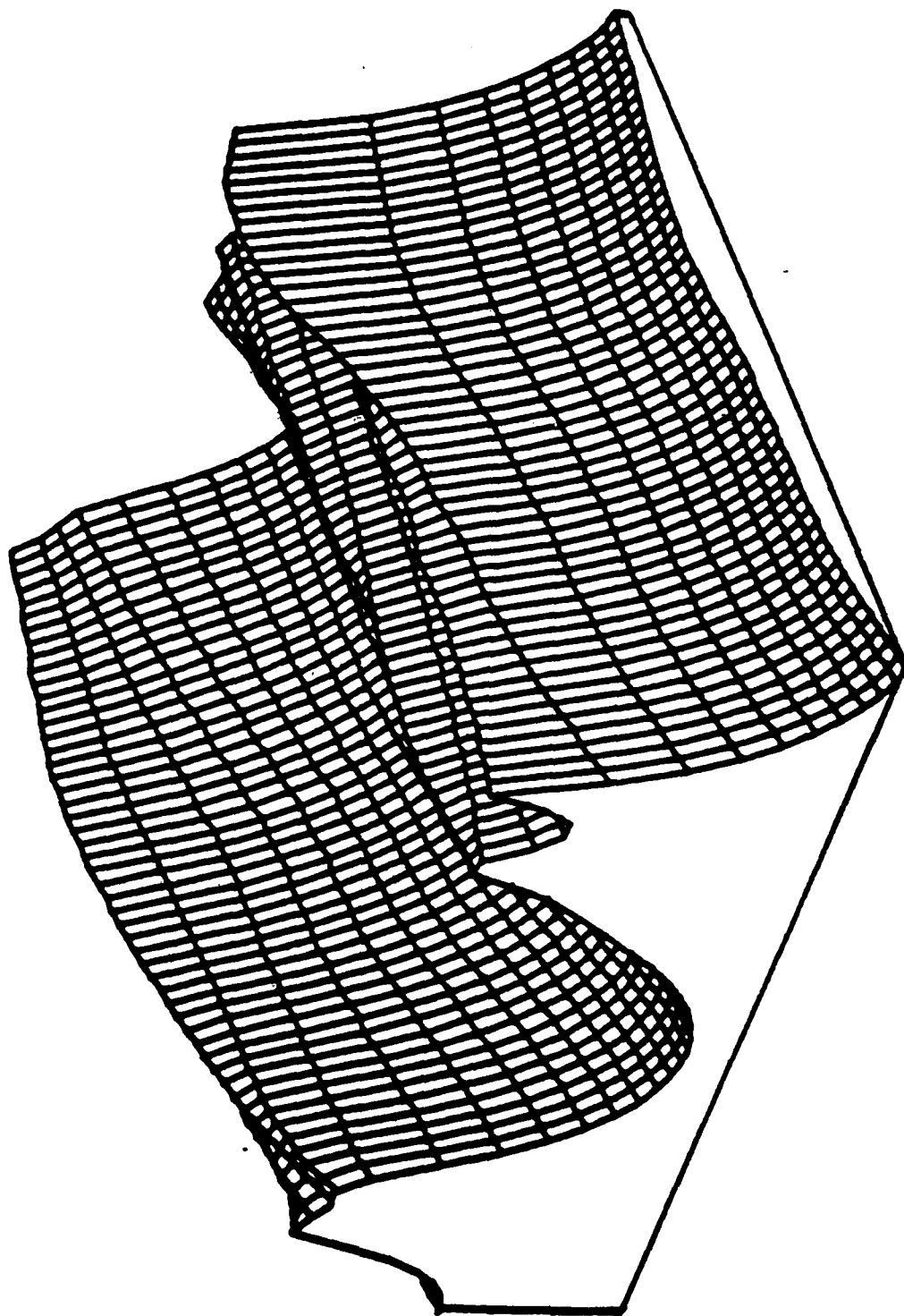


Figure IV-7,4.1.a

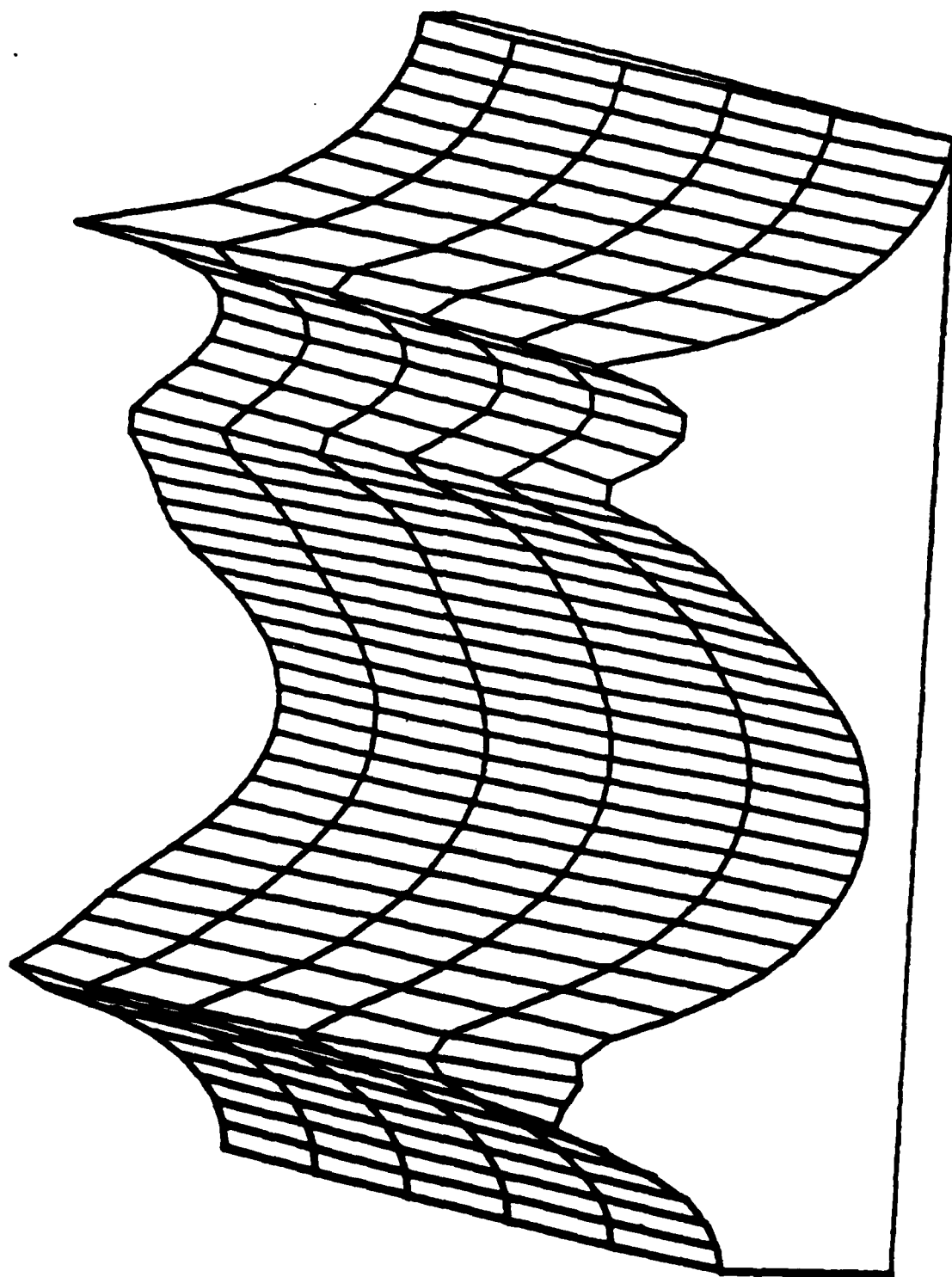


Figure IV-7.4.1.b

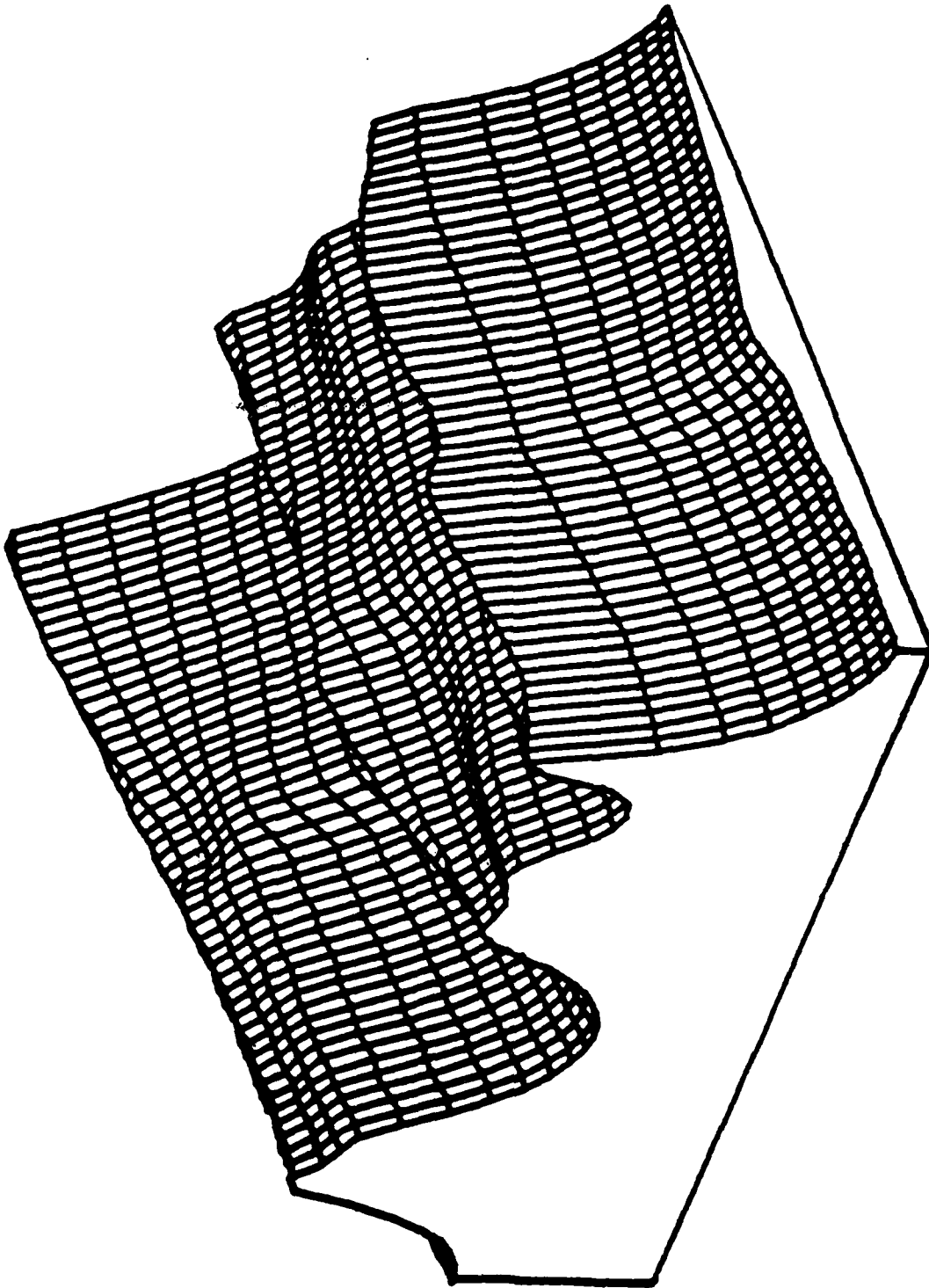


Figure IV-7.4.2.a

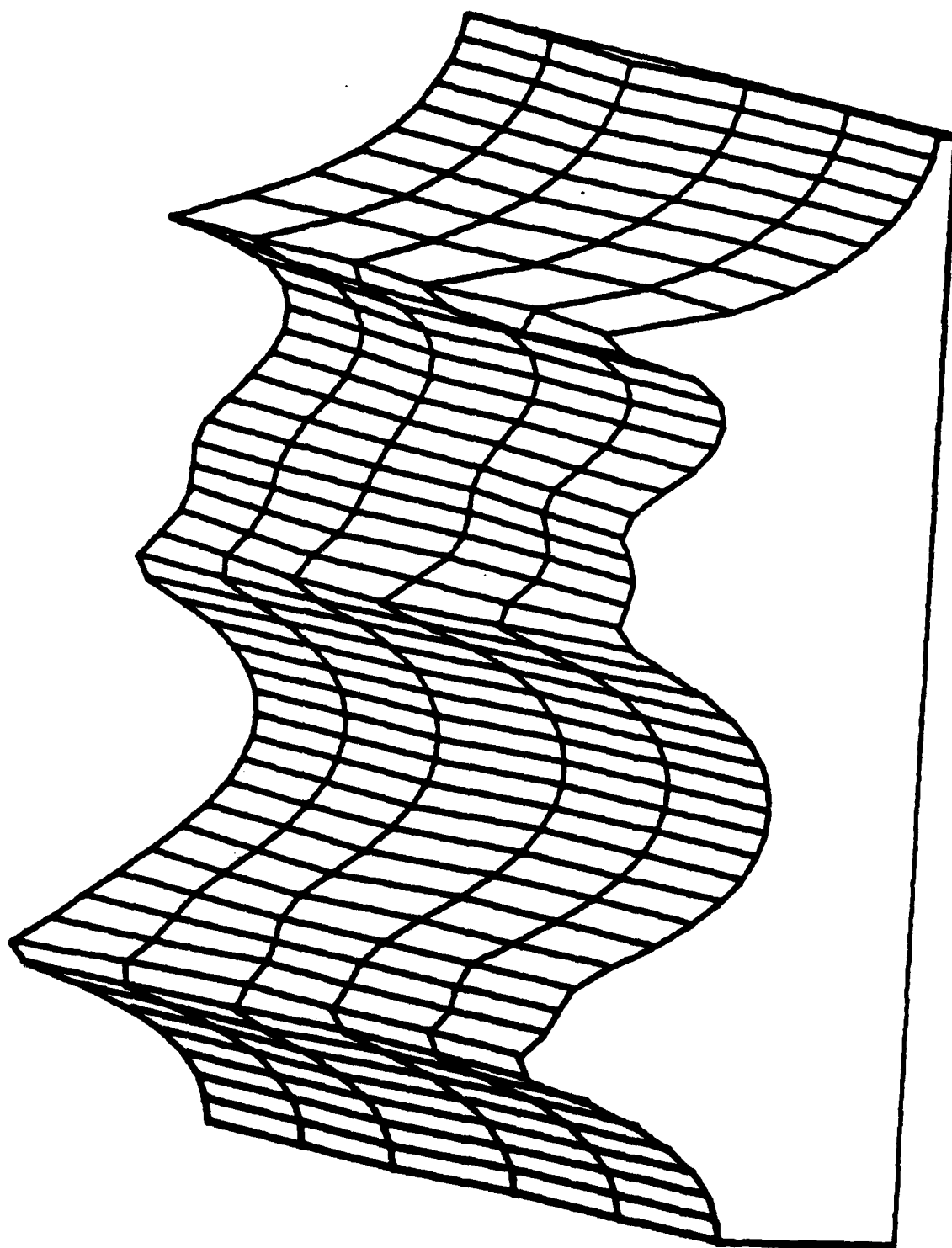


Figure IV-0.4.2.b

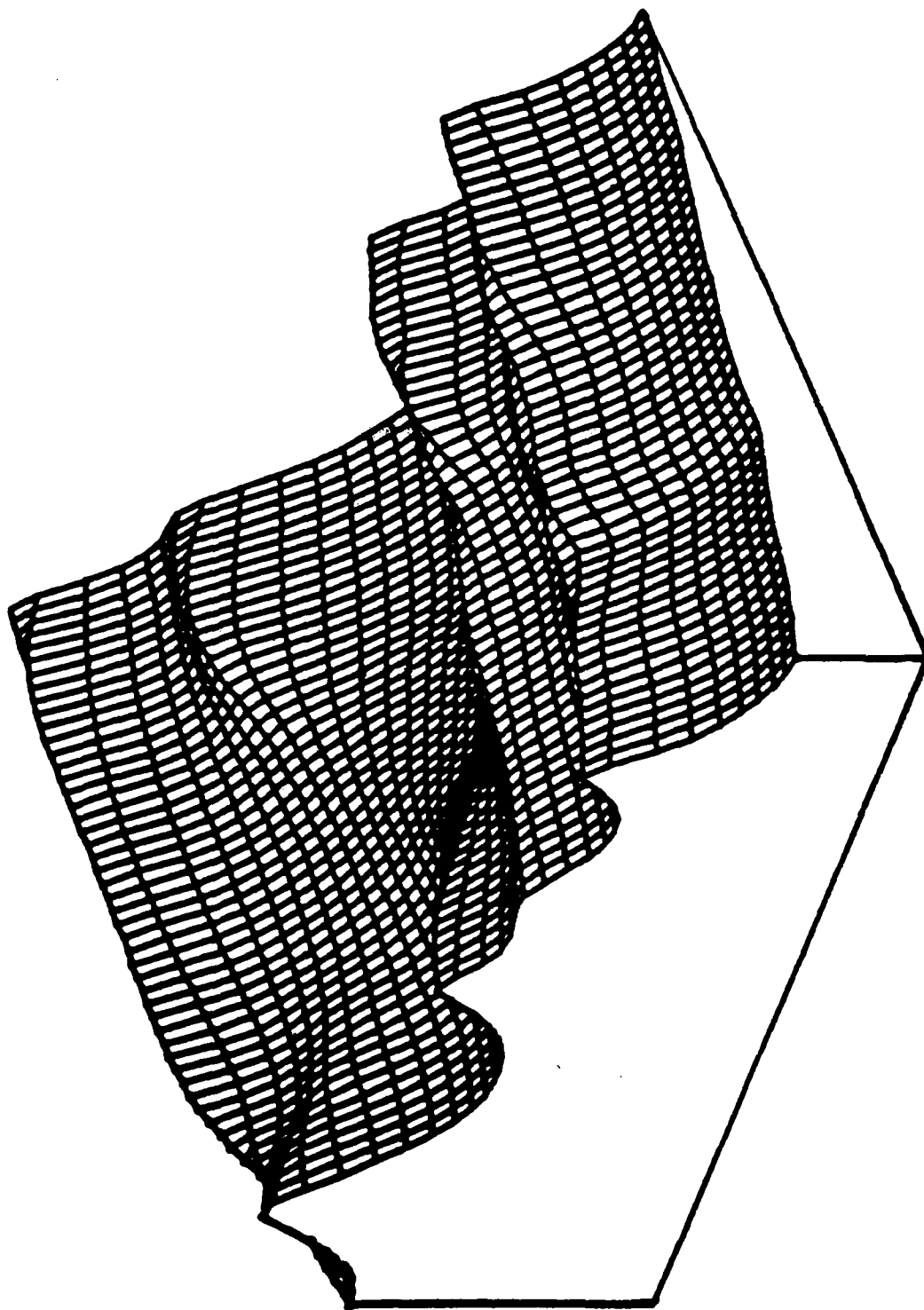


Figure IV-7.4.3.a

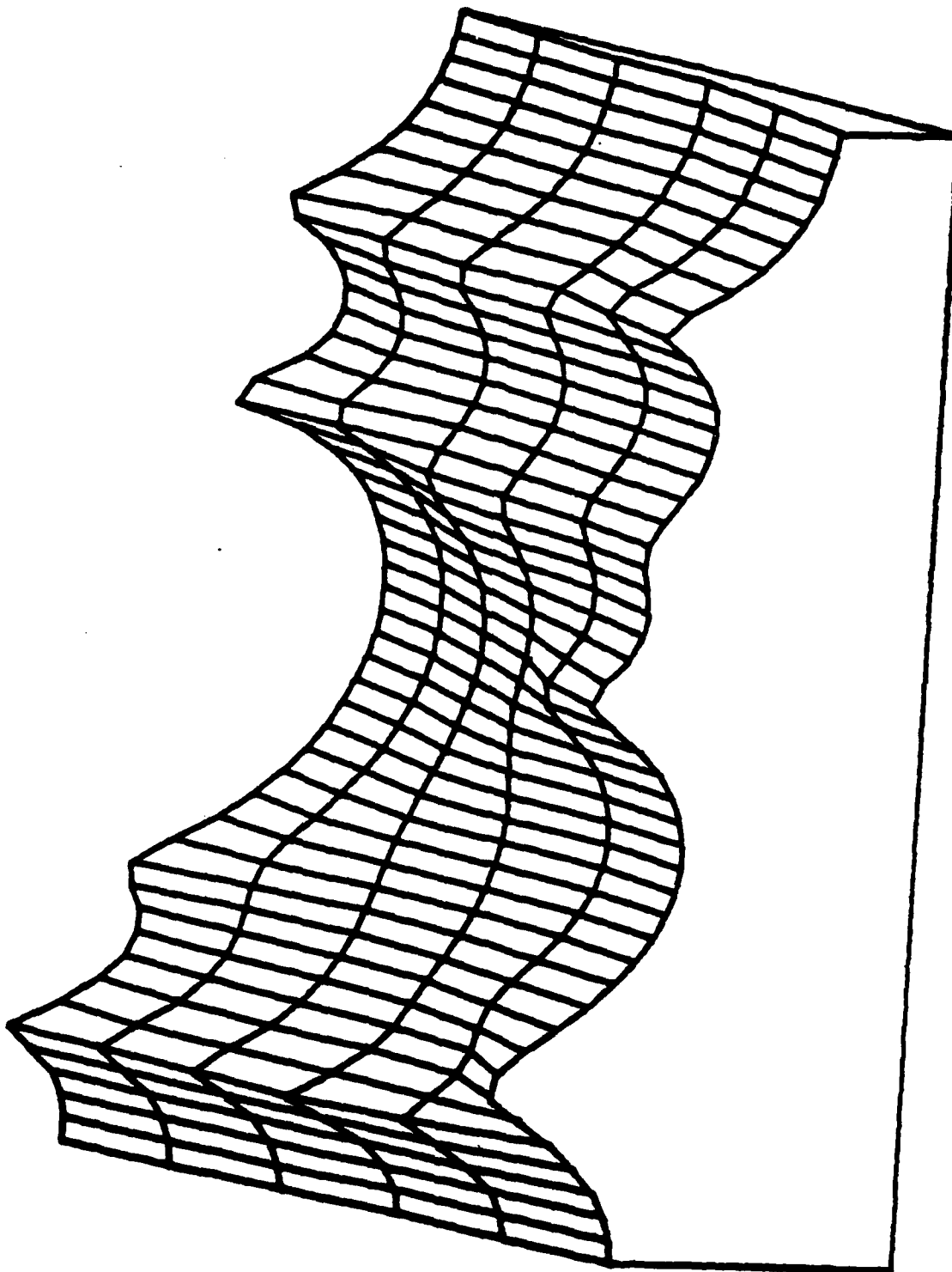


Figure IV-7.4.3.b

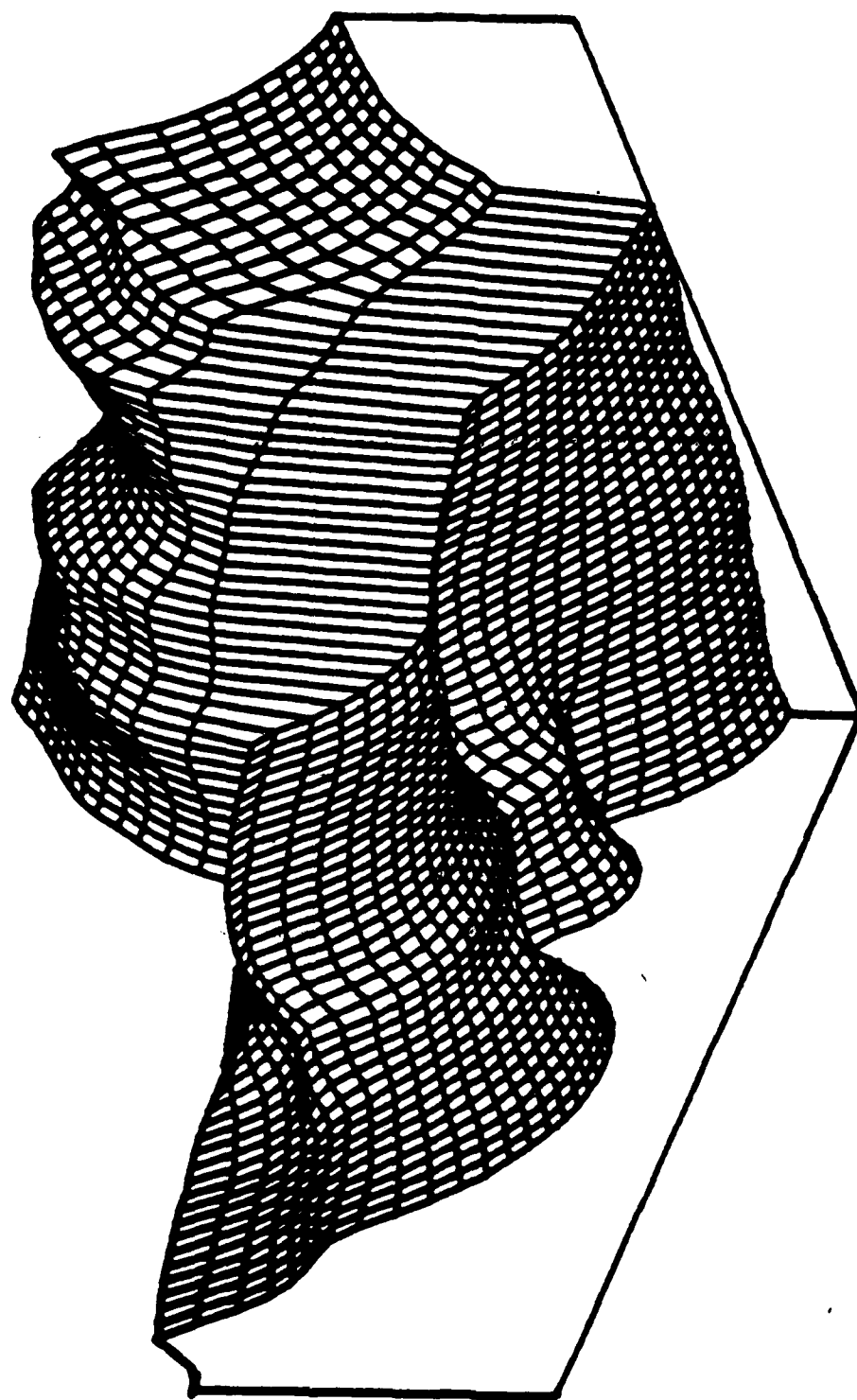


Figure IV-7.4.4.a

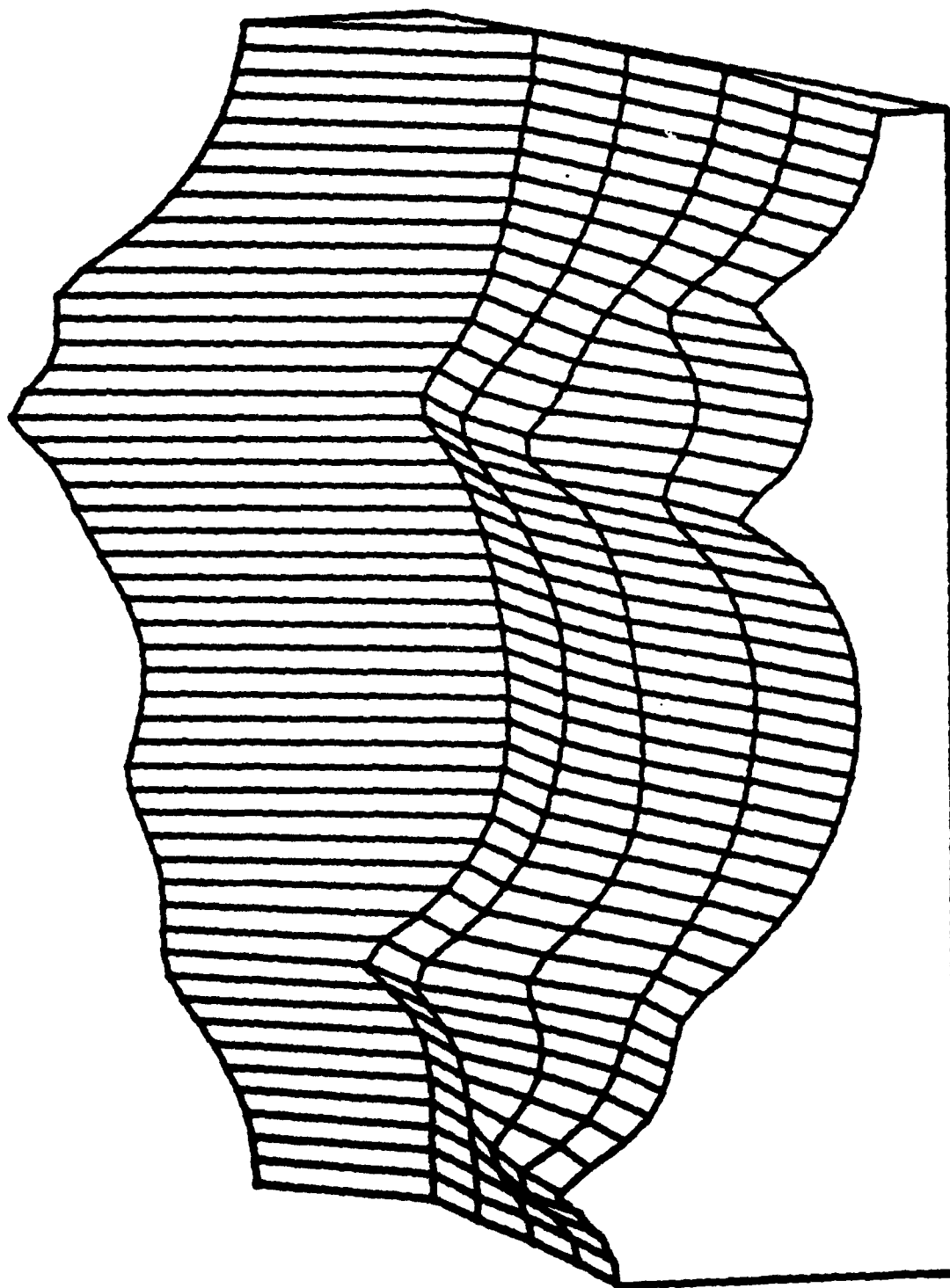


Figure IV-7.4.4.b

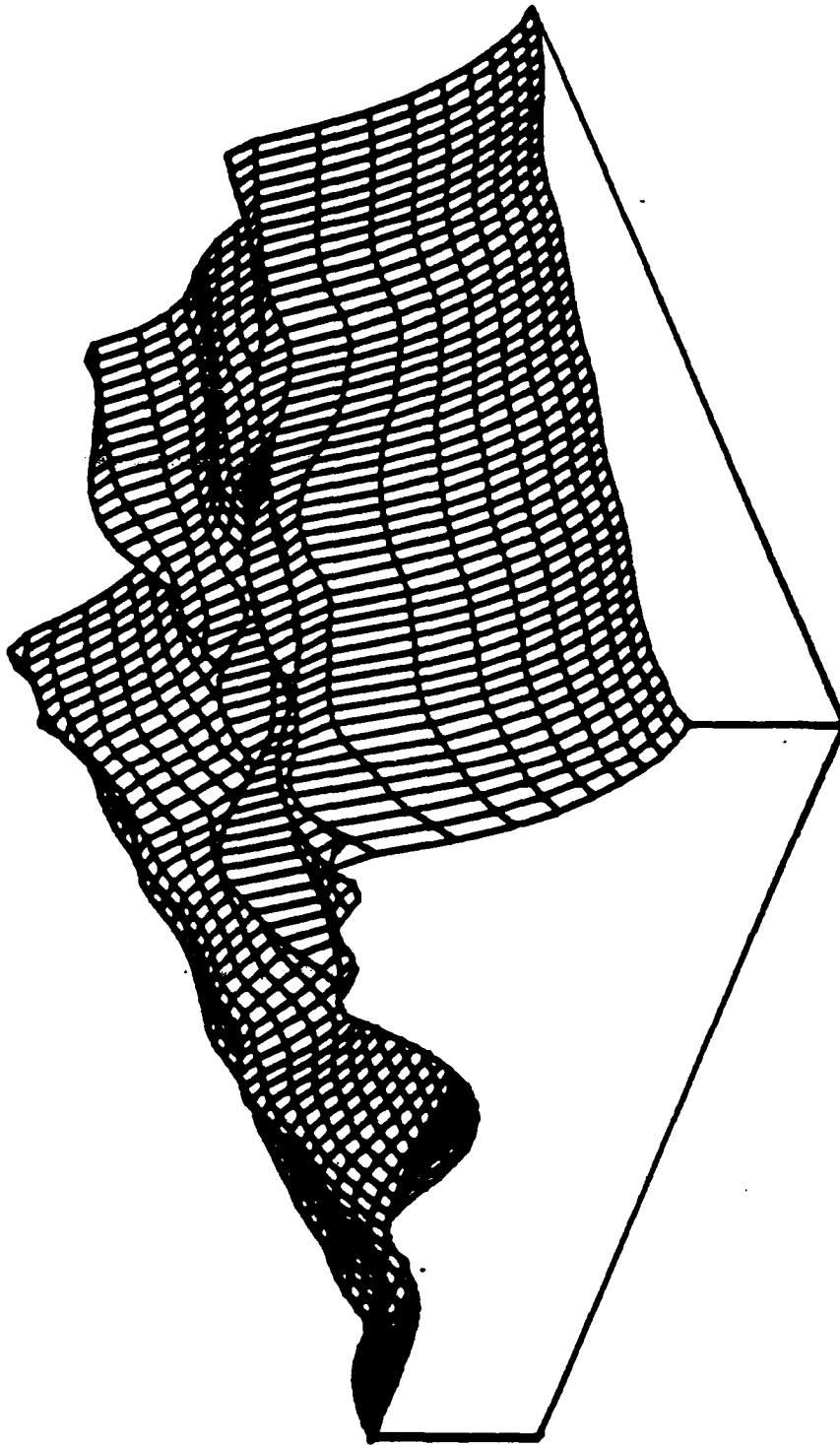


Figure IV-7.4.5.a

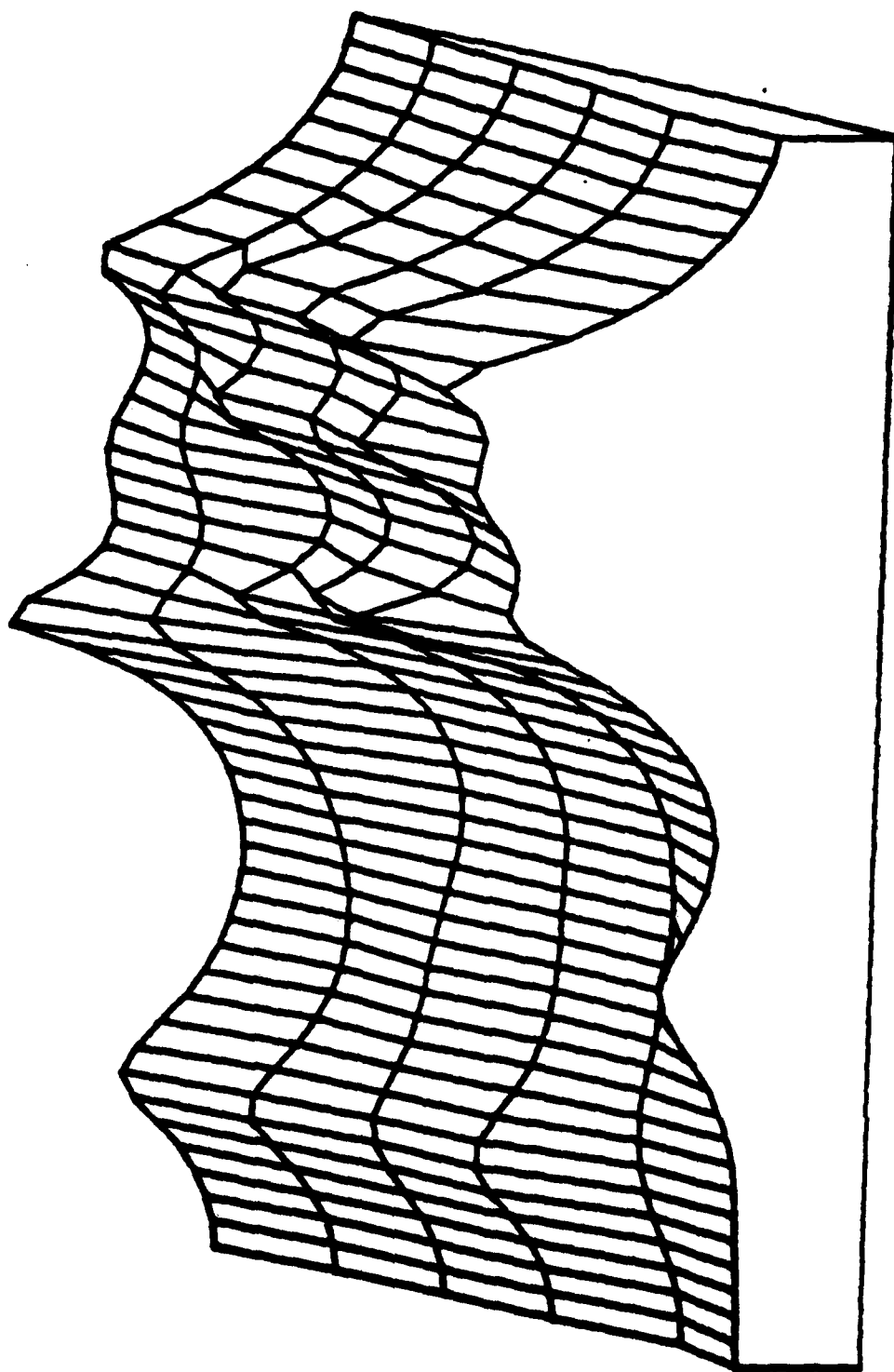


Figure IV-7.4.5.b

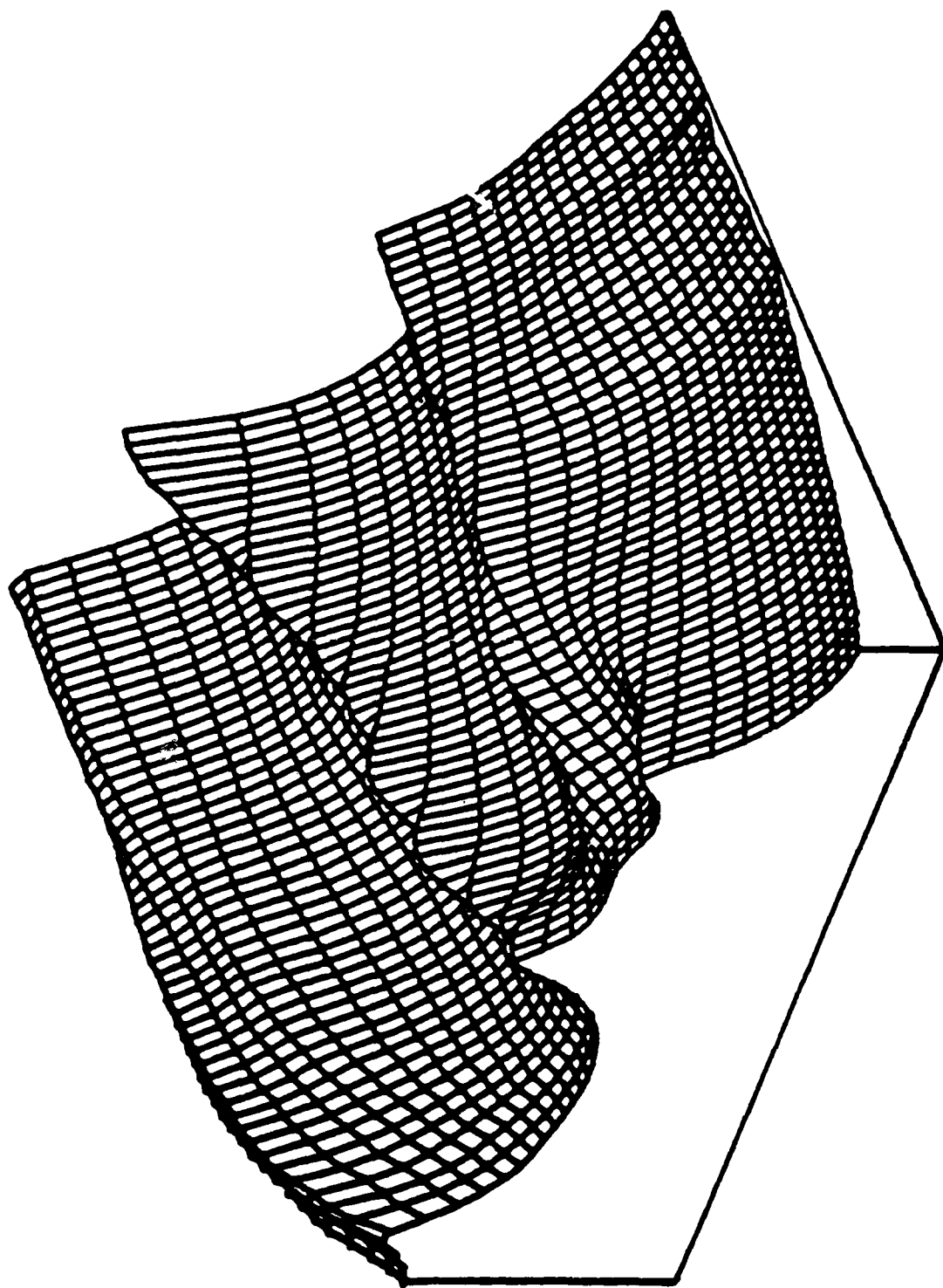


Figure IV-7.5.1.a

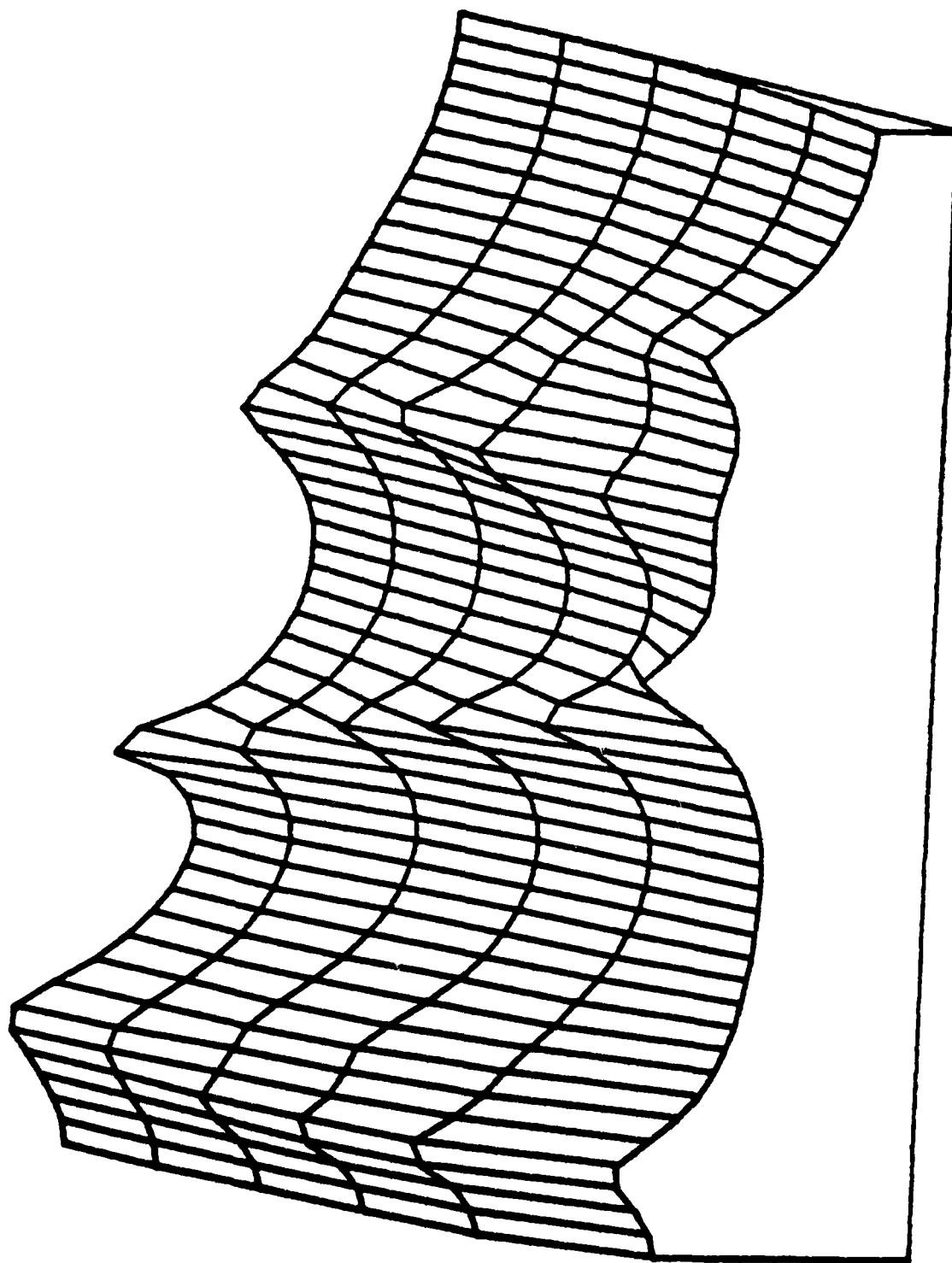


Figure IV-7.5.1.b

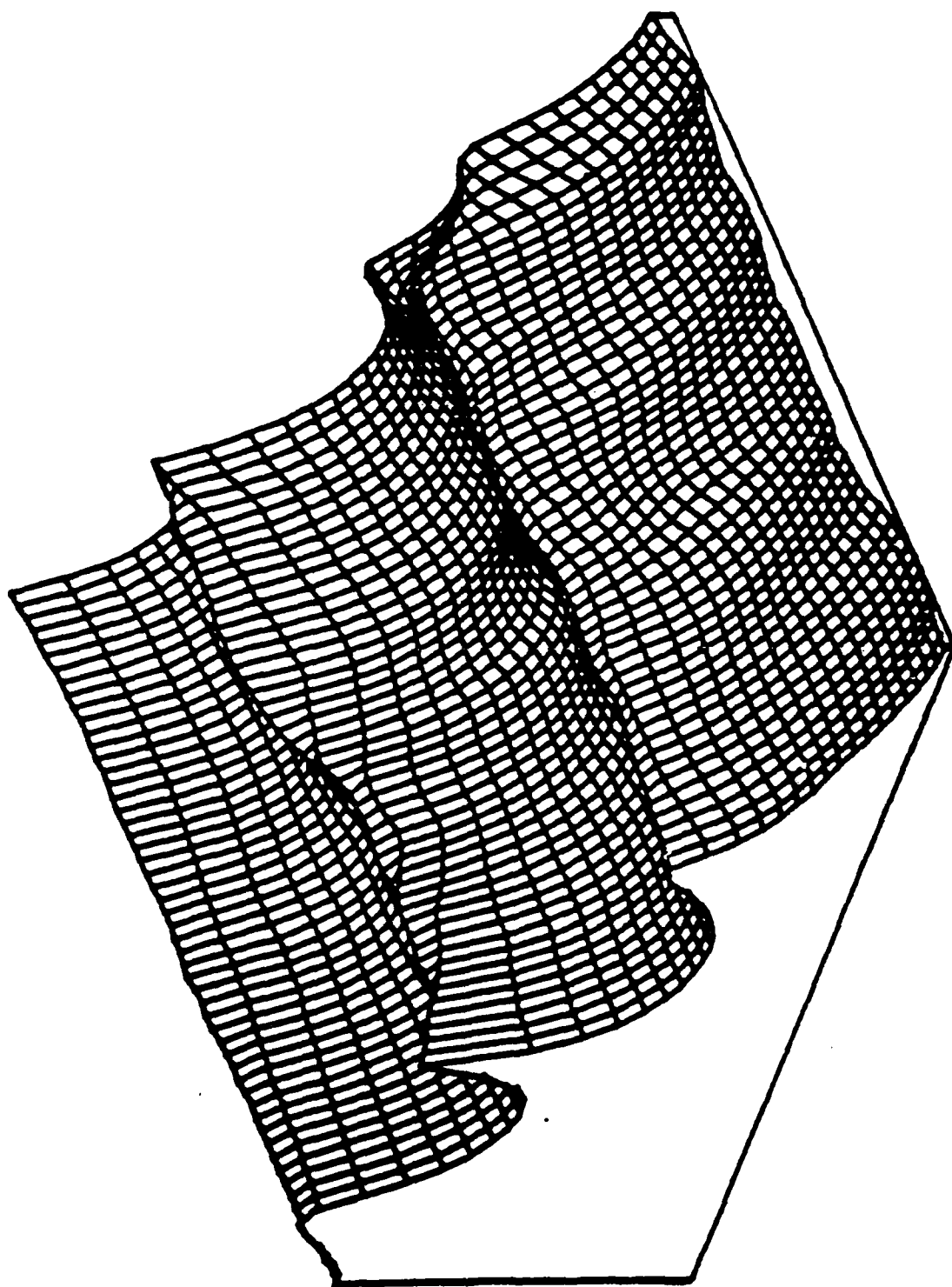


Figure IV-7.5.2.a

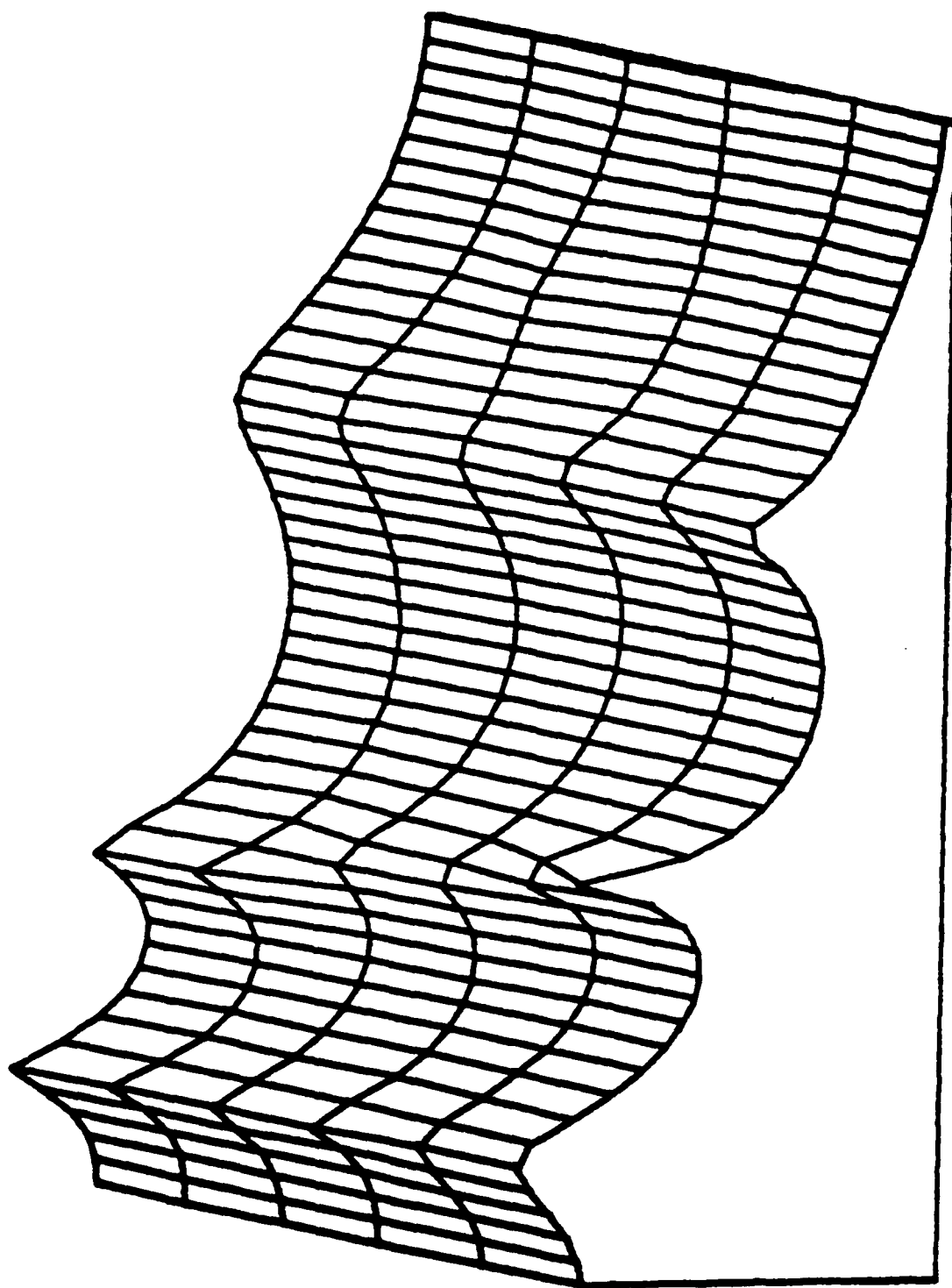


Figure IV-7.5.2.b

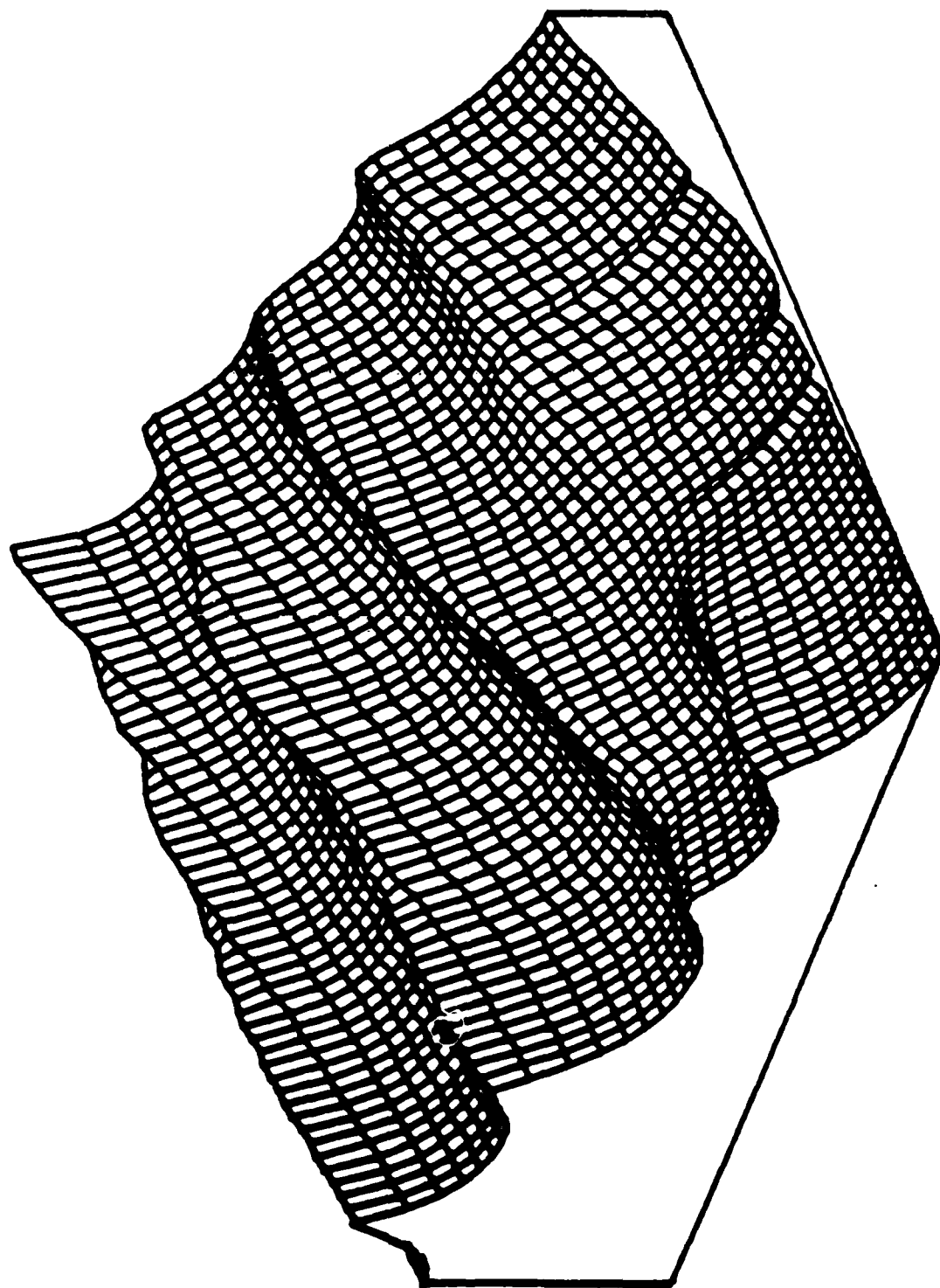


Figure IV-7.5.3.a

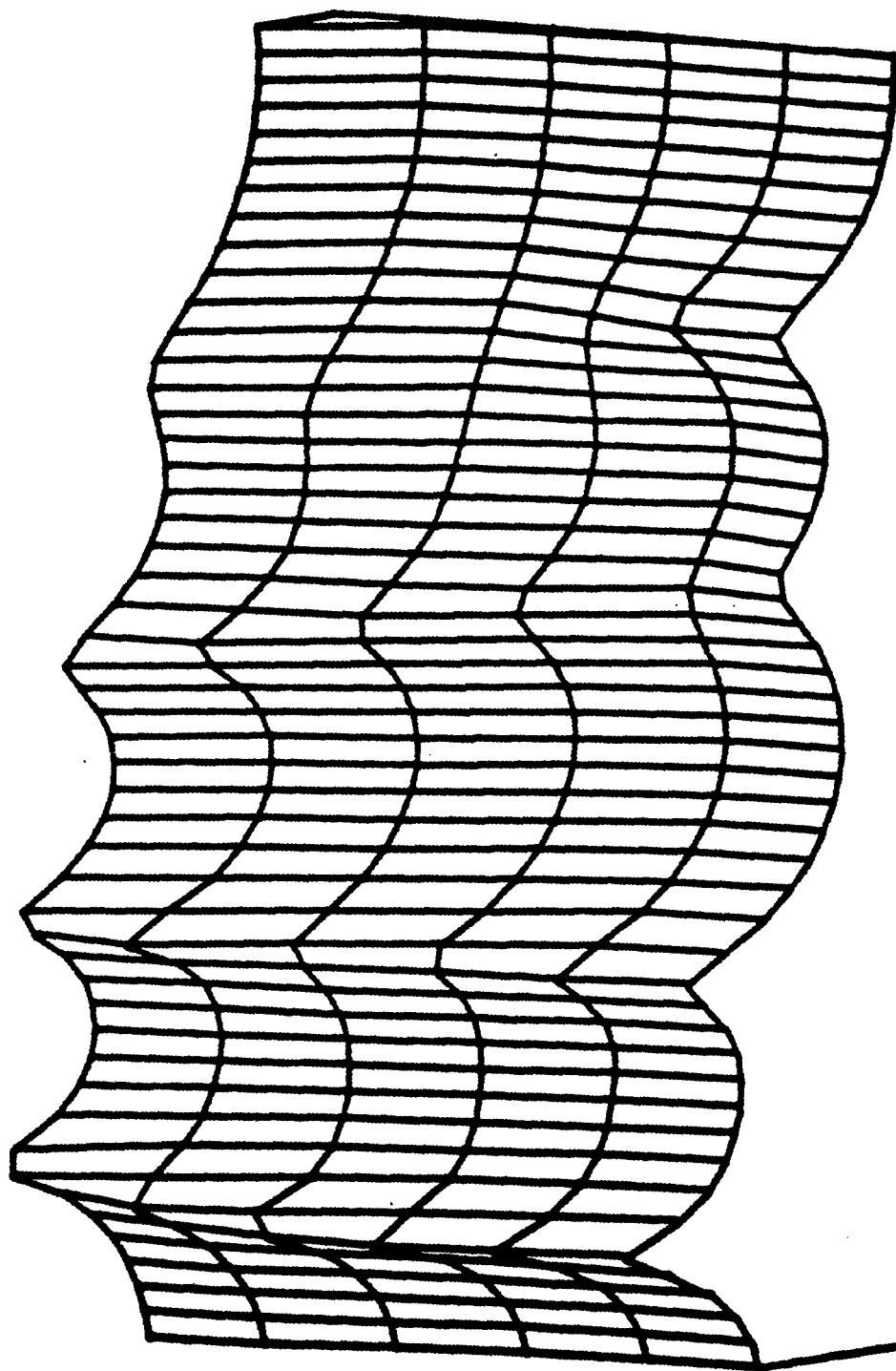


Figure IV-7.5.3.b

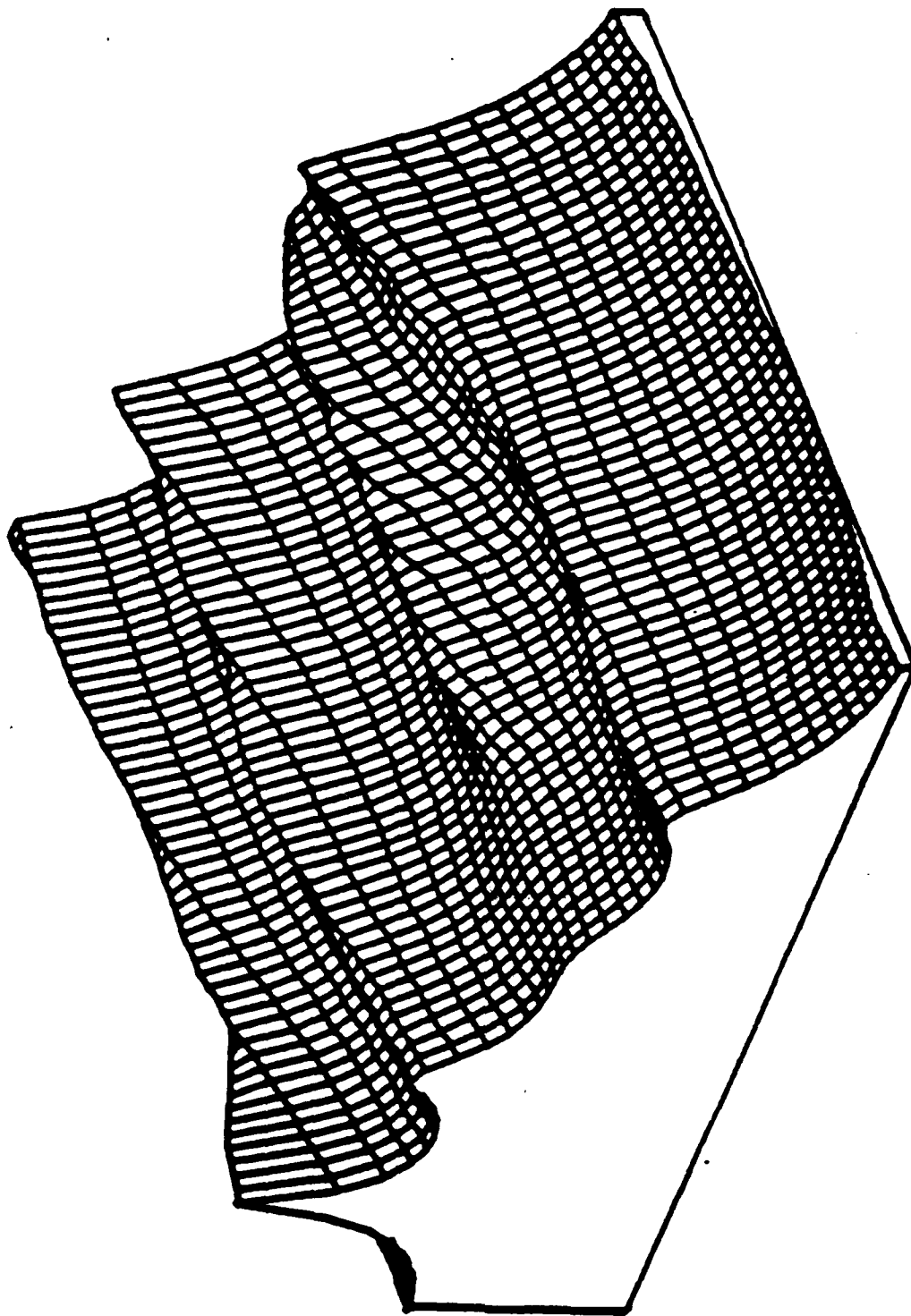


Figure IV-7.5.4.a

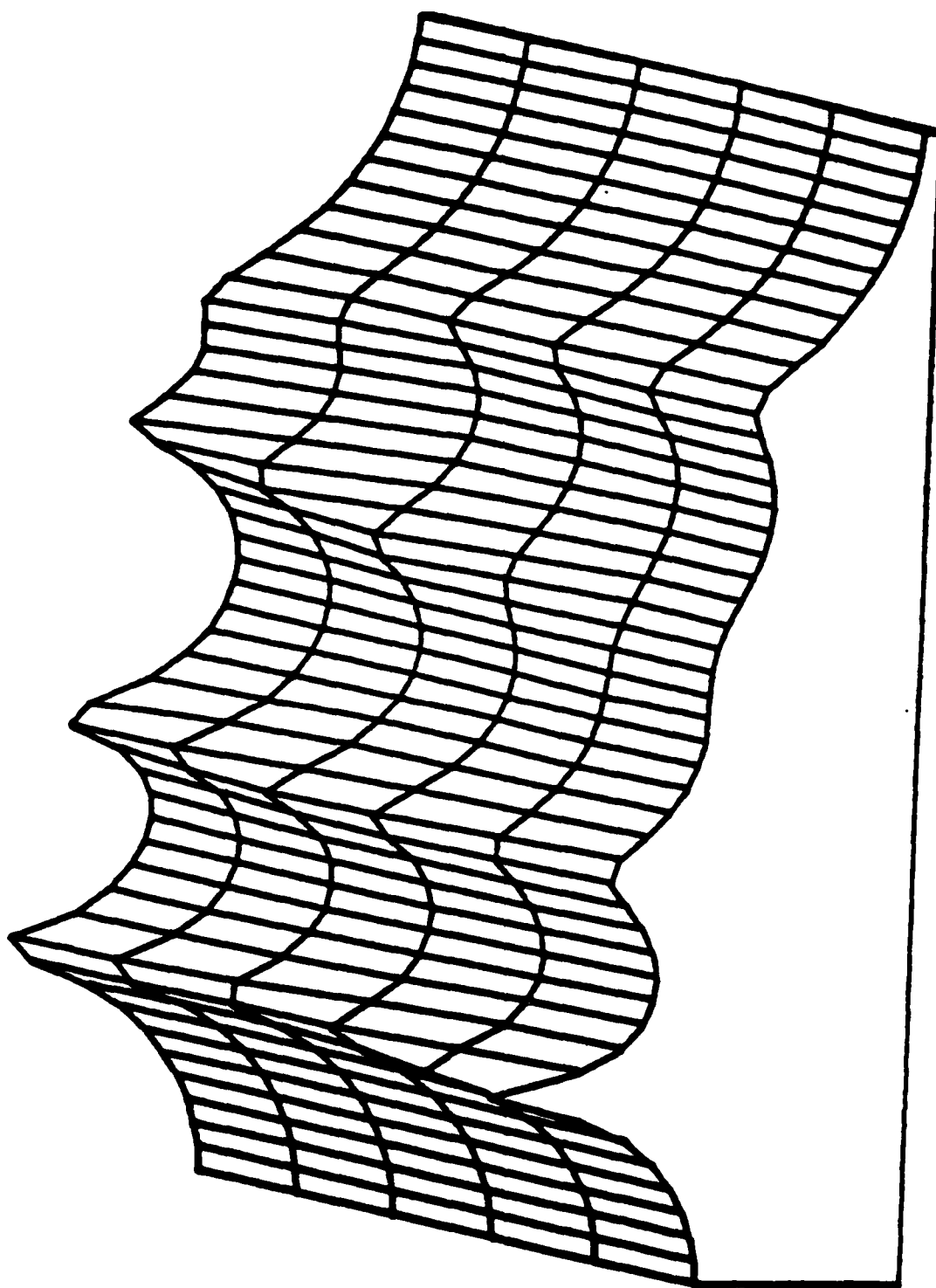


Figure IV-7.5.4.b

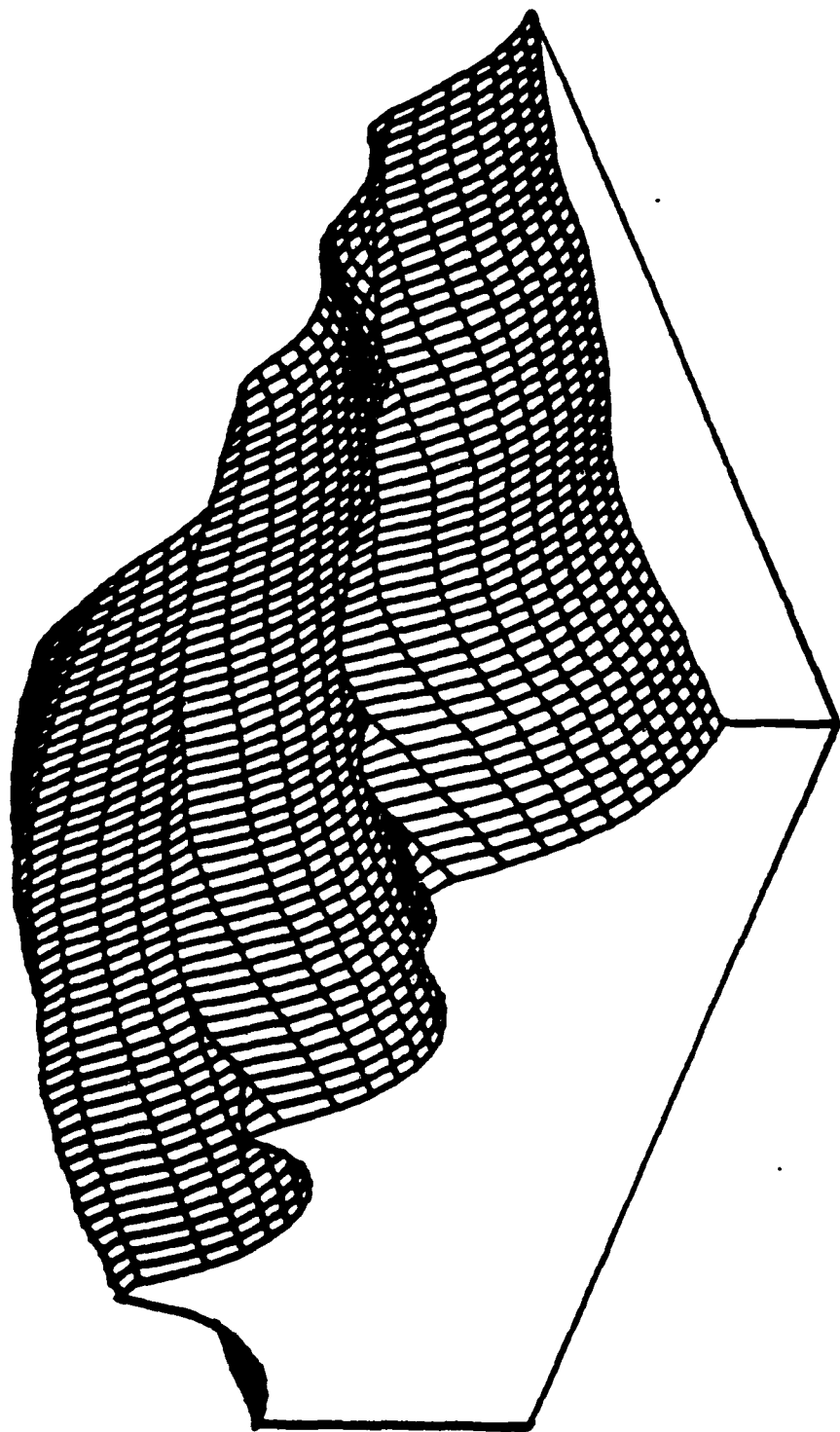


Figure IV-7.5.5.a

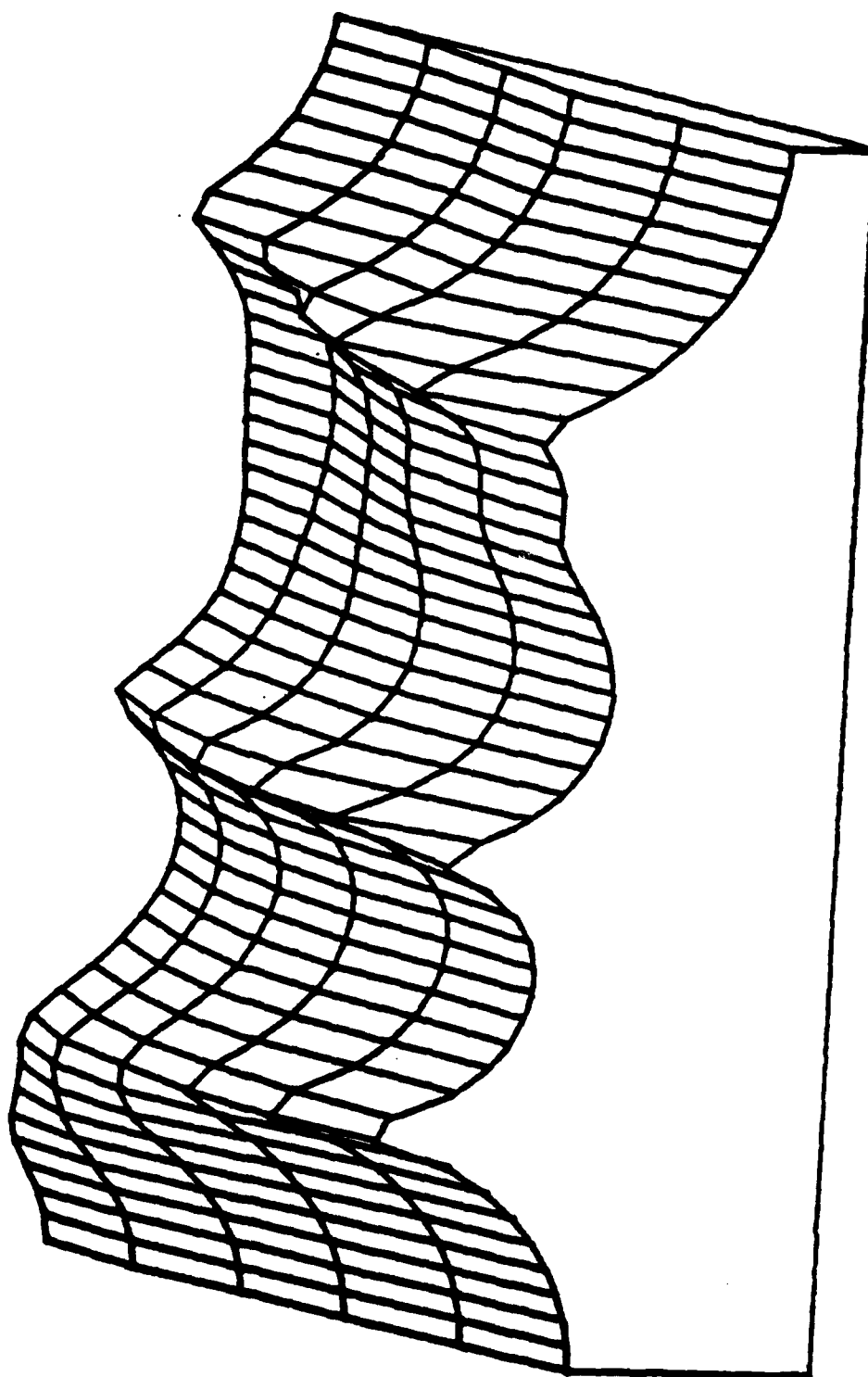


Figure IV-7.5.5.b

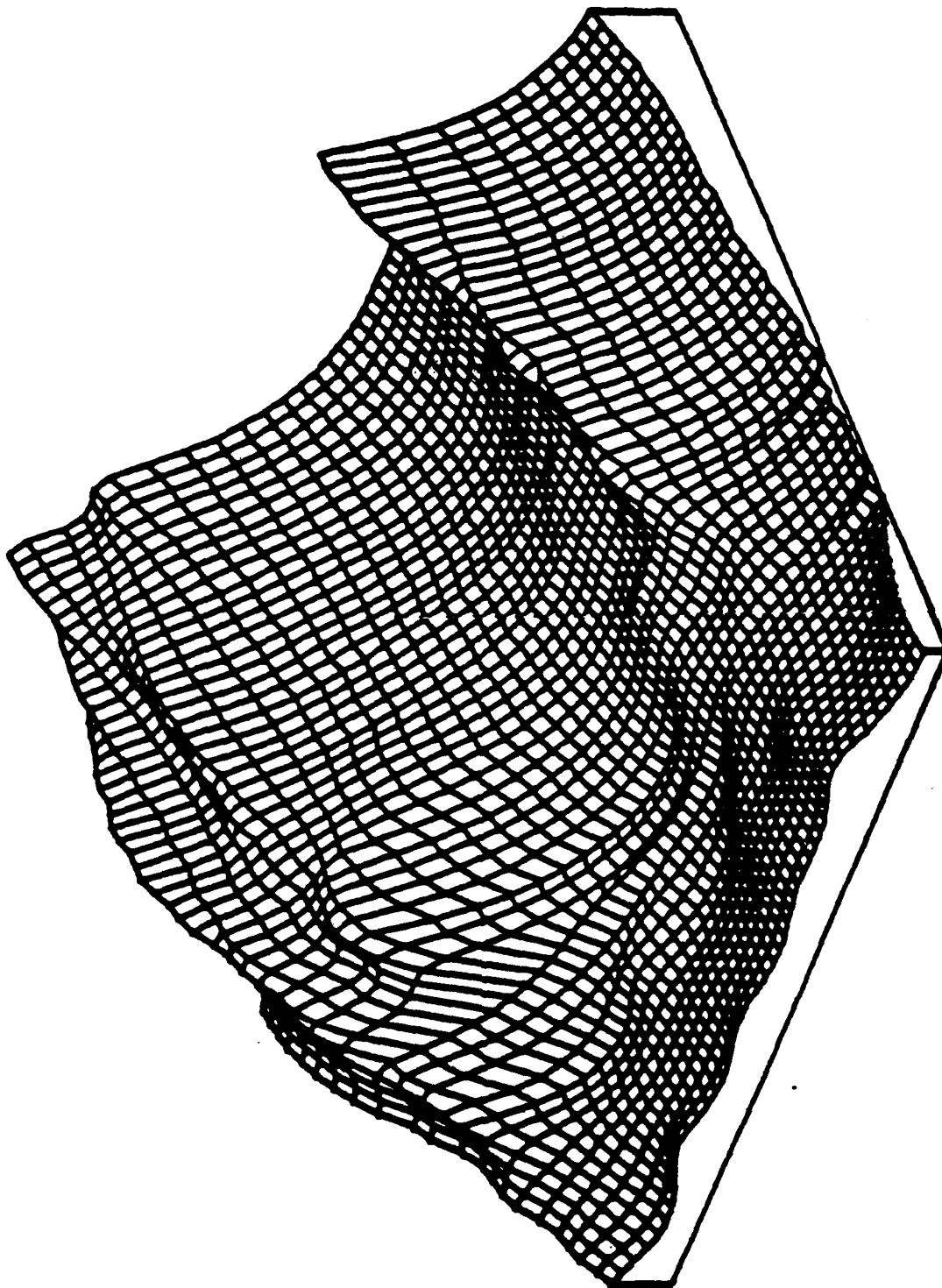


Figure IV-7.6.4.a

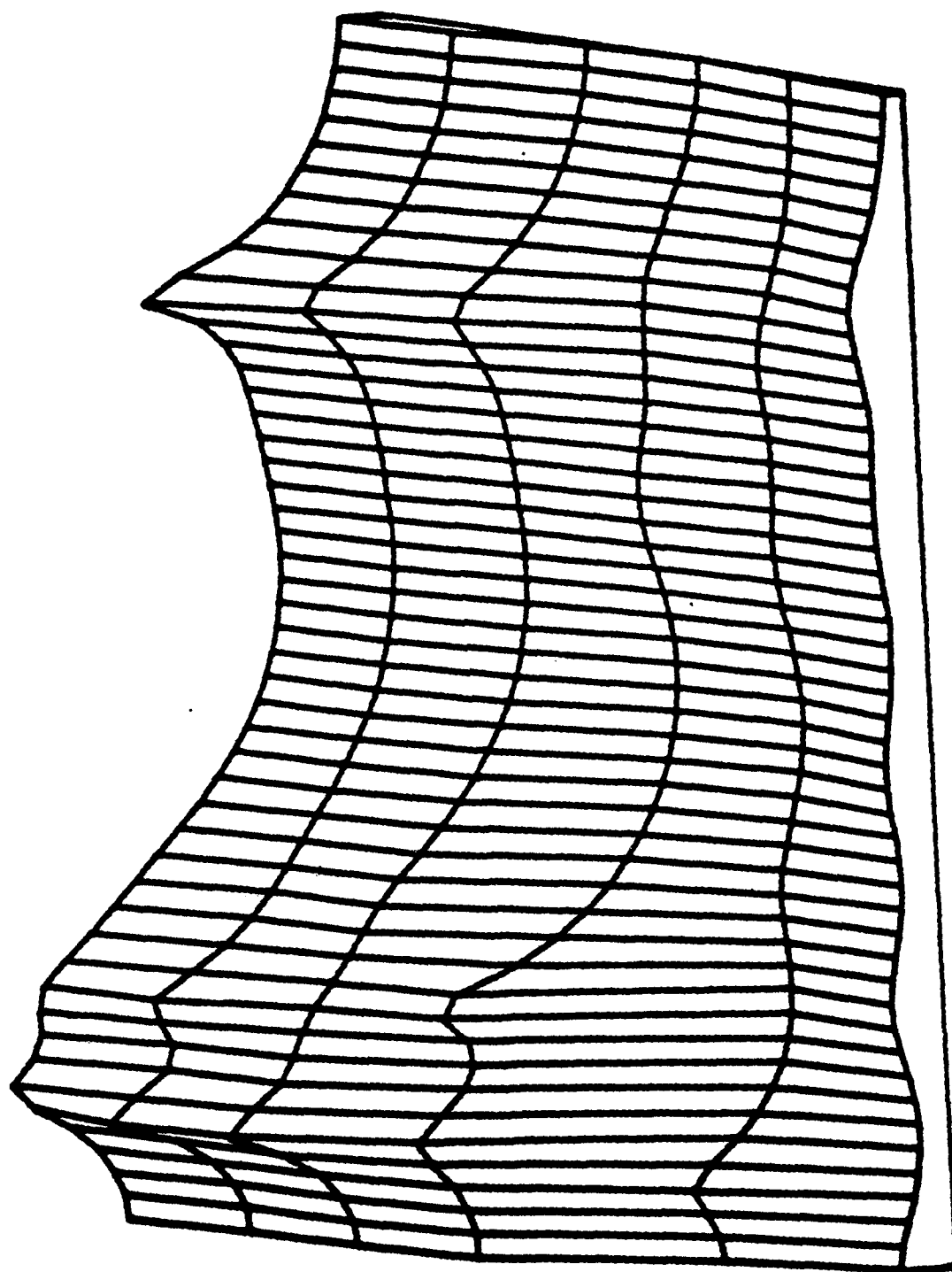


Figure IV-7.6.4.b

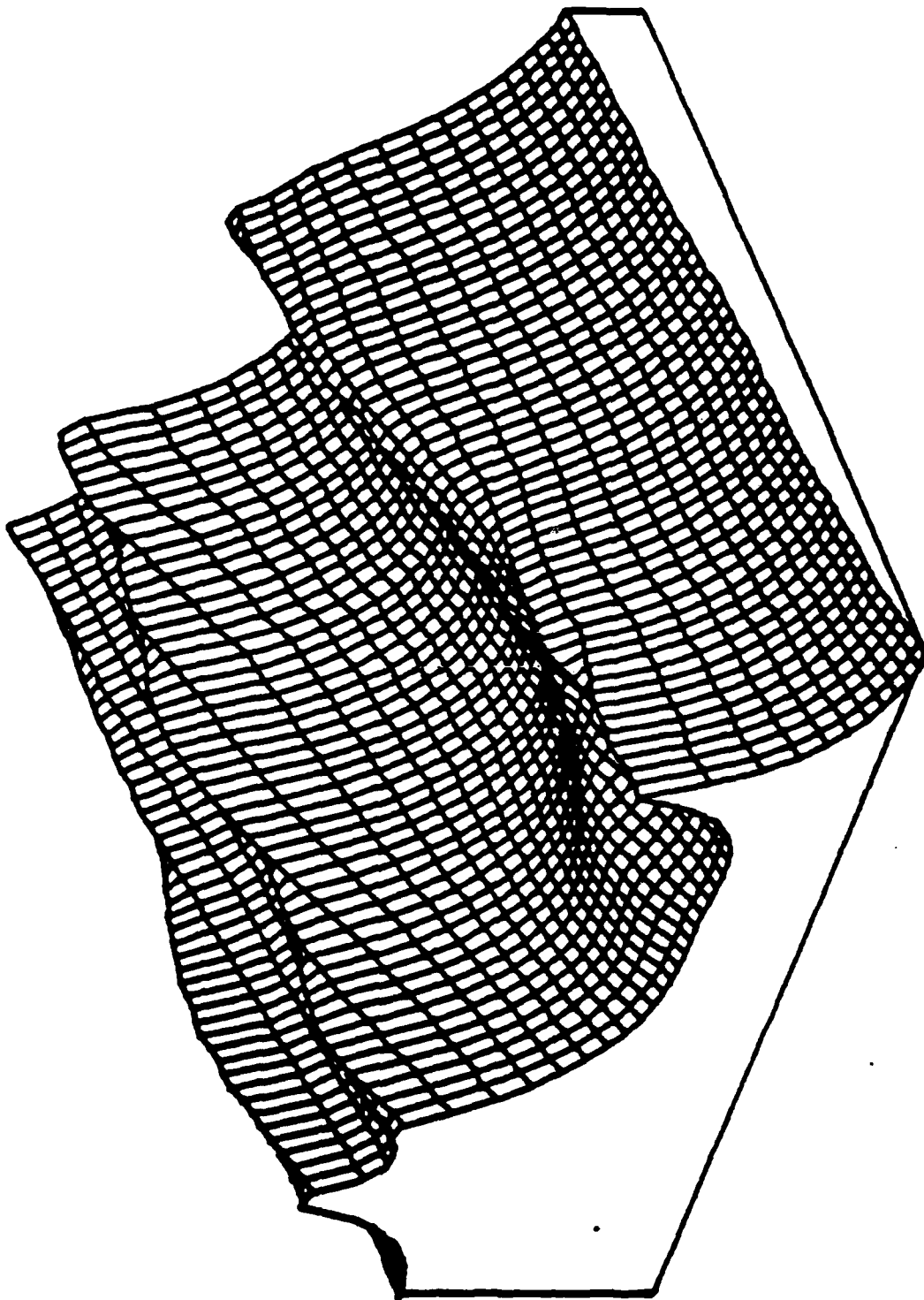


Figure IV-7.6.5.a

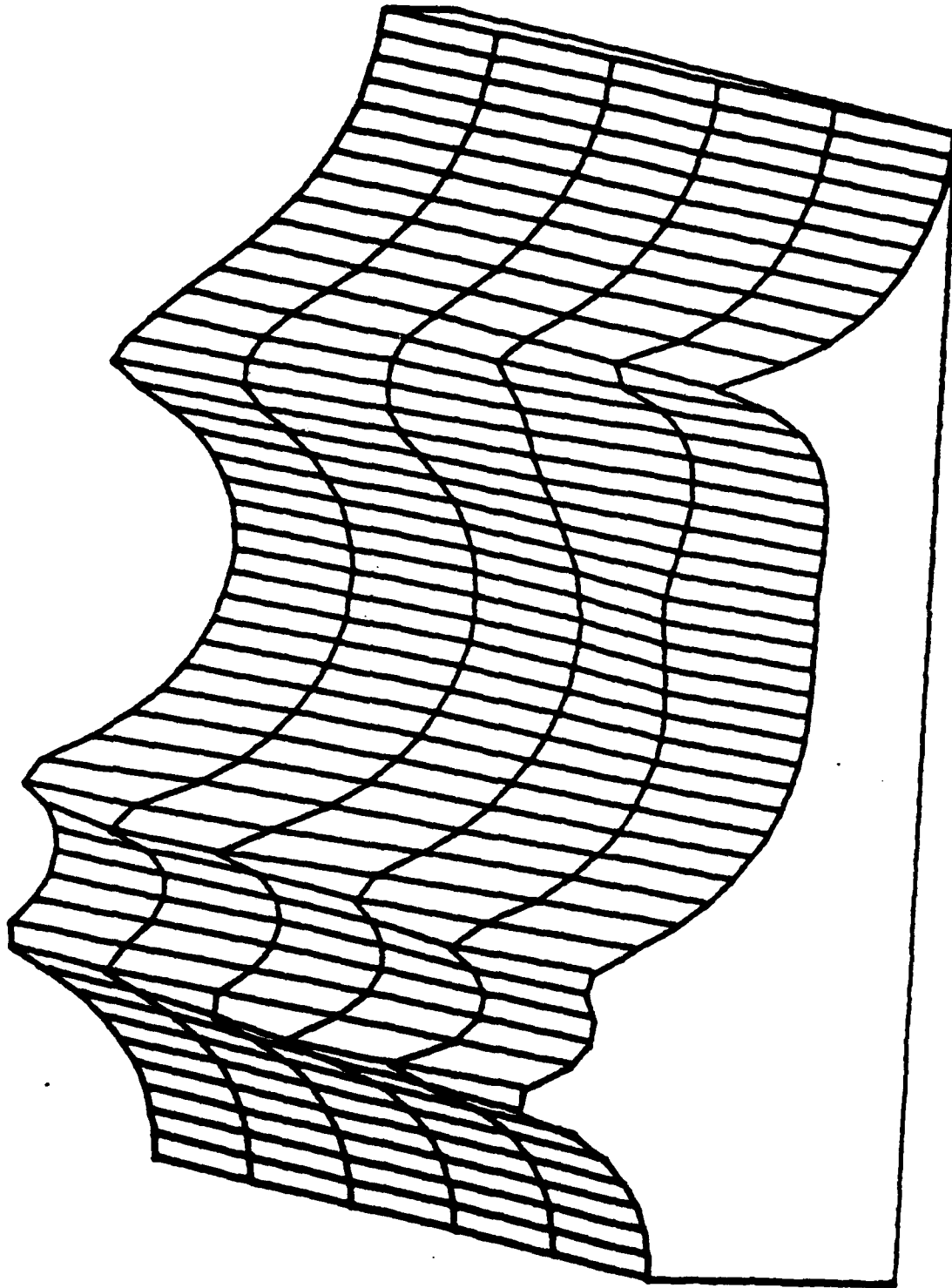


Figure IV-7.6.5.b

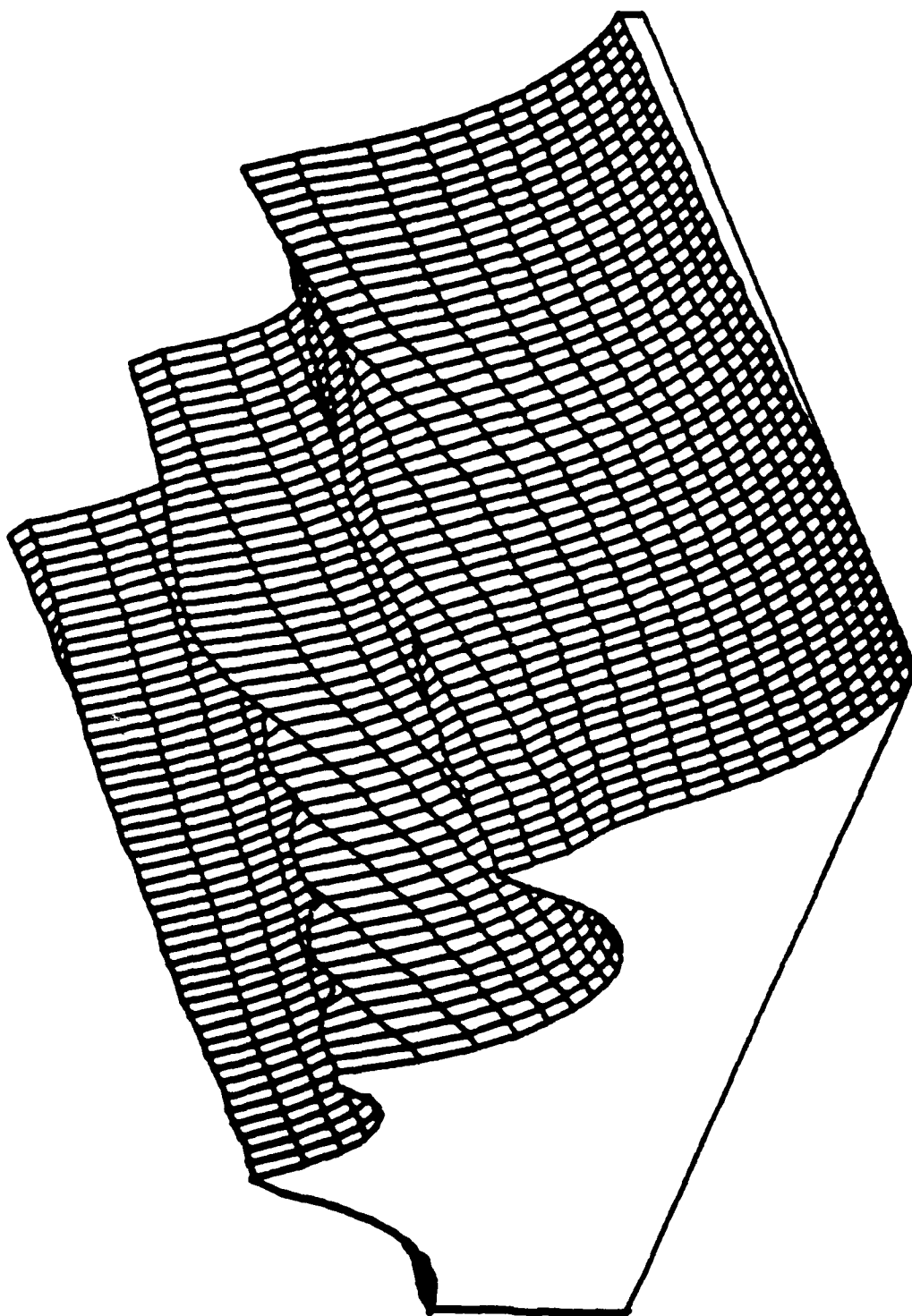


Figure IV-7.7.1.a

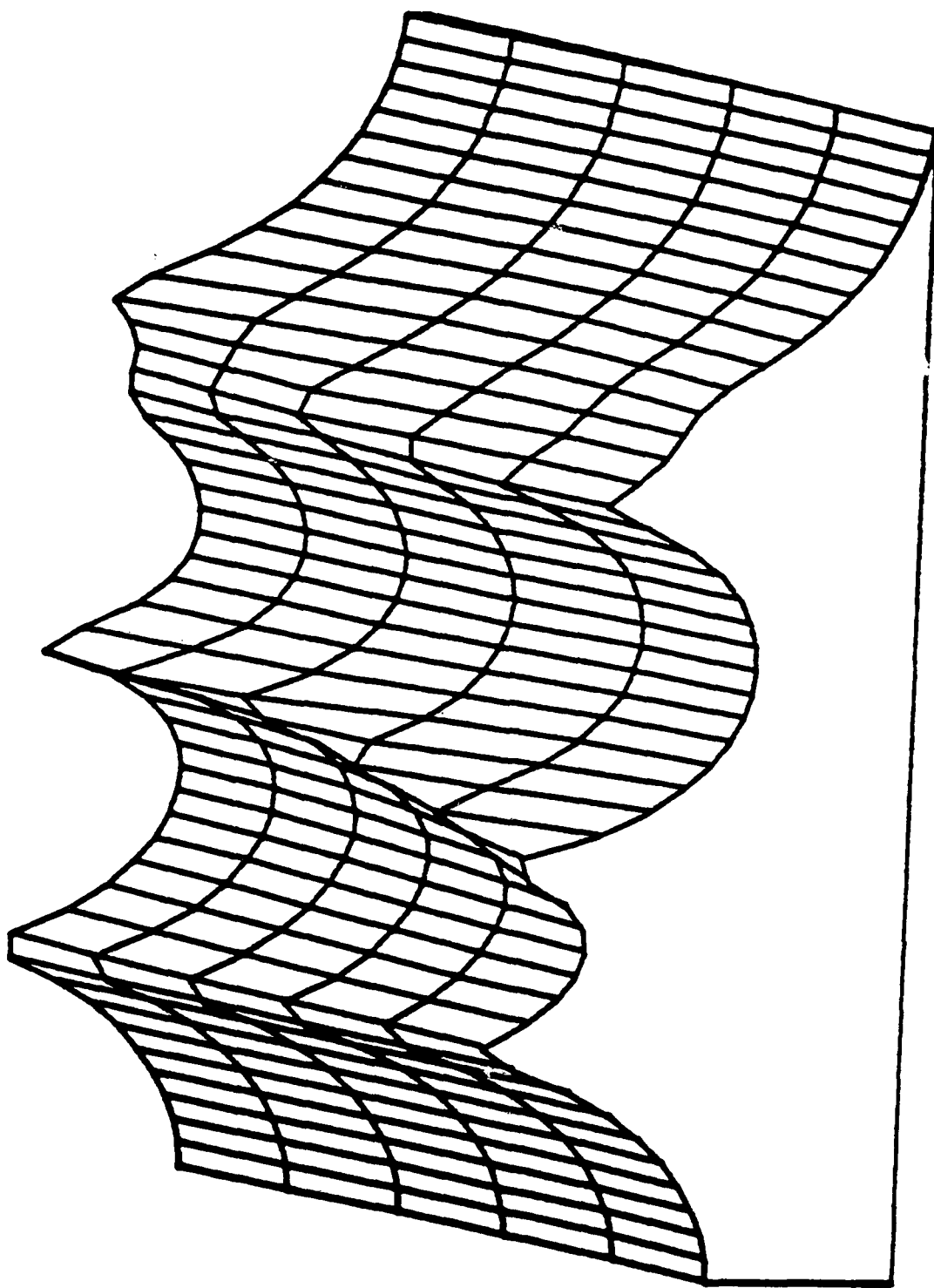


Figure IV-7.7.1.b

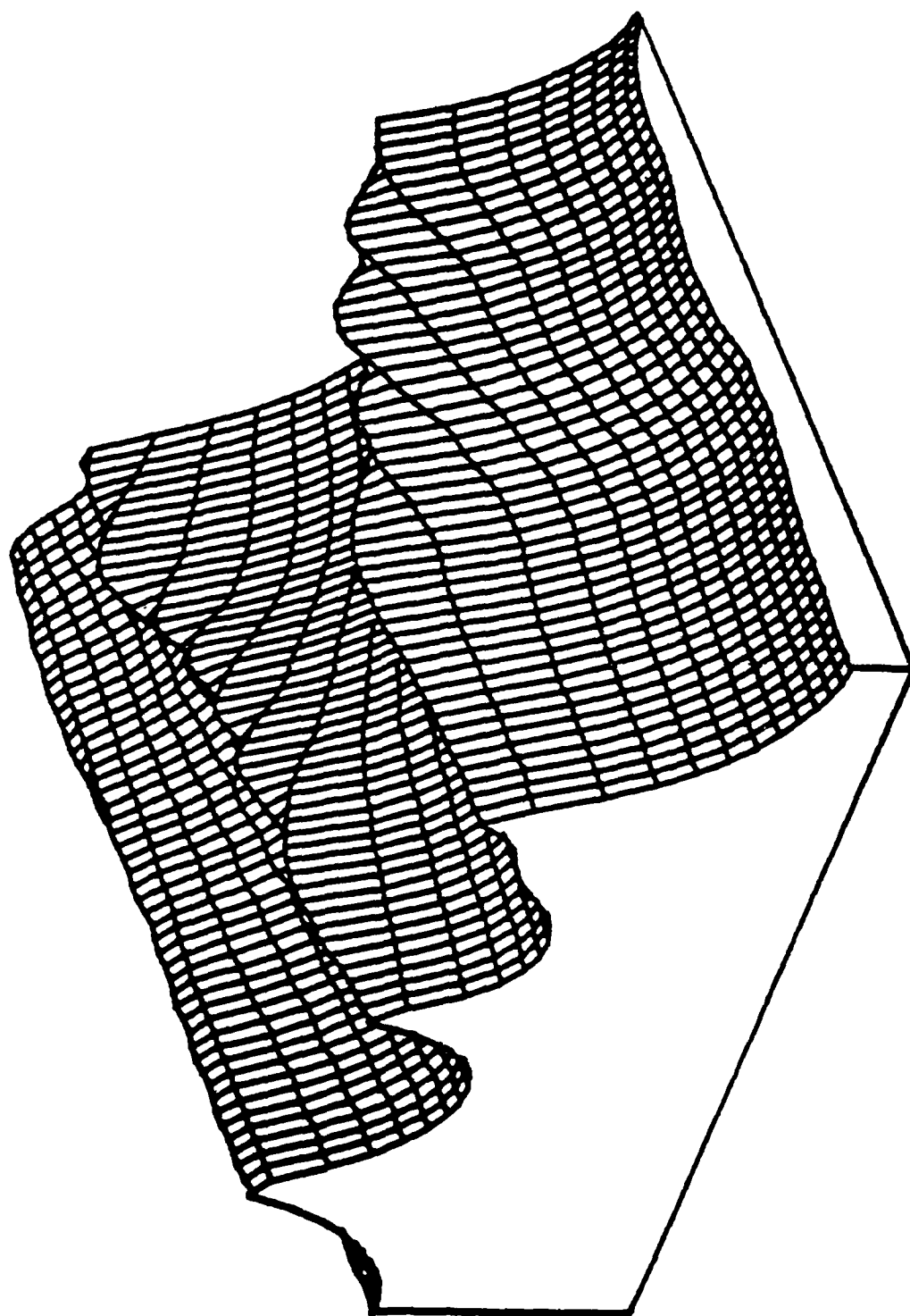


Figure IV-7.7.2.a

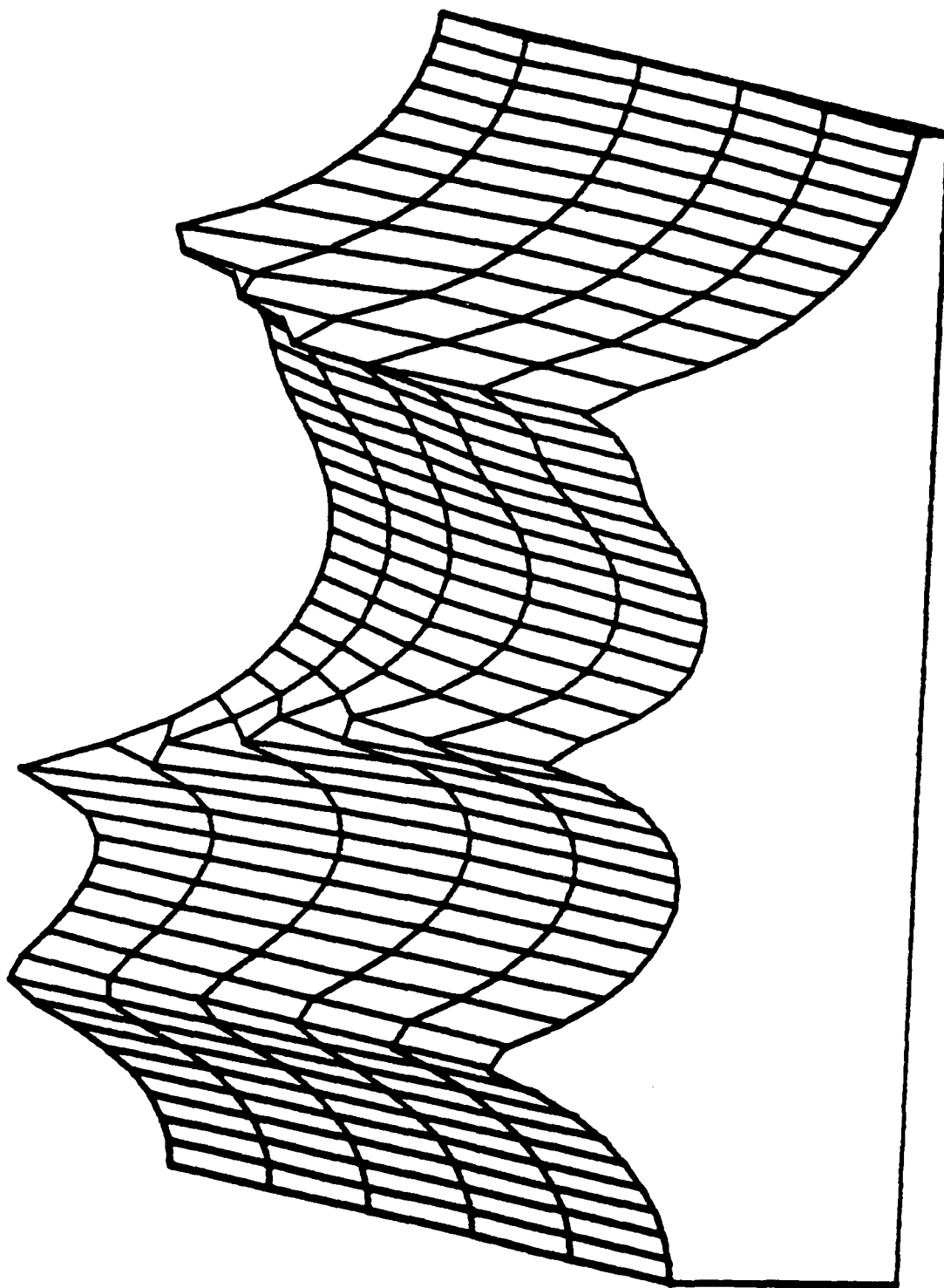


Figure IV-7.7.2.b

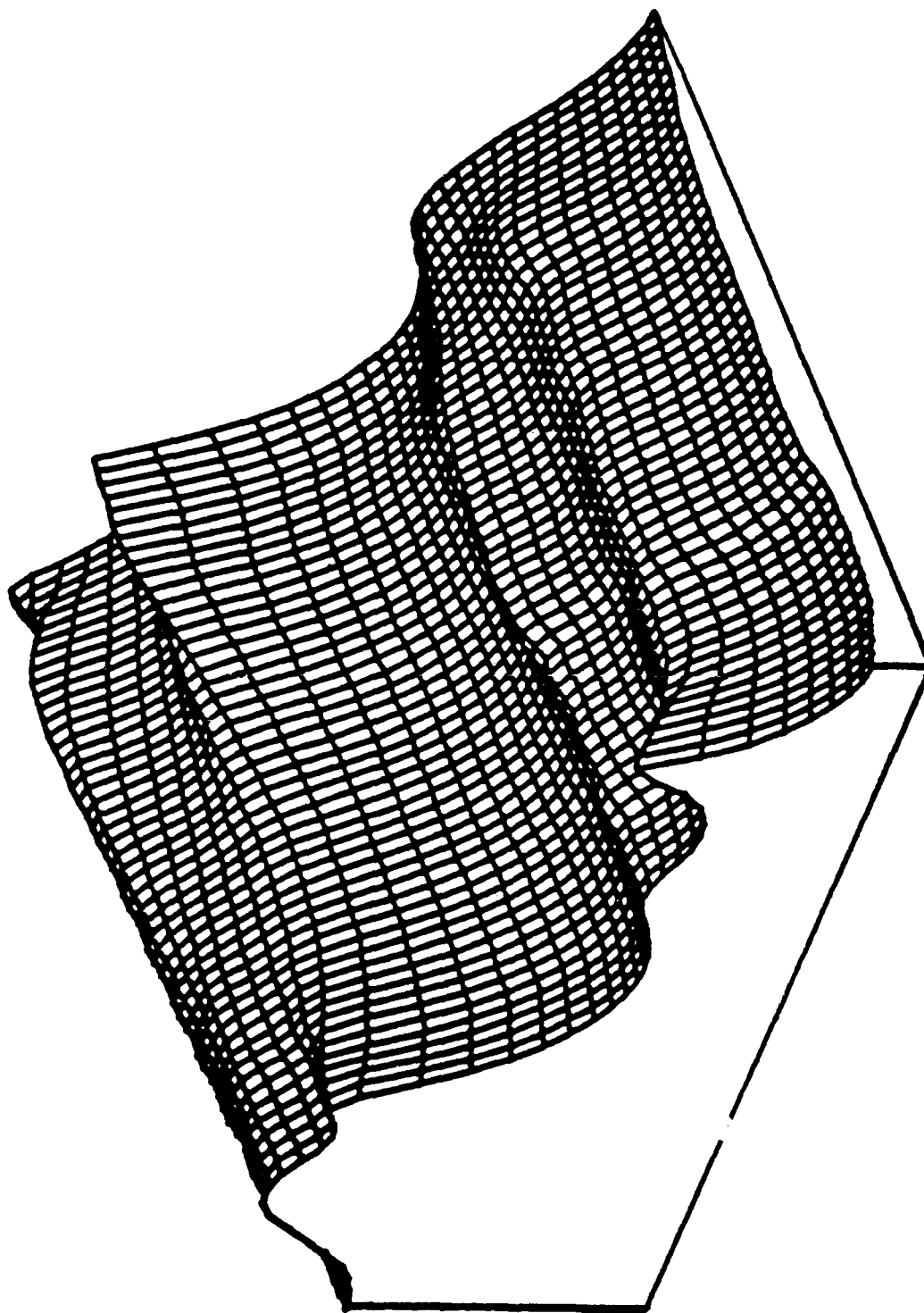


Figure IV-7.7.3.a

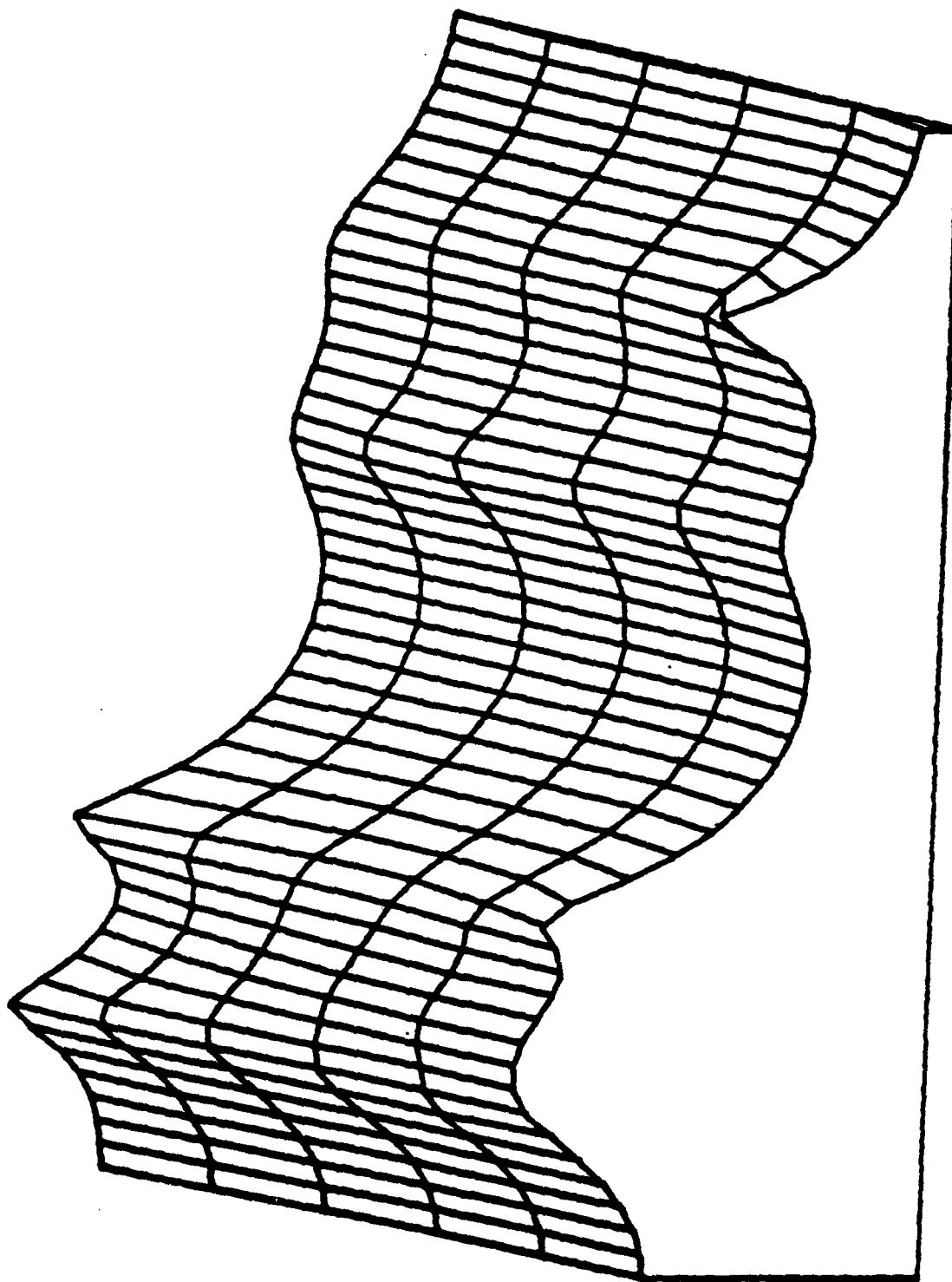


Figure IV-7.7.3.b

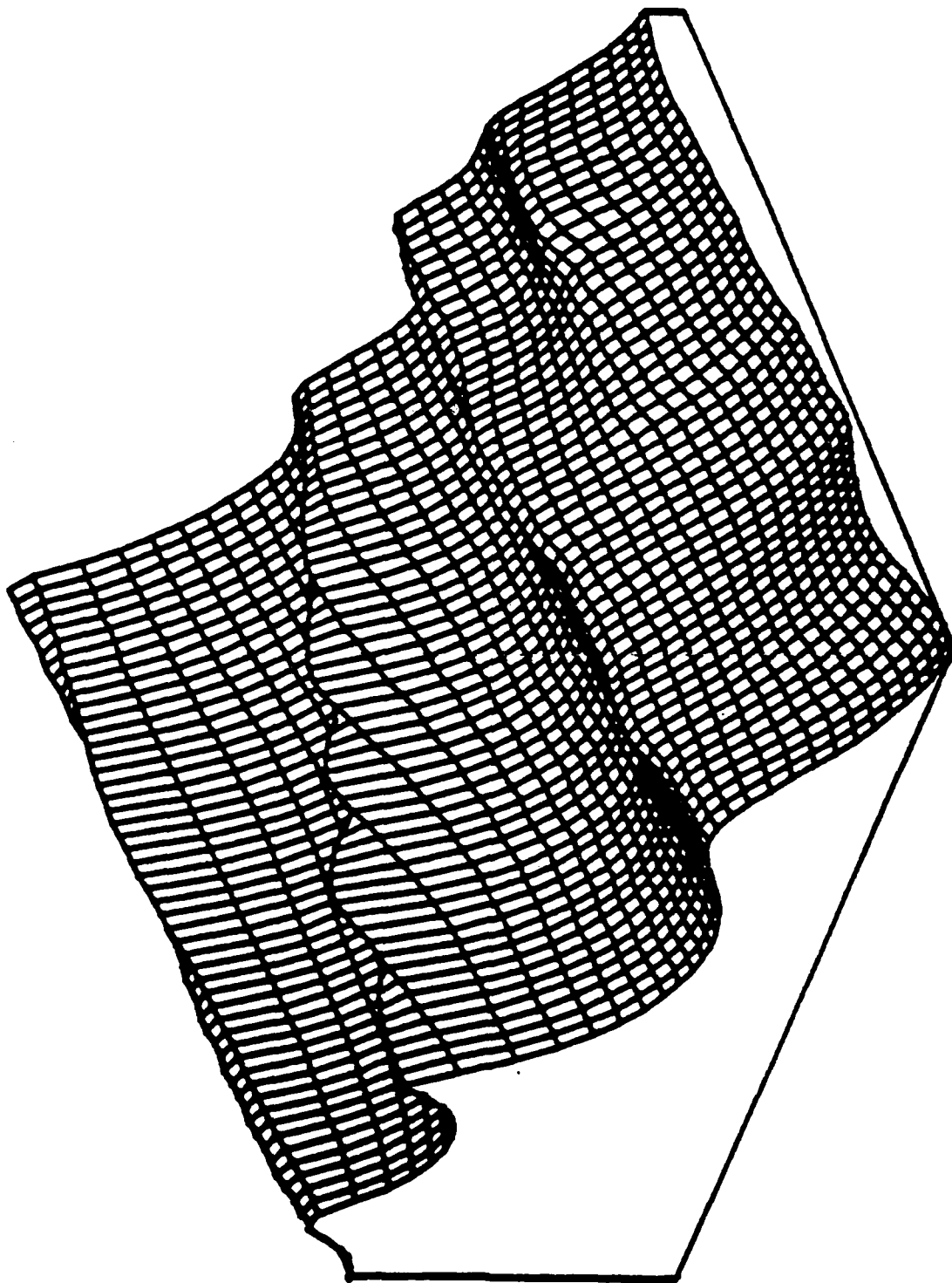


Figure IV-7.7.4.a

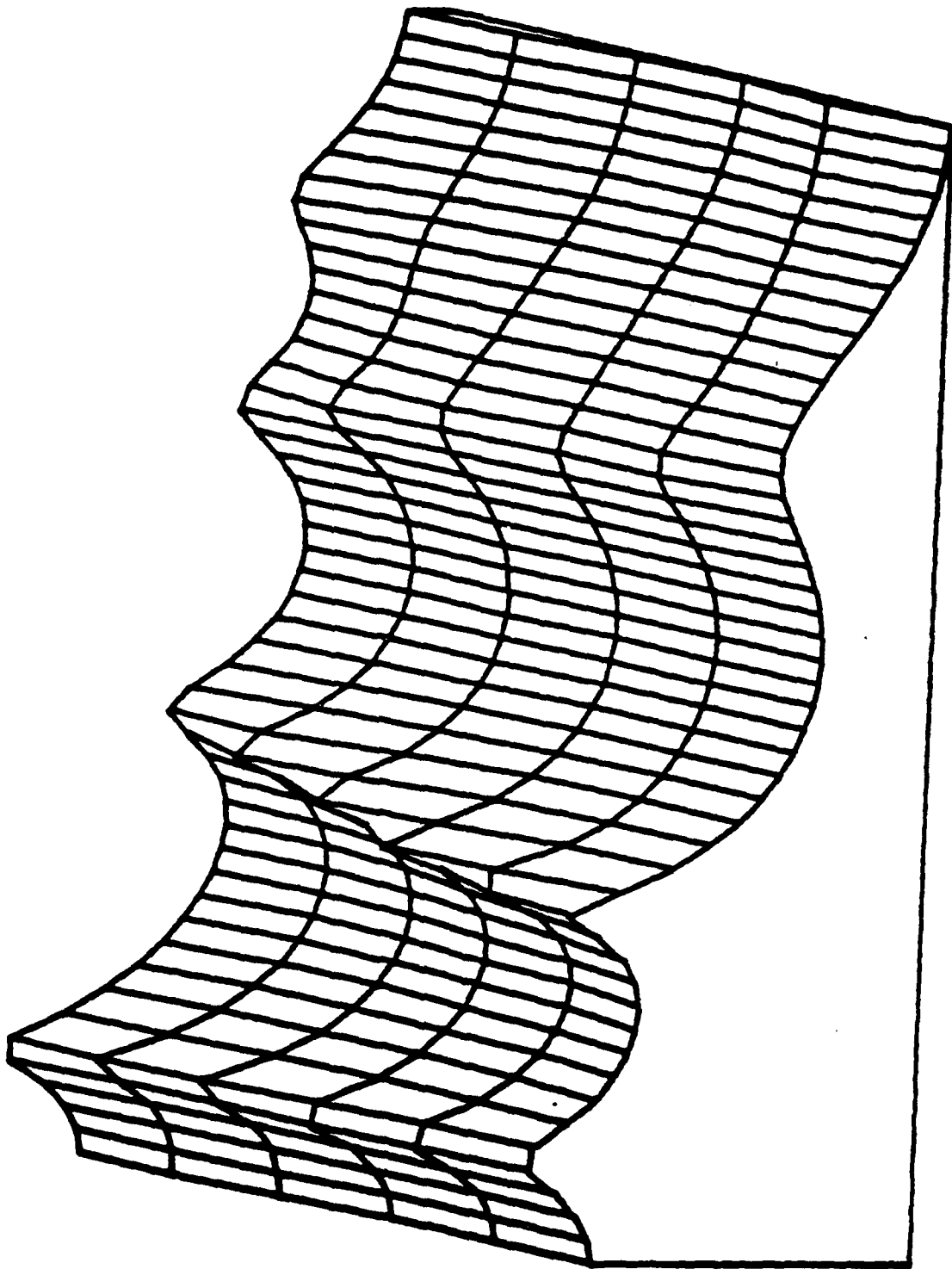


Figure IV-7.7.4.b

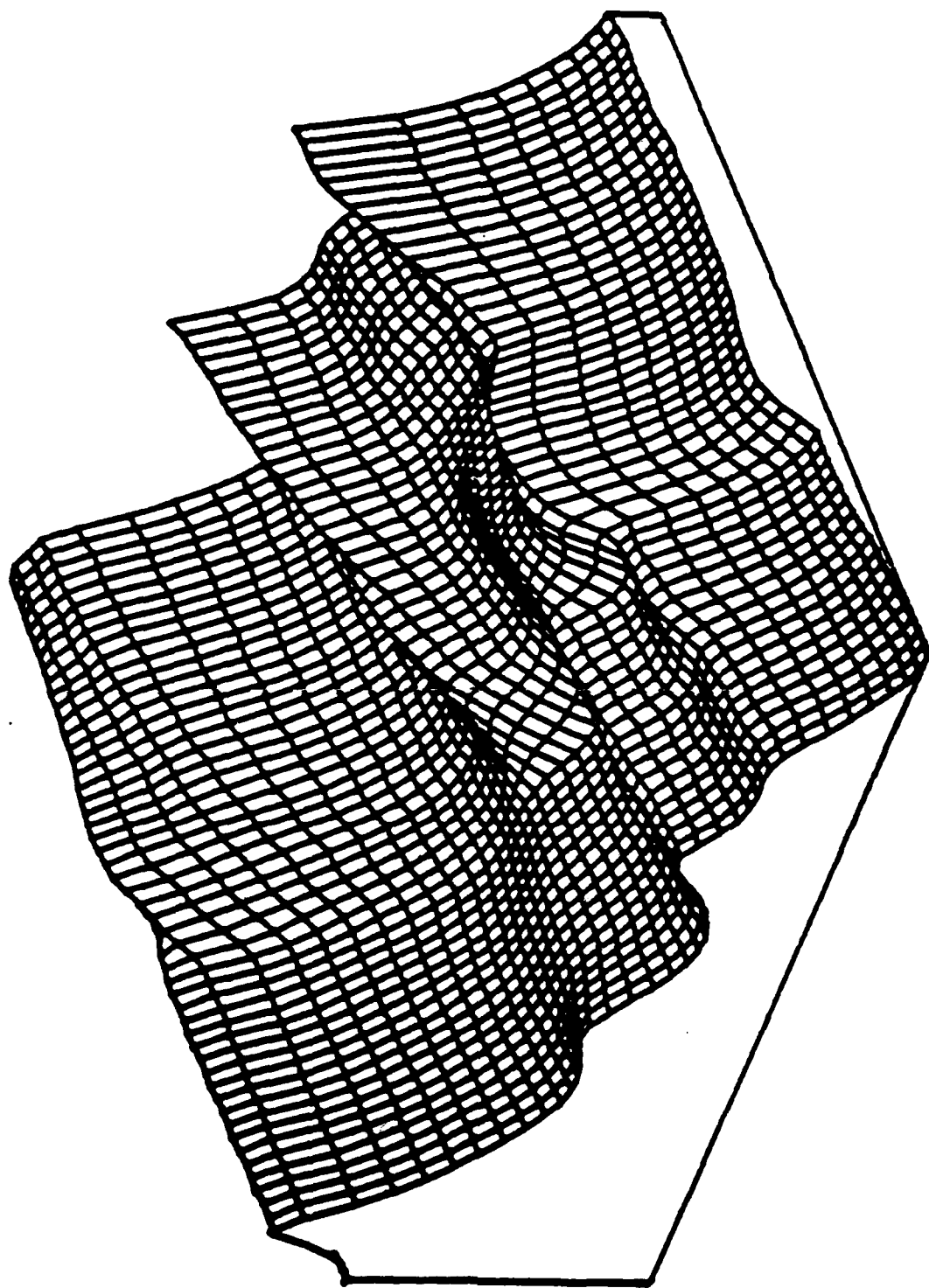


Figure IV-7.7.5.a

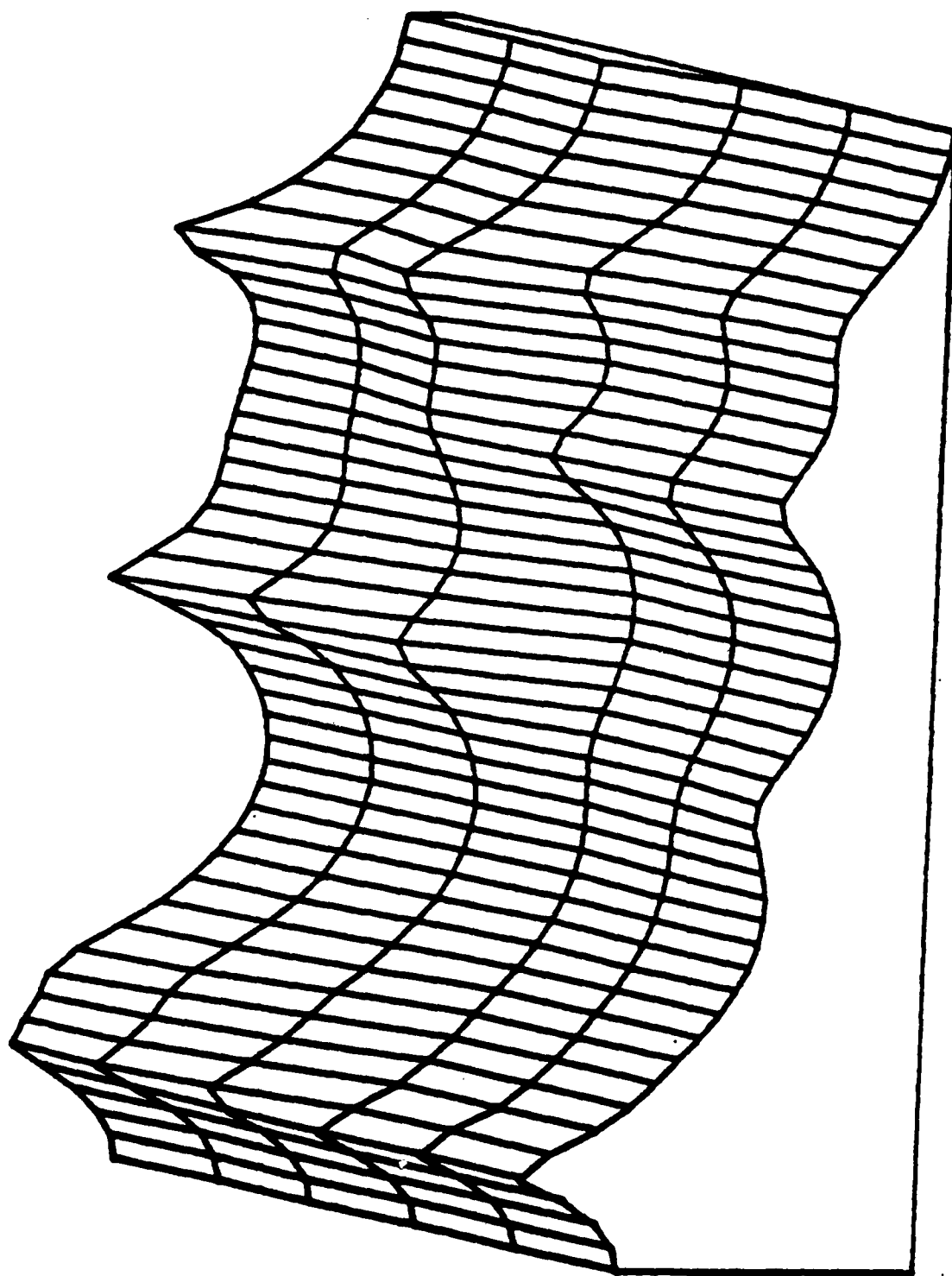


Figure IV-7.7.5.b

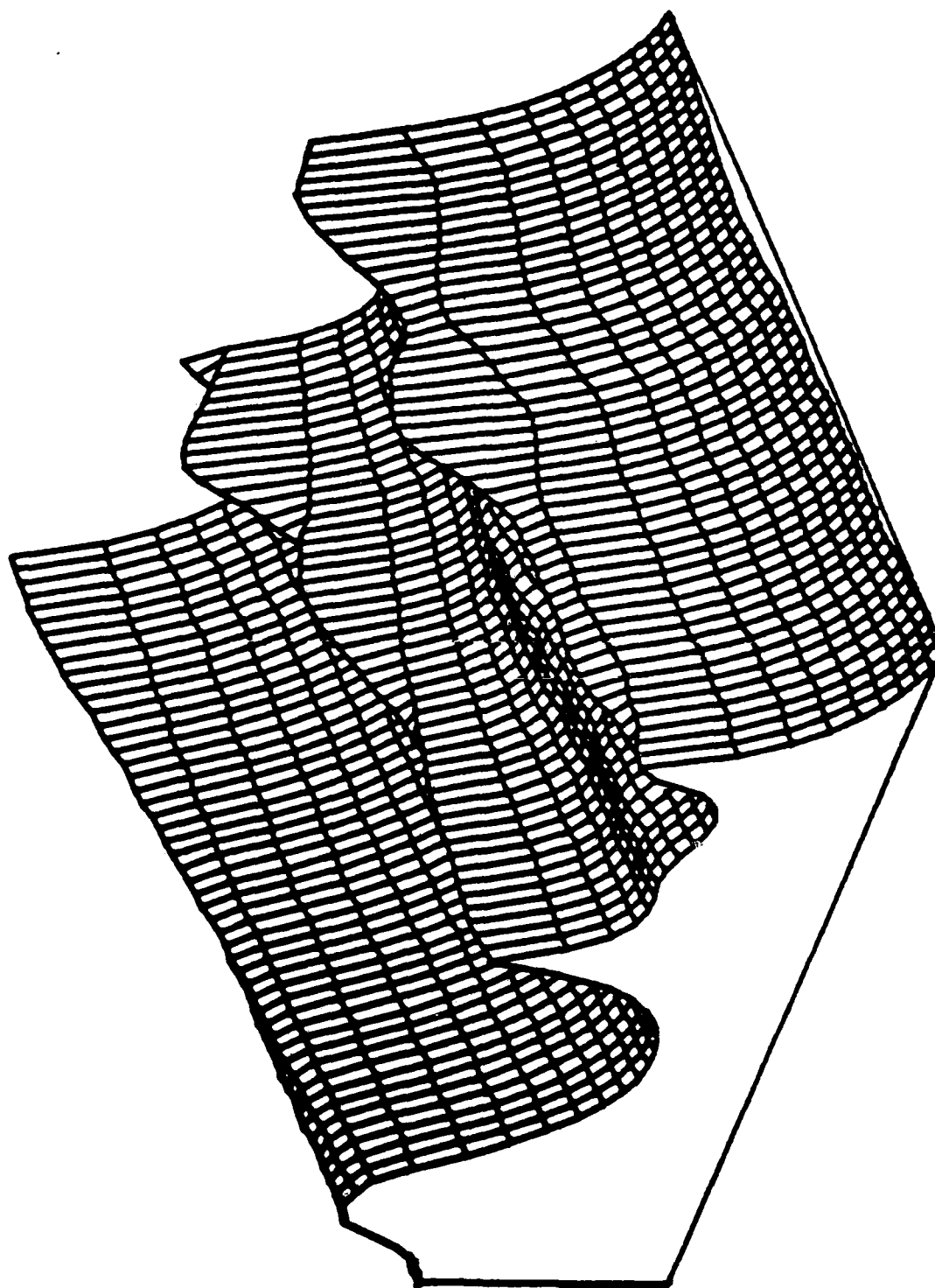


Figure IV-7.8.1.a

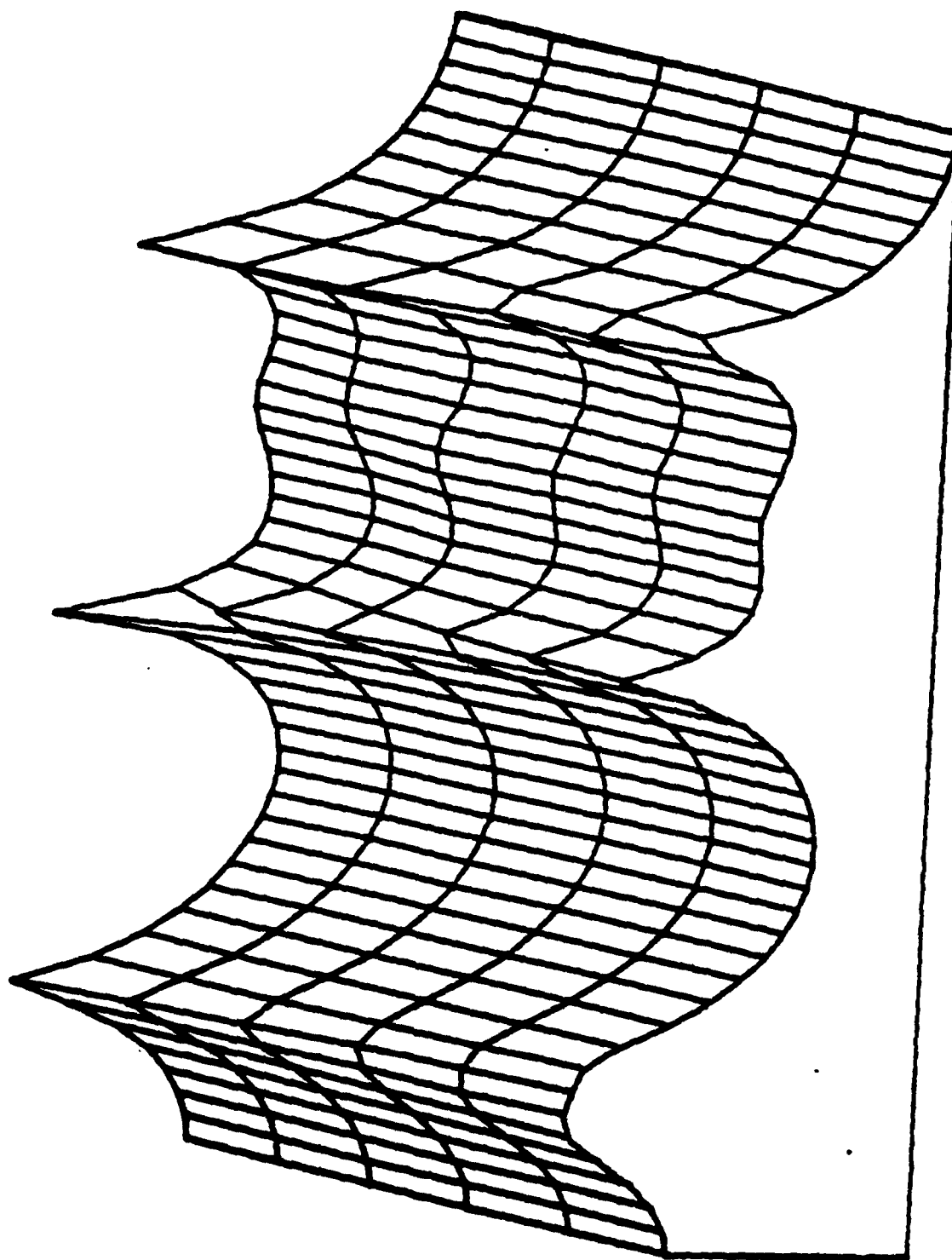


Figure IV-7.8.1.b

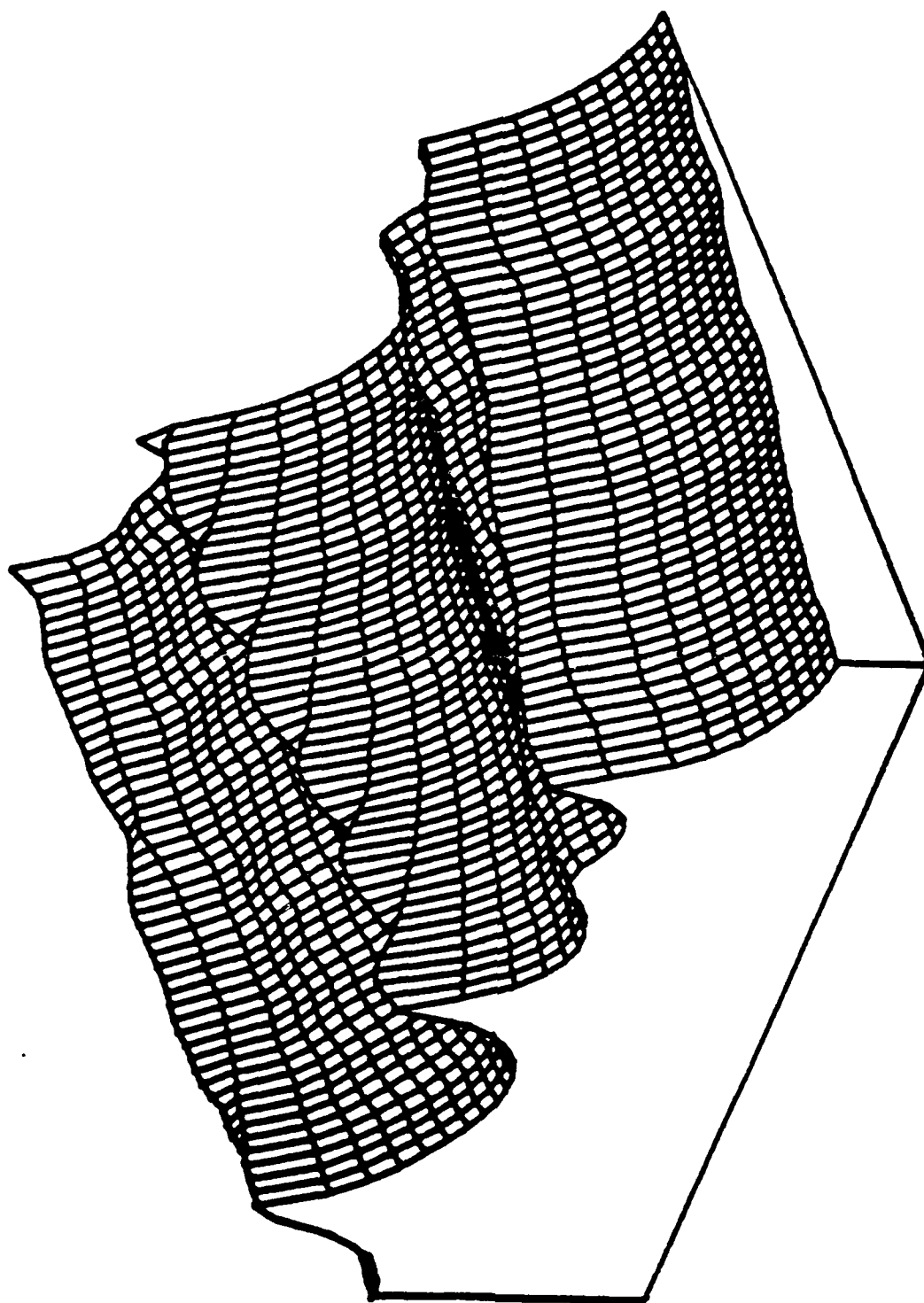


Figure IV-7.8.2.a

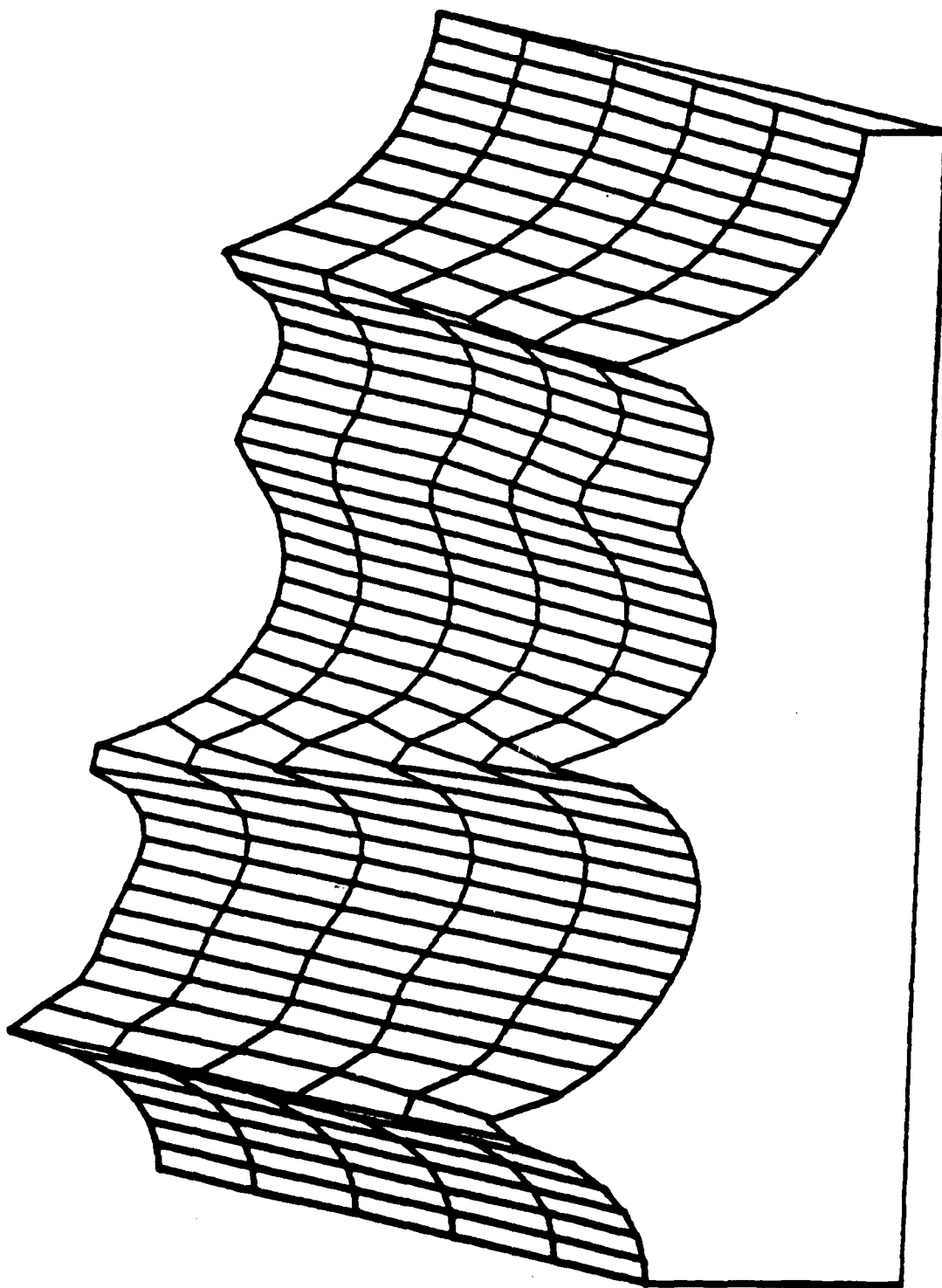


Figure IV-7.8.2.b

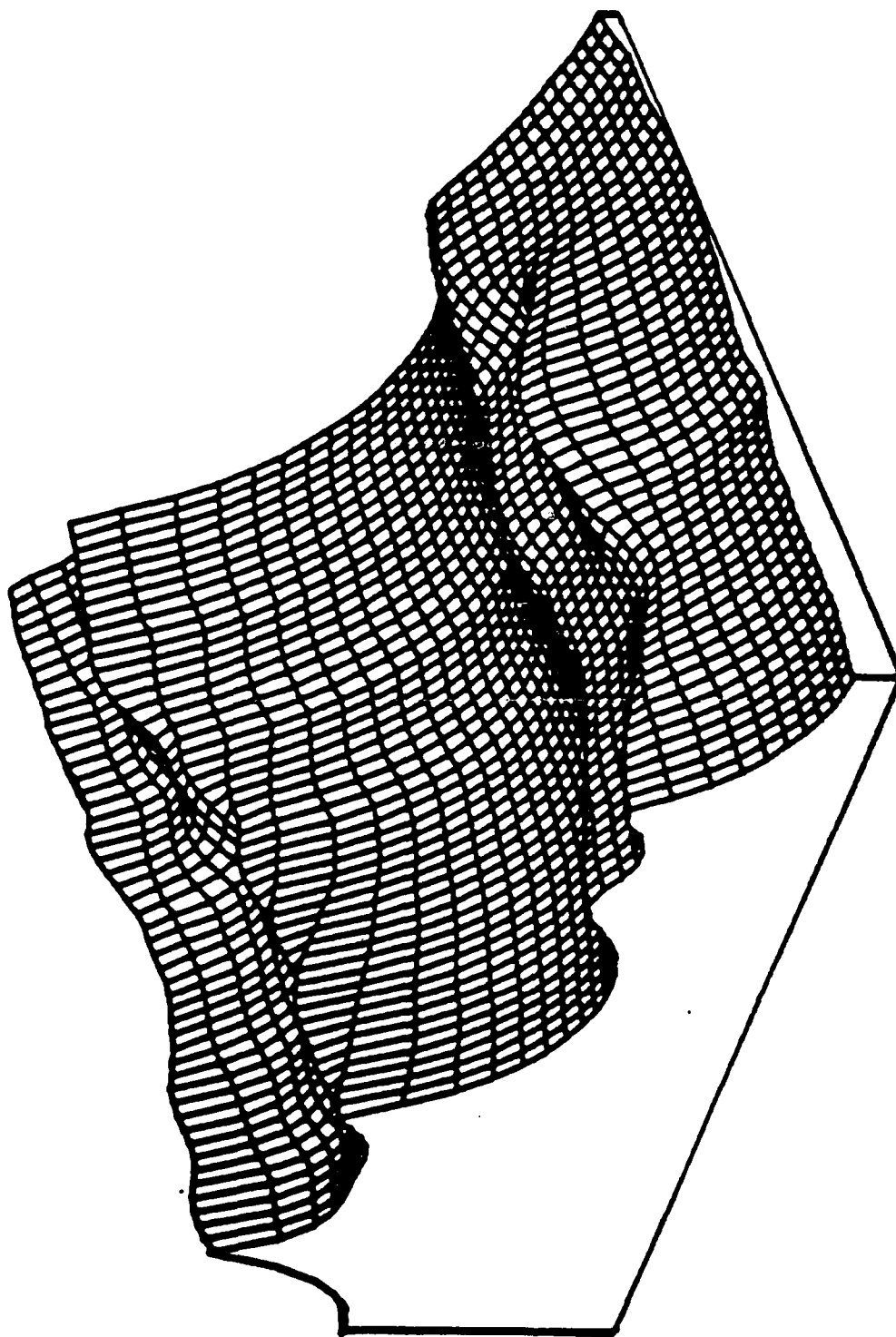


Figure IV-7.8.3.a

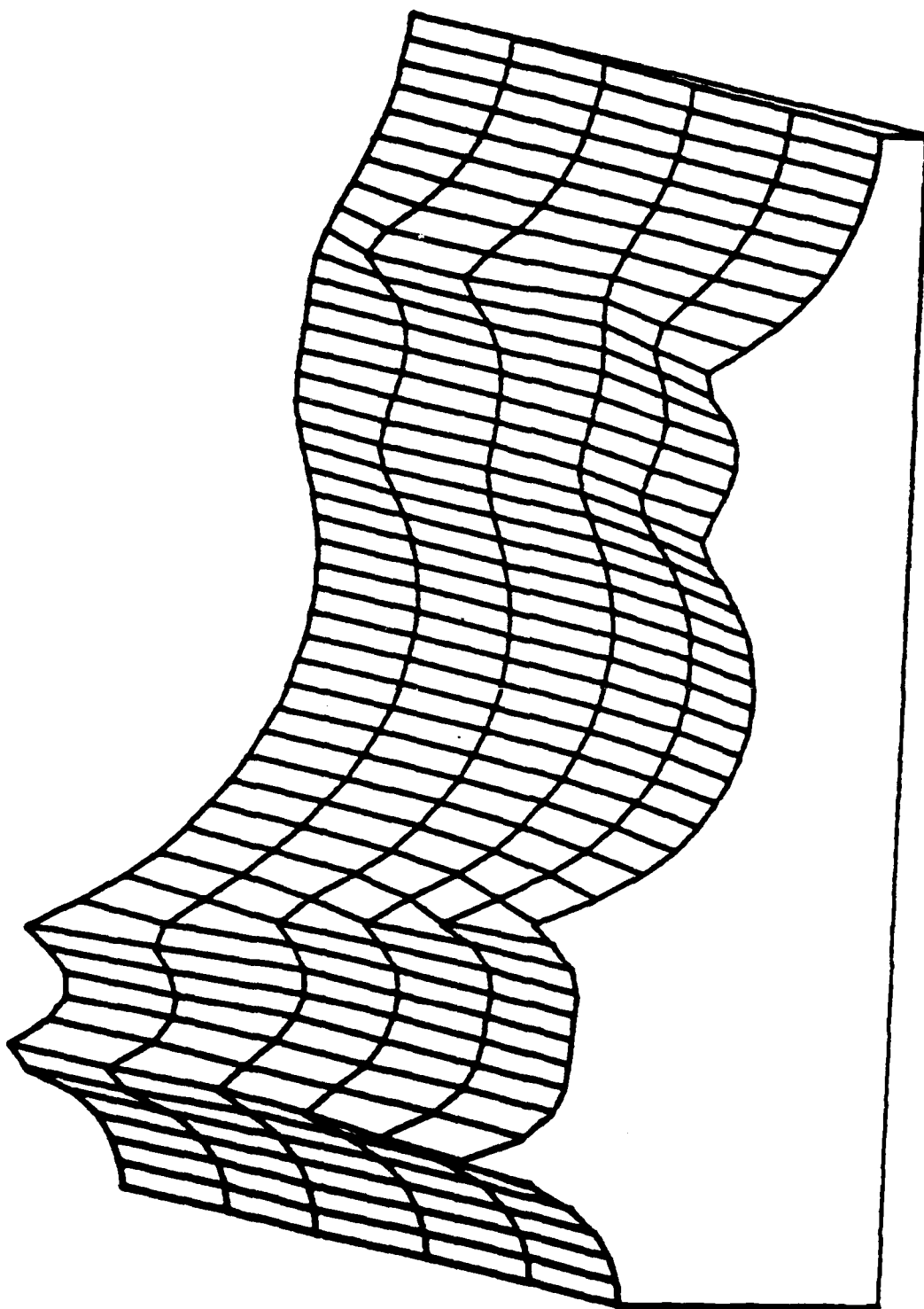


Figure IV-7.8.3.b

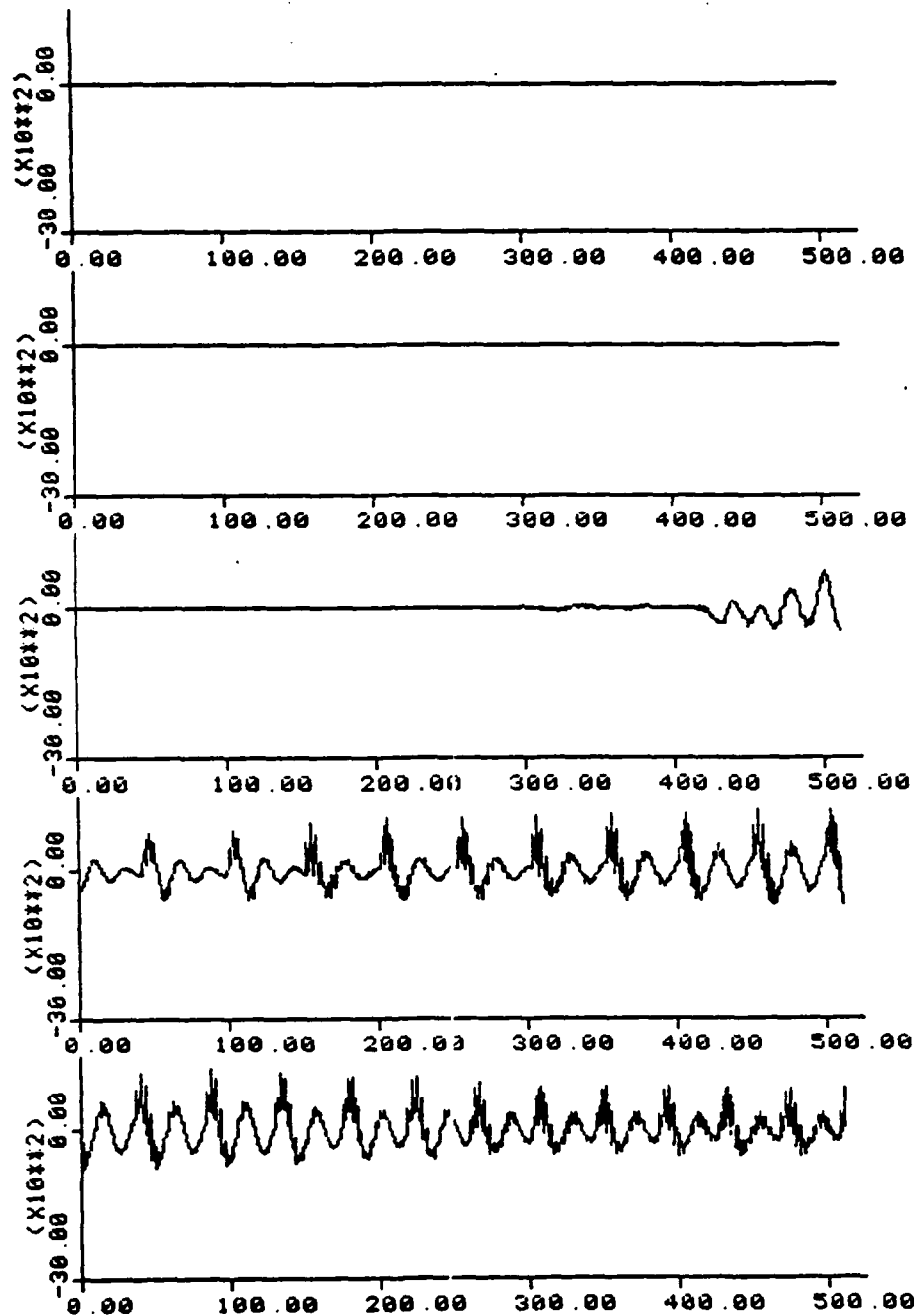


Figure IV-8.1 Original speech file
"Thieves who rob friends deserve jail"

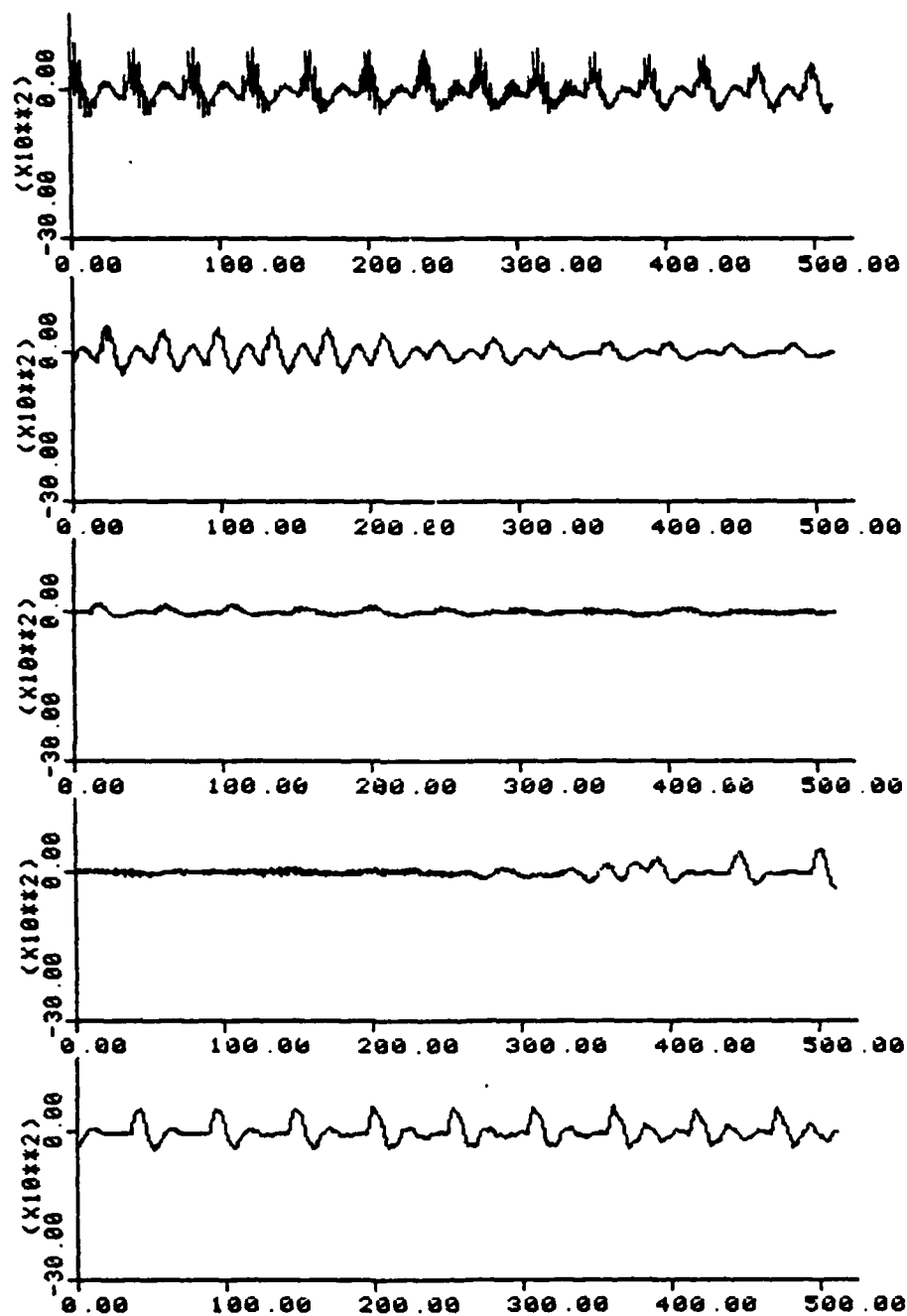


Figure IV-8.2 "Thieves who rob friends deserve jail"

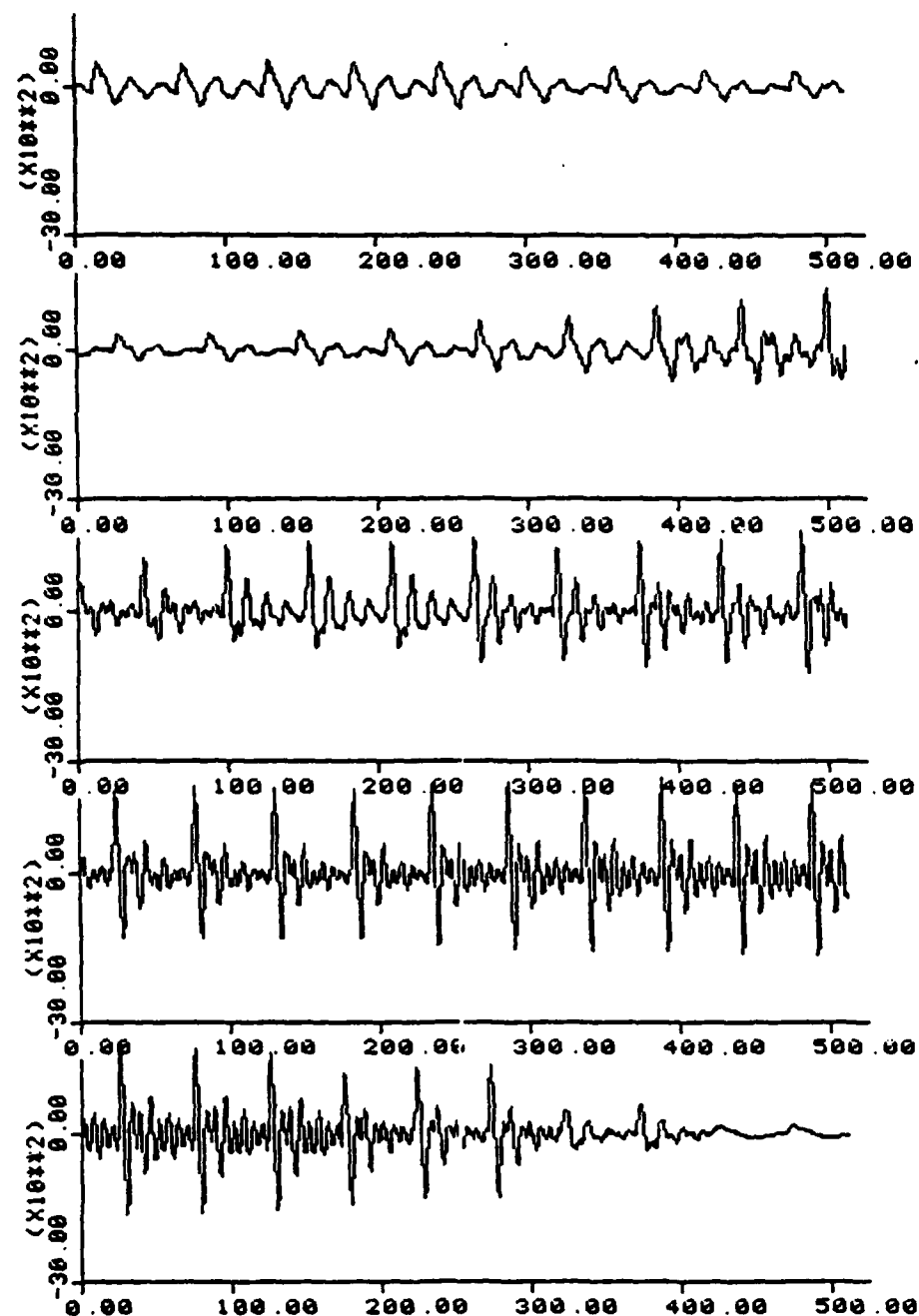


Figure IV-8.3 "Thieves who rob friends deserve jail"

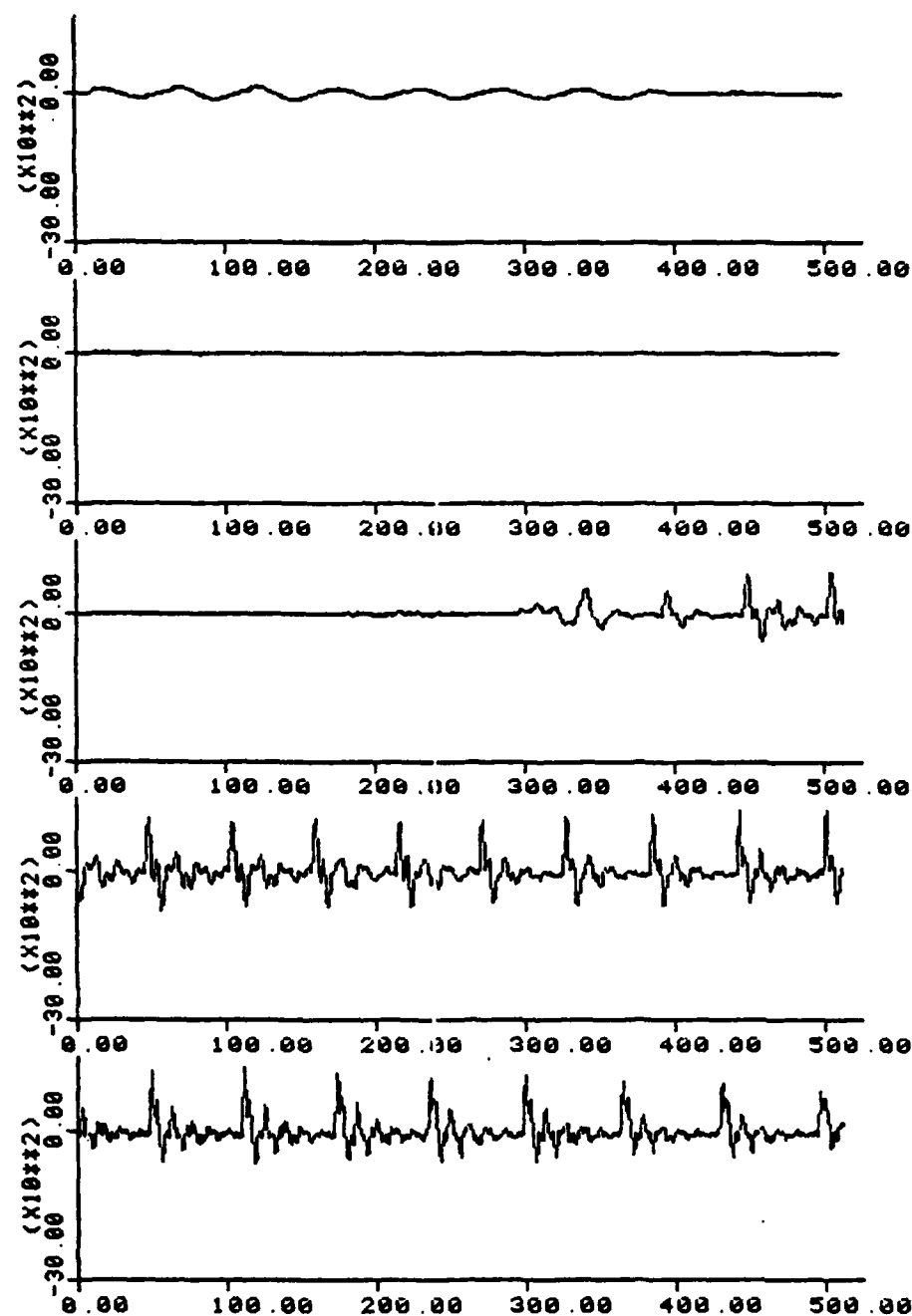


Figure IV-8.4 "Thieves who rob friends deserve jail"

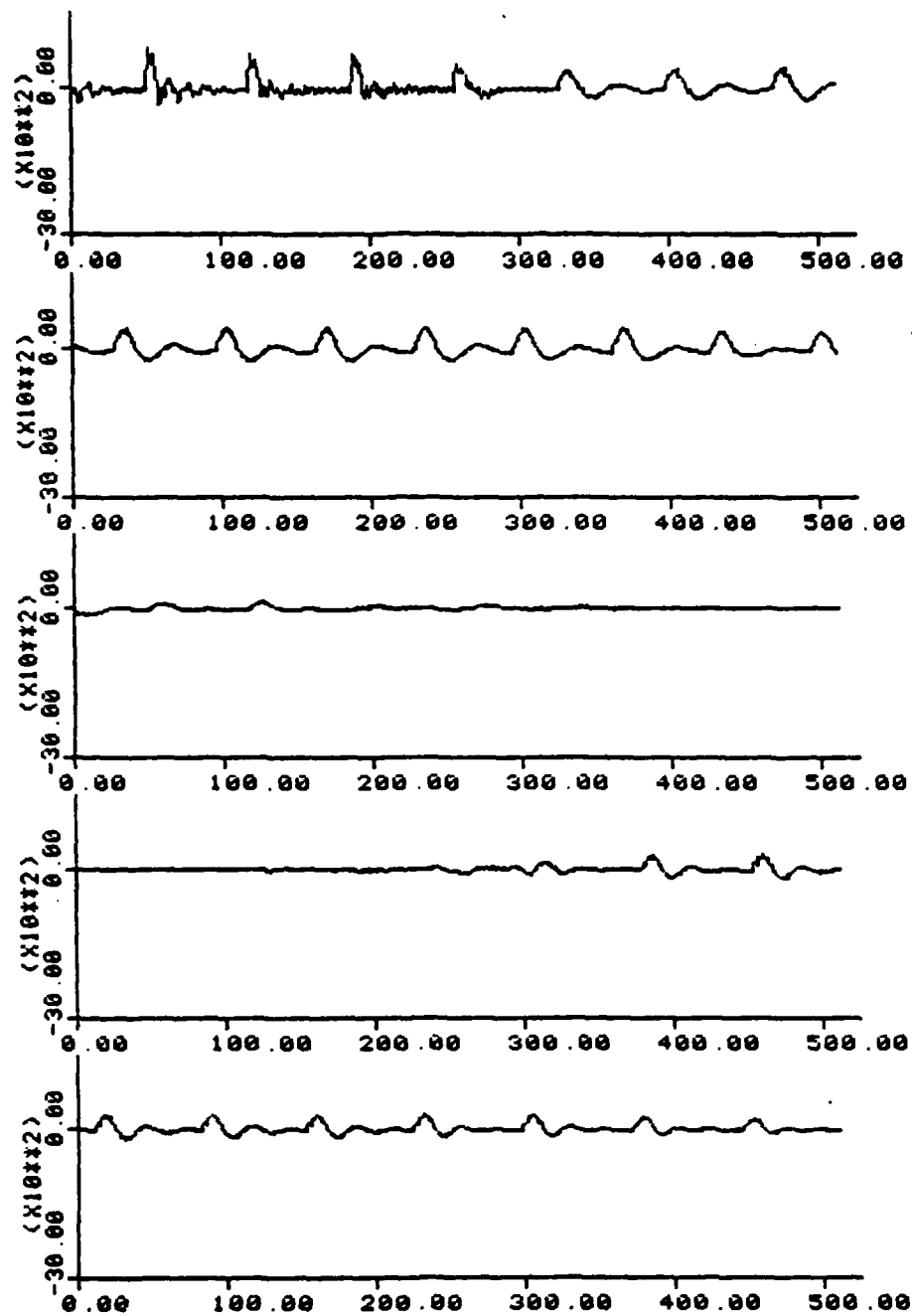


Figure IV-8.5 "Thieves who rob friends deserve jail"

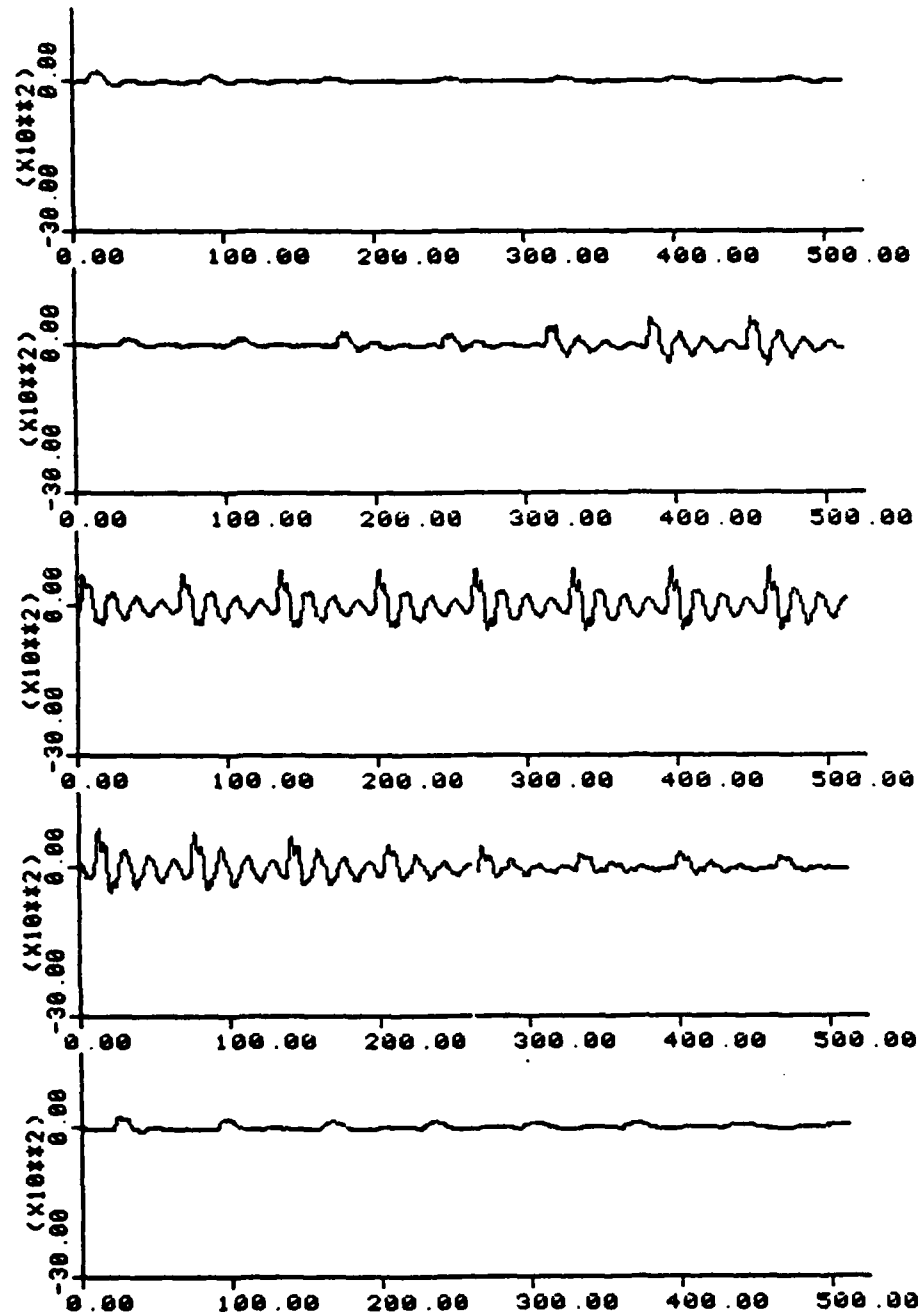


Figure IV-8.6 "Thieves who rob friends deserve jail"

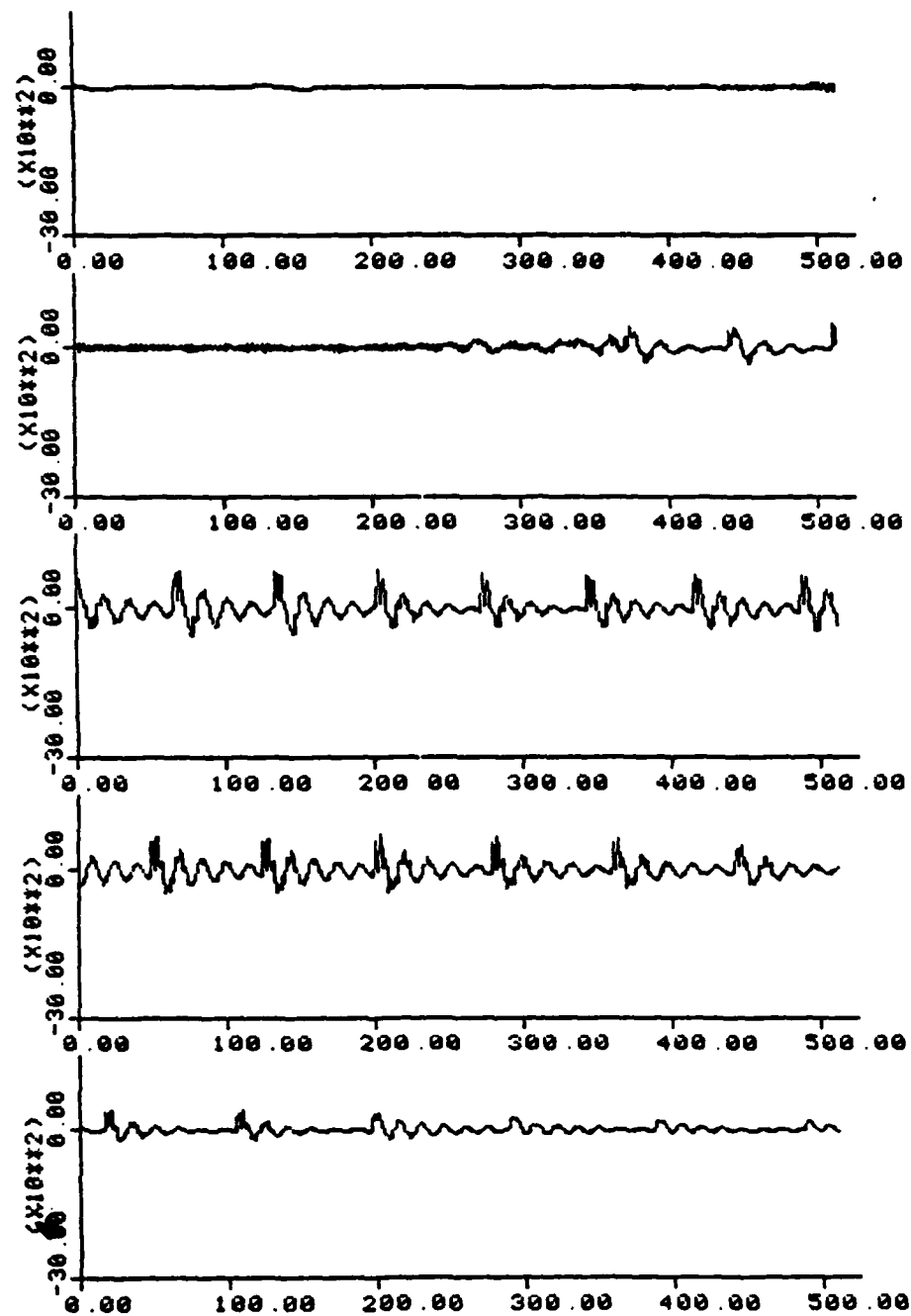


Figure IV-8.7 "Thieves who rob friends deserve jail"

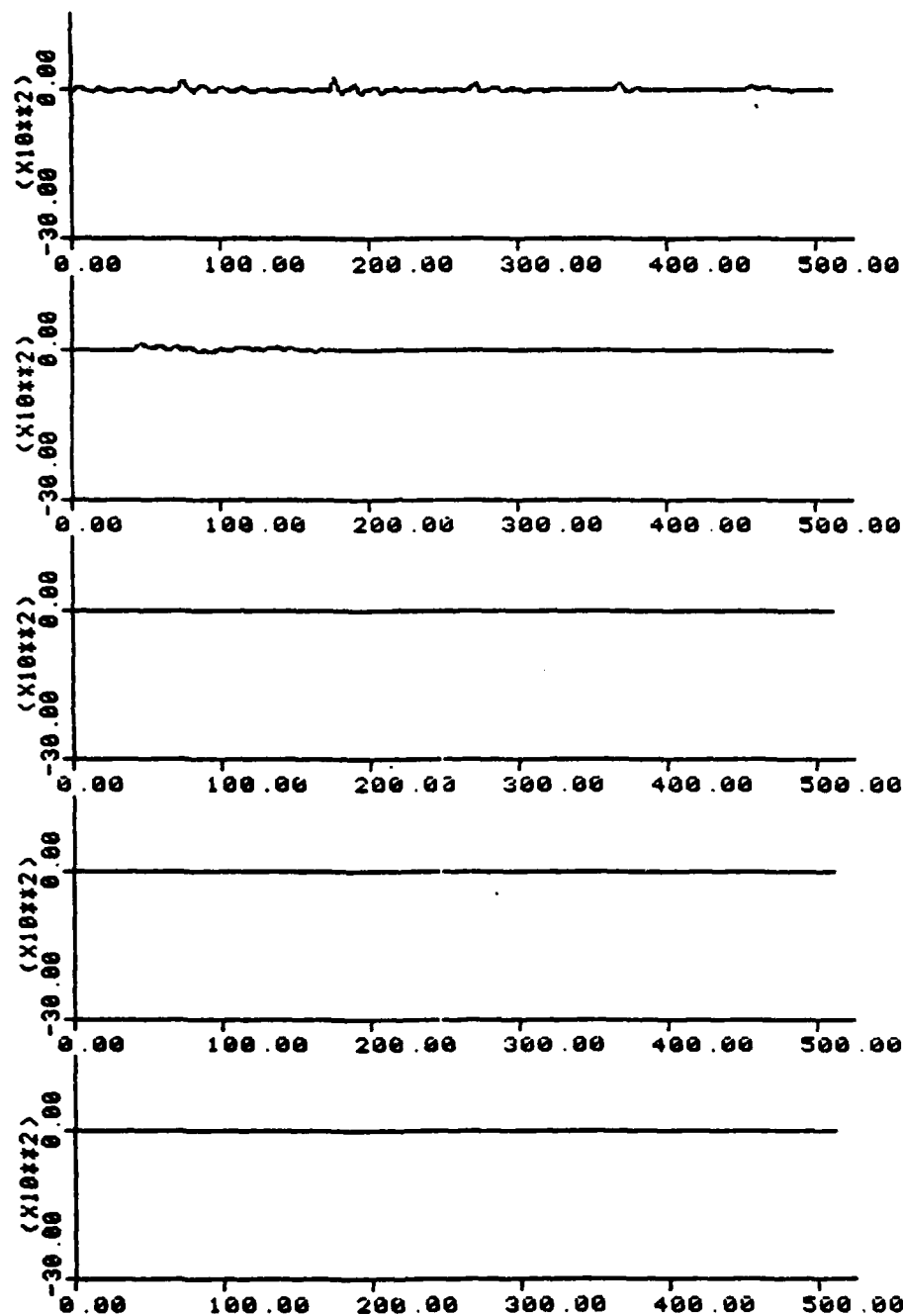


Figure IV-8.8 "Thieves who rob friends deserve jail"

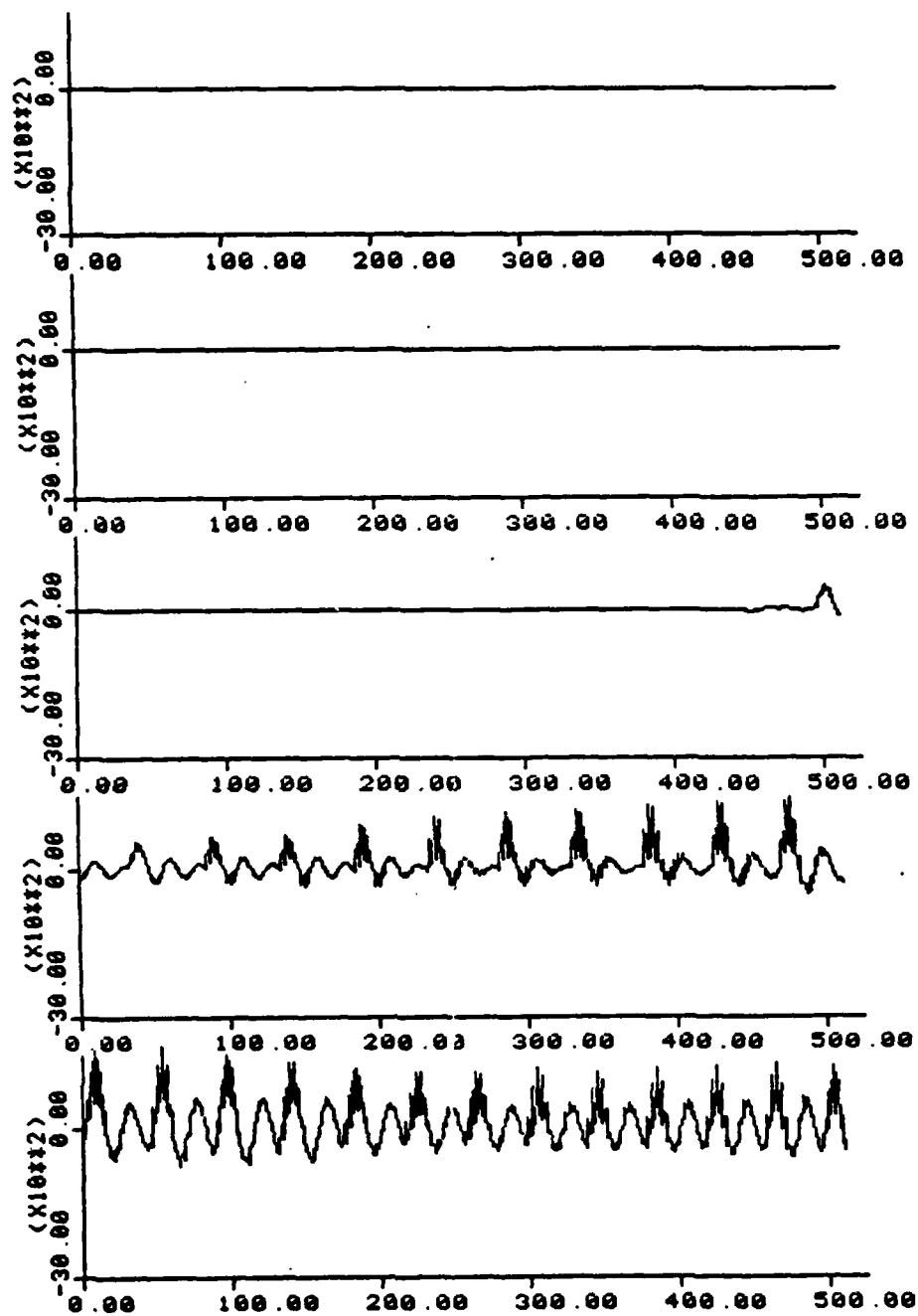


Figure IV-9.1 Synthesized Speech
"Thieves who rob friends deserve jail"

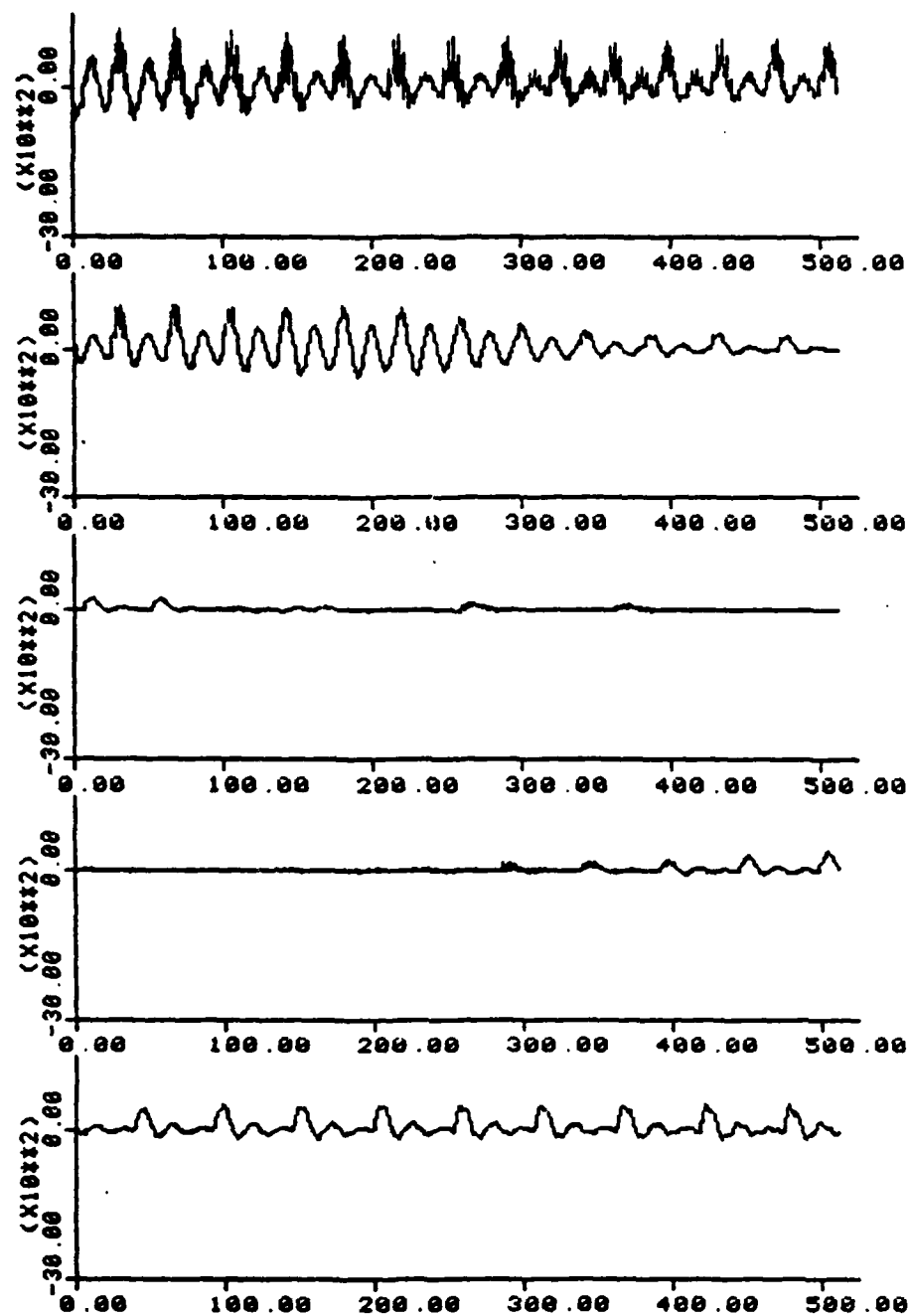


Figure IV-9.2 "Thieves who rob friends deserve jail"

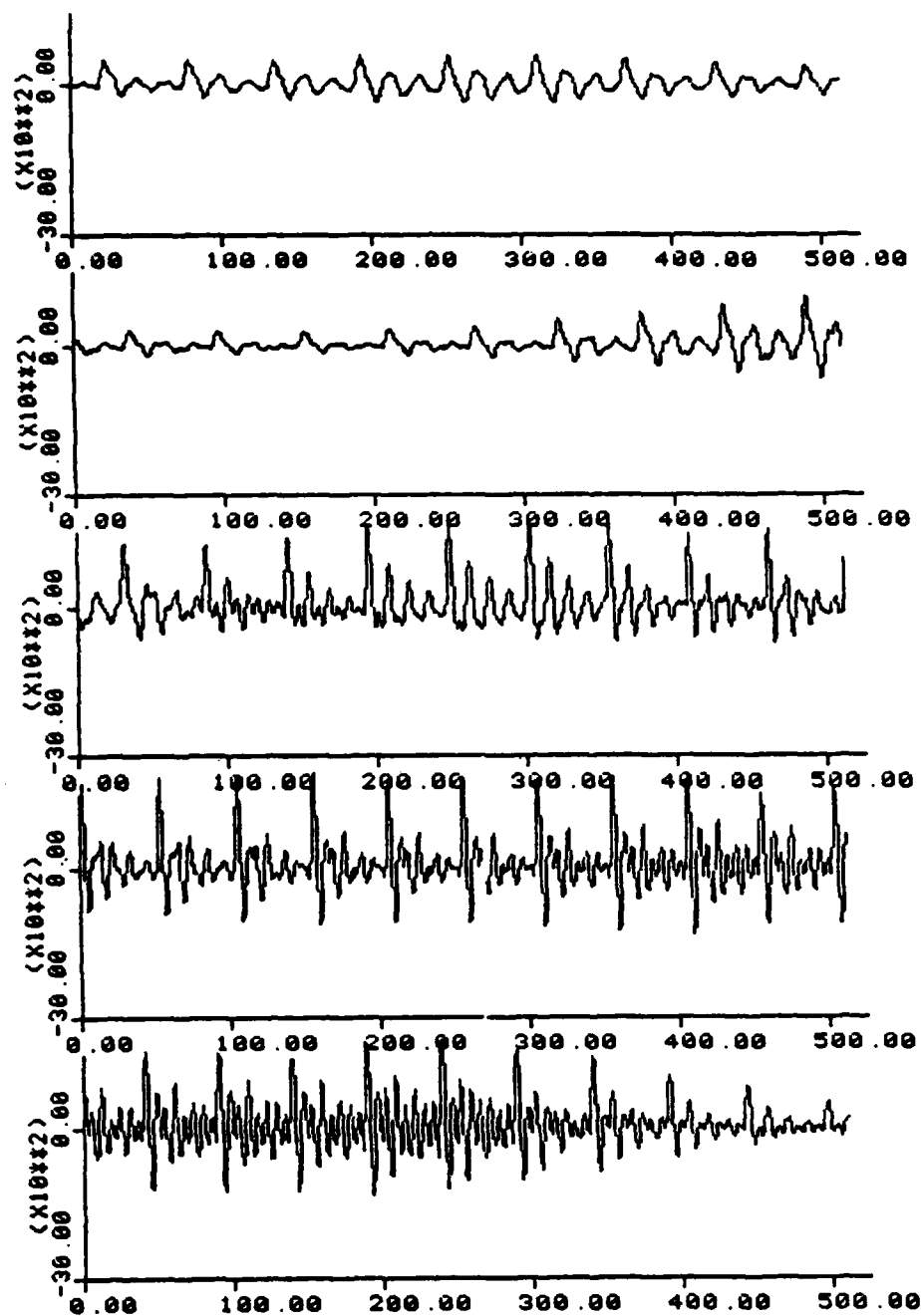


Figure IV-9.3 "Thieves who rob friends deserve jail"

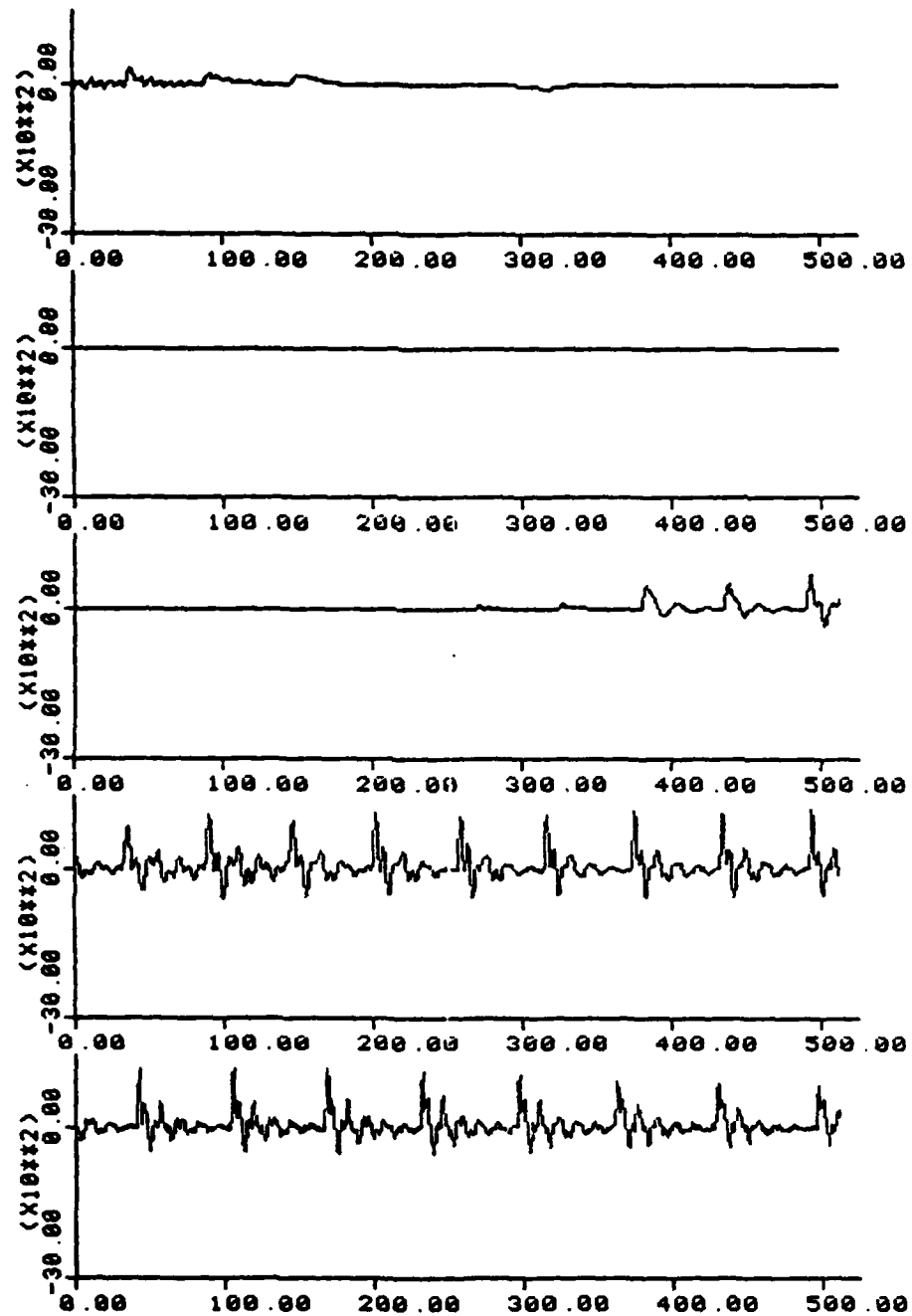


Figure IV-9.4 "Thieves who rob friends deserve jail"

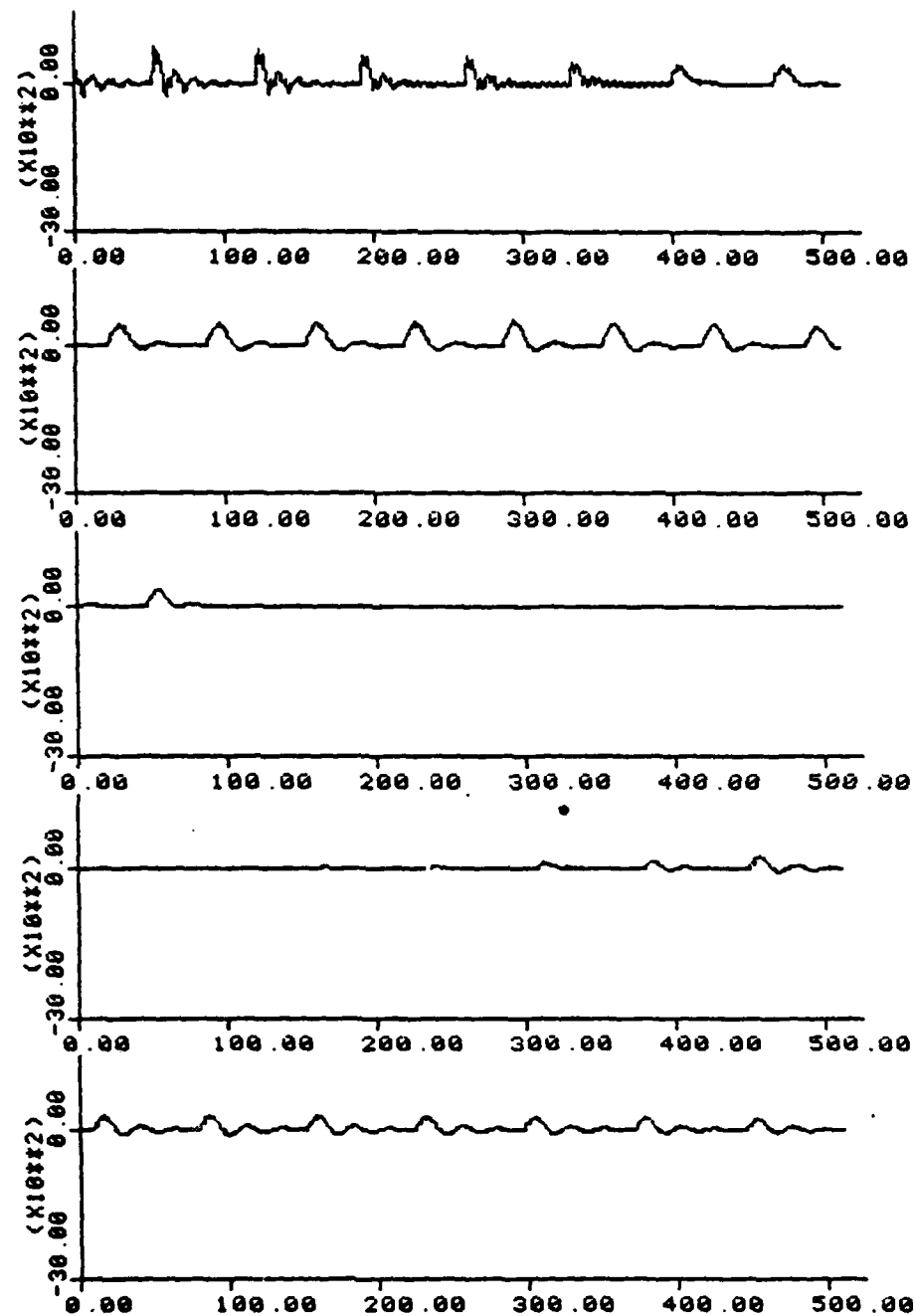


Figure IV-9.5 "Thieves who rob friends deserve jail"

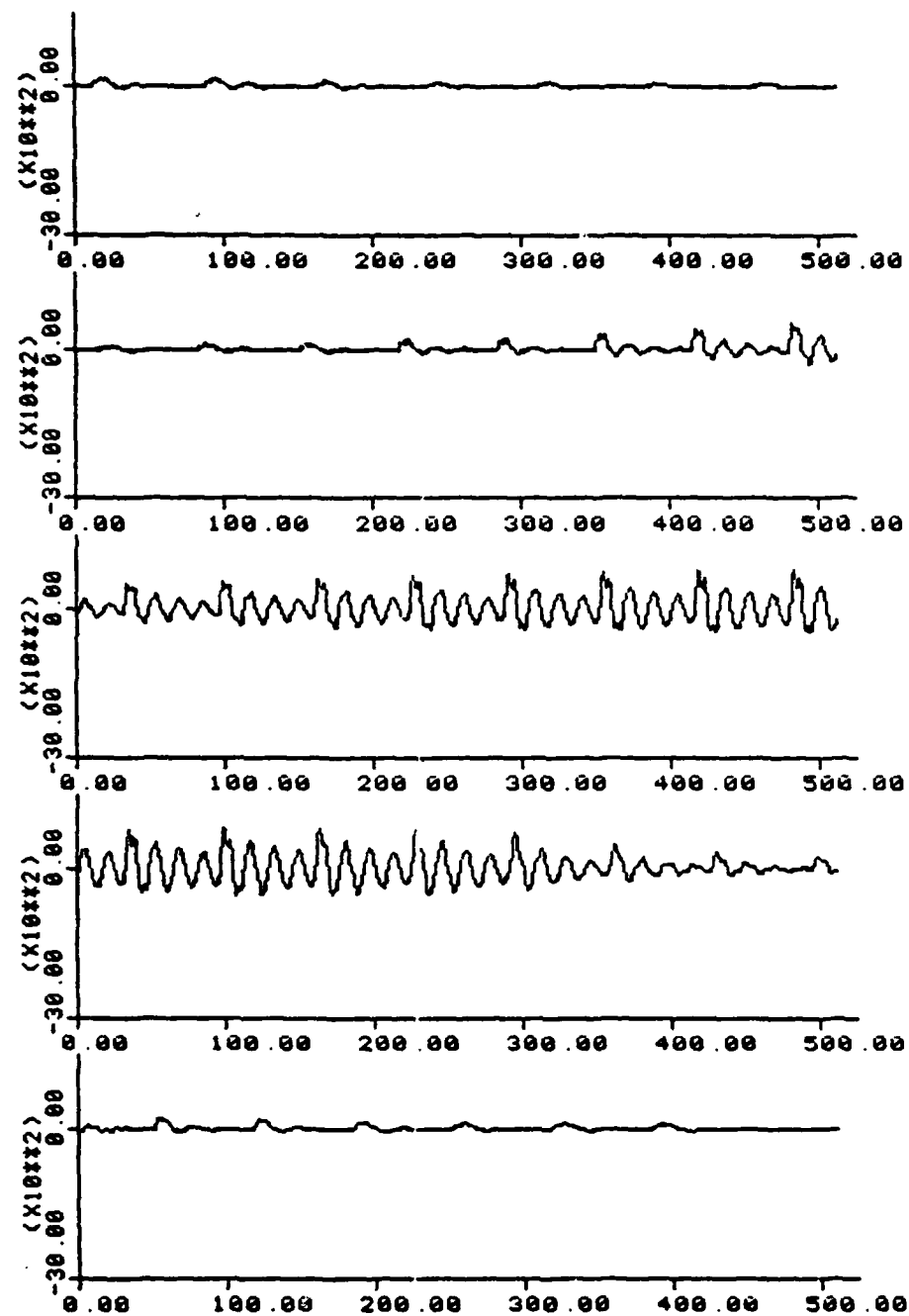


Figure IV-9.6 "Thieves who rob friends deserve jail"

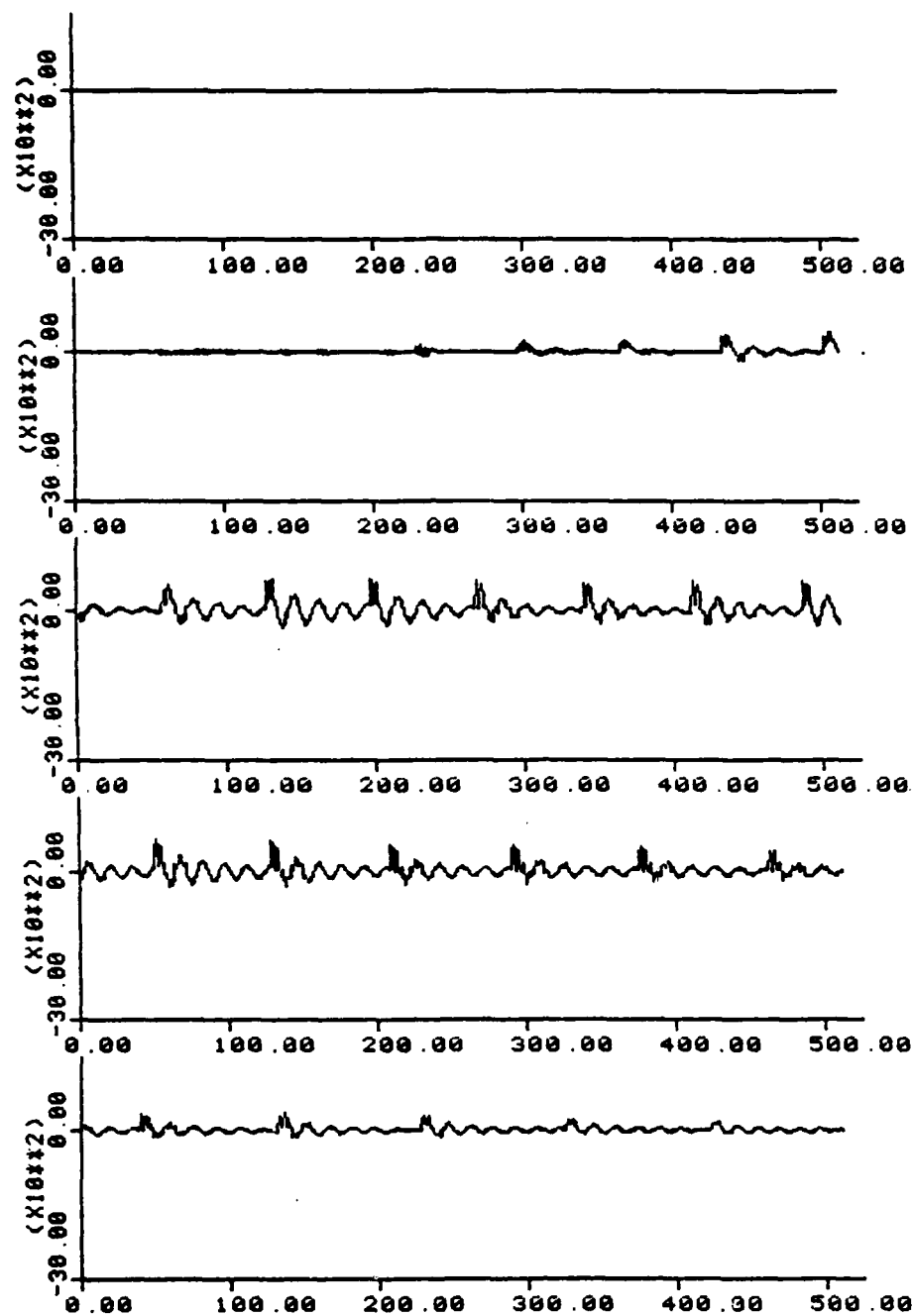


Figure IV-9.7 "Thieves who rob friends deserve jail"

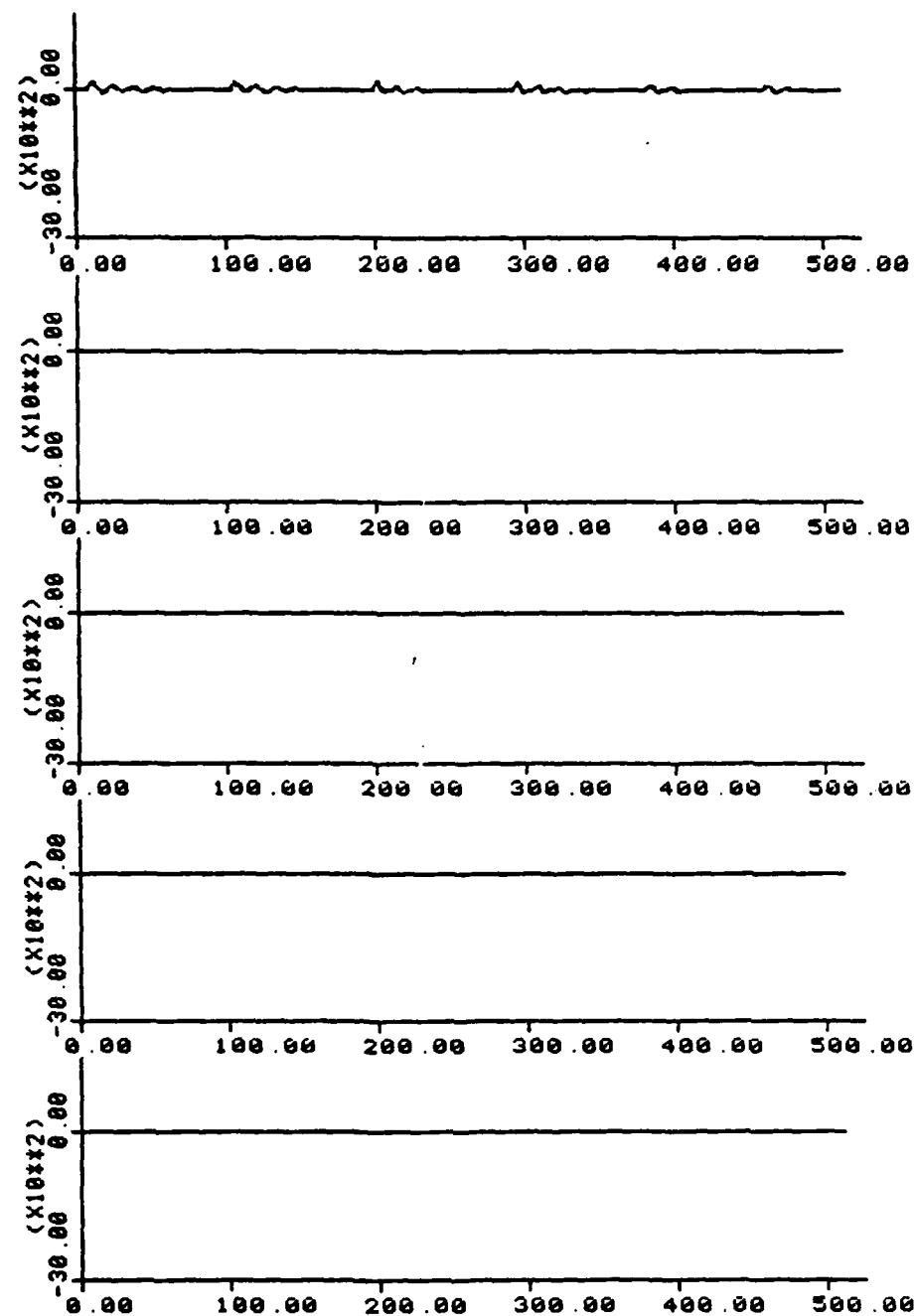
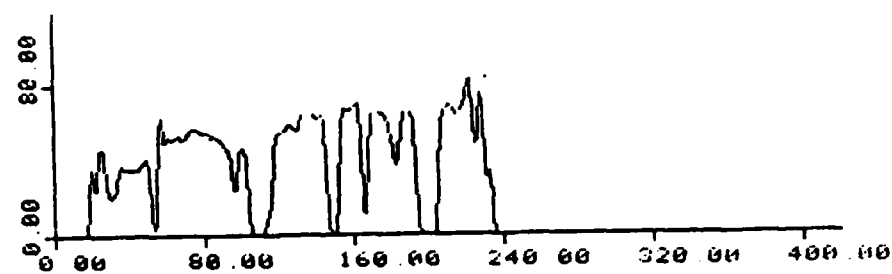
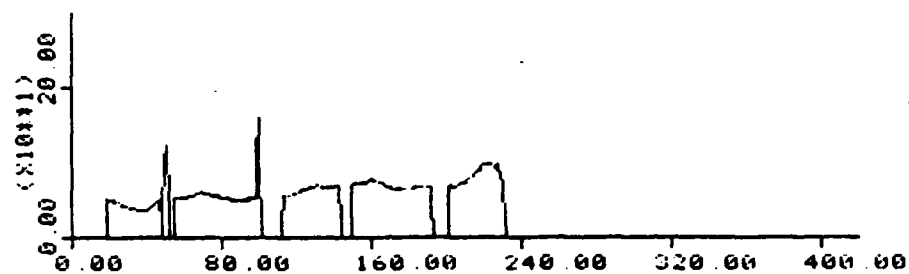


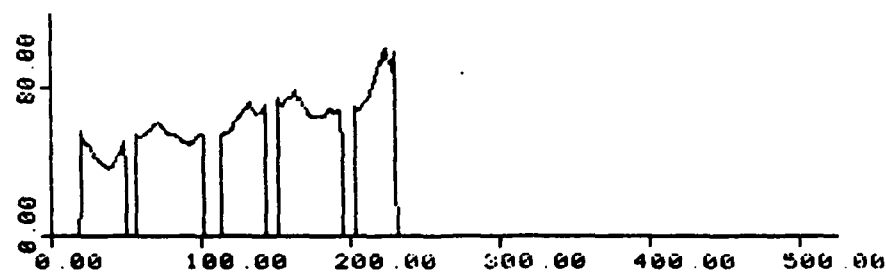
Figure IV-9.8 "Thieves who rob friends deserve jail"



(a) AUTOC



(b) SIFT



(c) Handpainted

Figure IV-10 Pitch Period Plots of S2

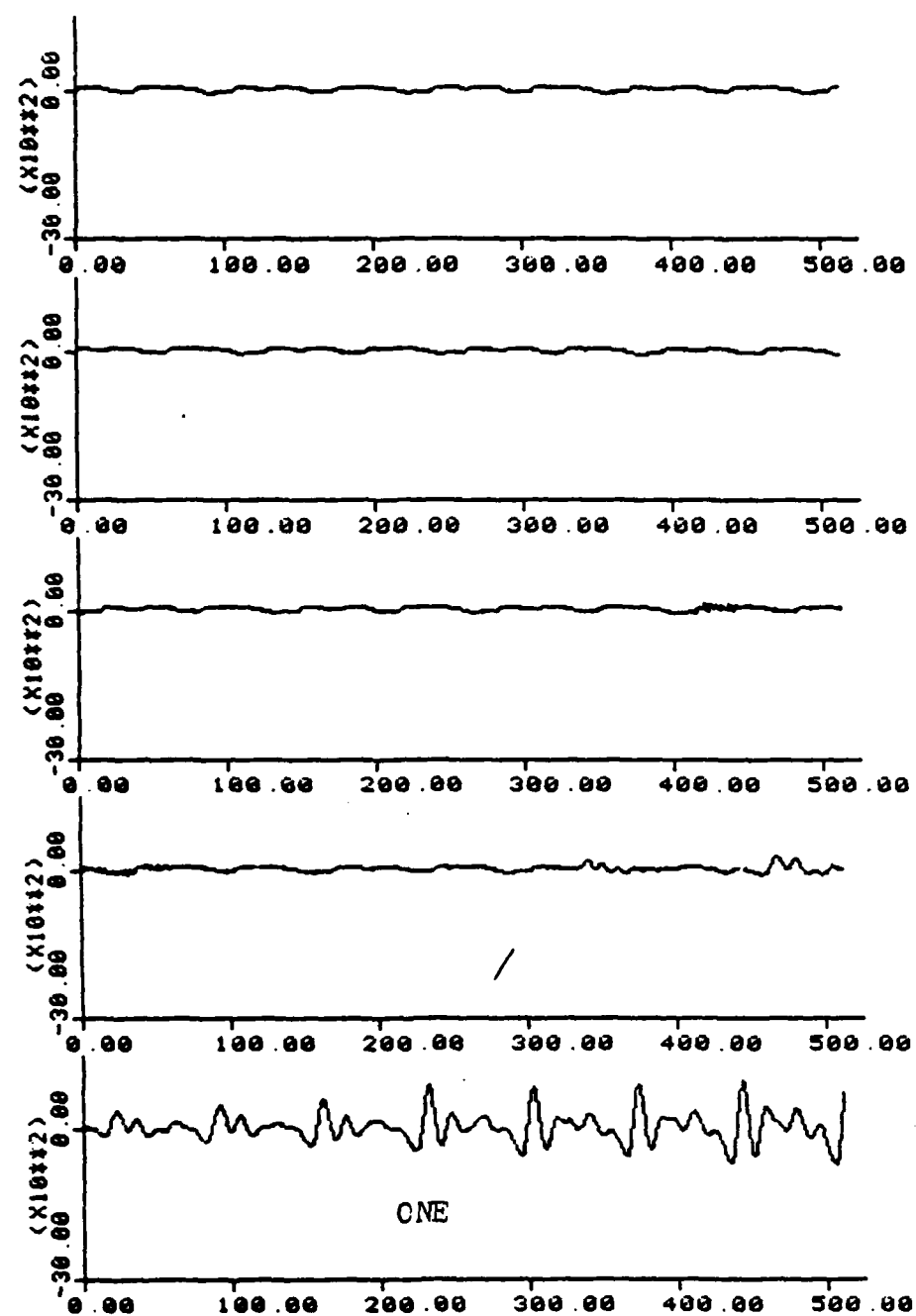


Figure IV-11.1 "ONE...TWO...THREE...FOUR"

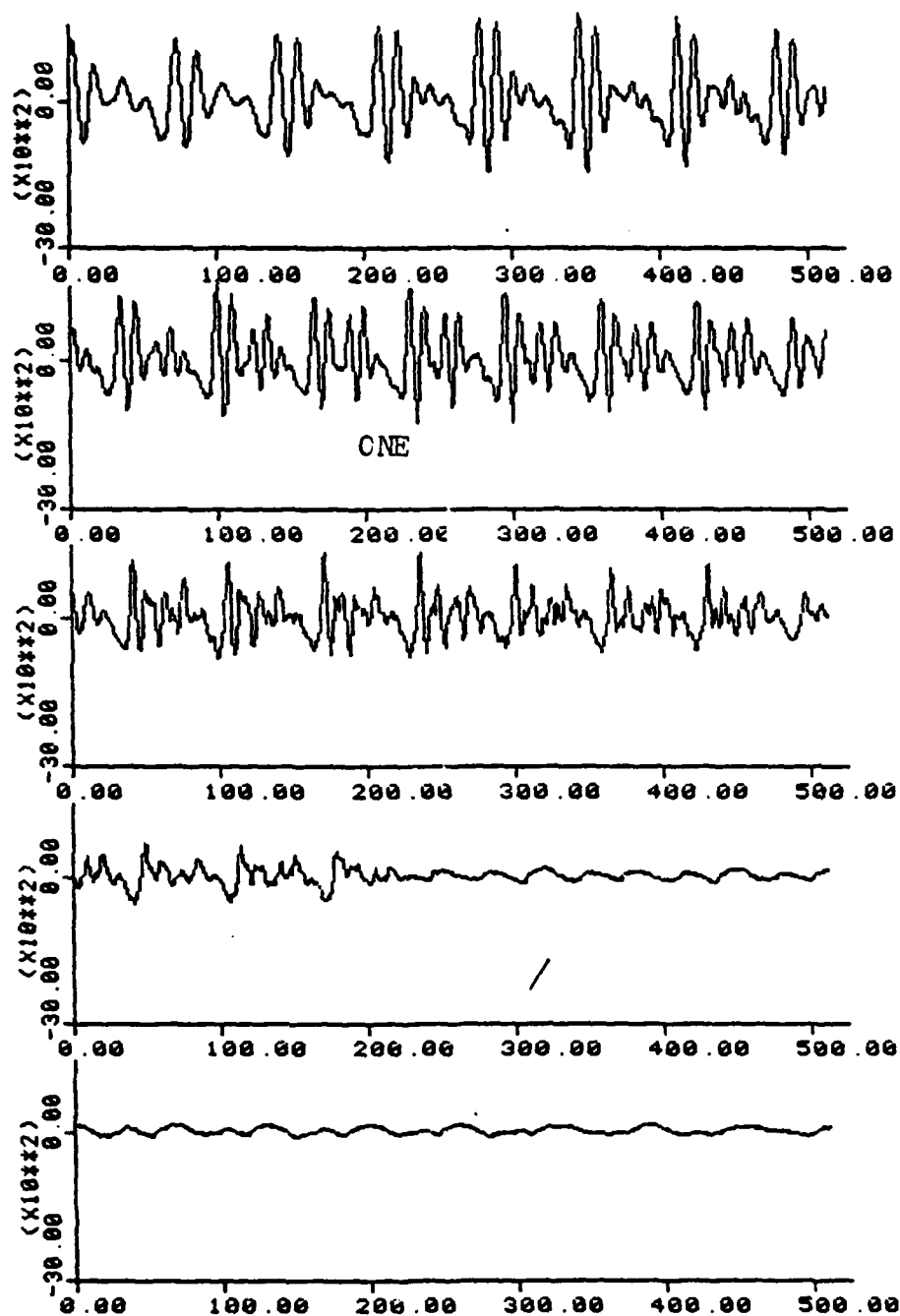


Figure IV-11.2 "CNE...TWO...THREE...FOUR"

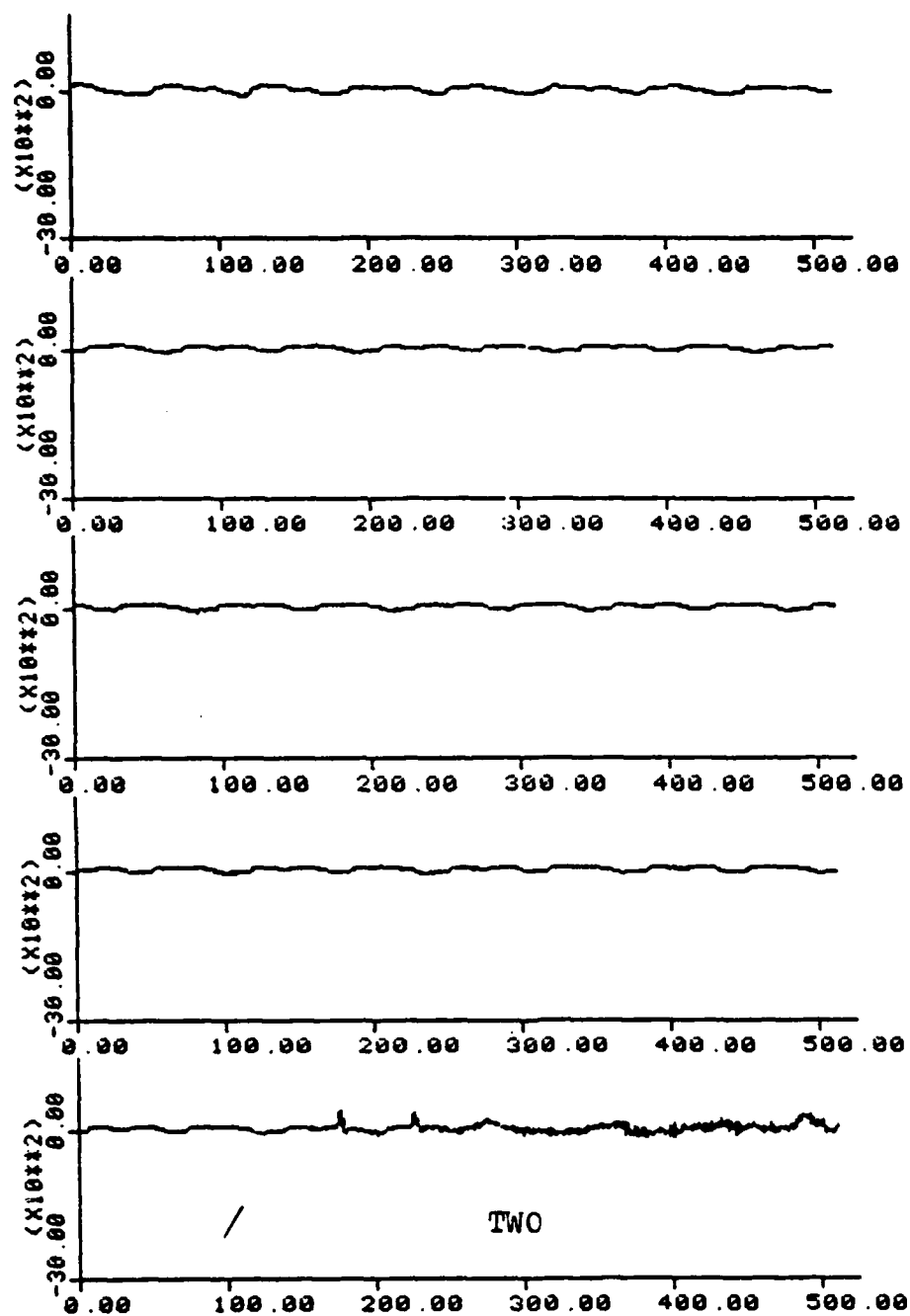


Figure IV-11.3 "ONE...TWO...THREE...FOUR"

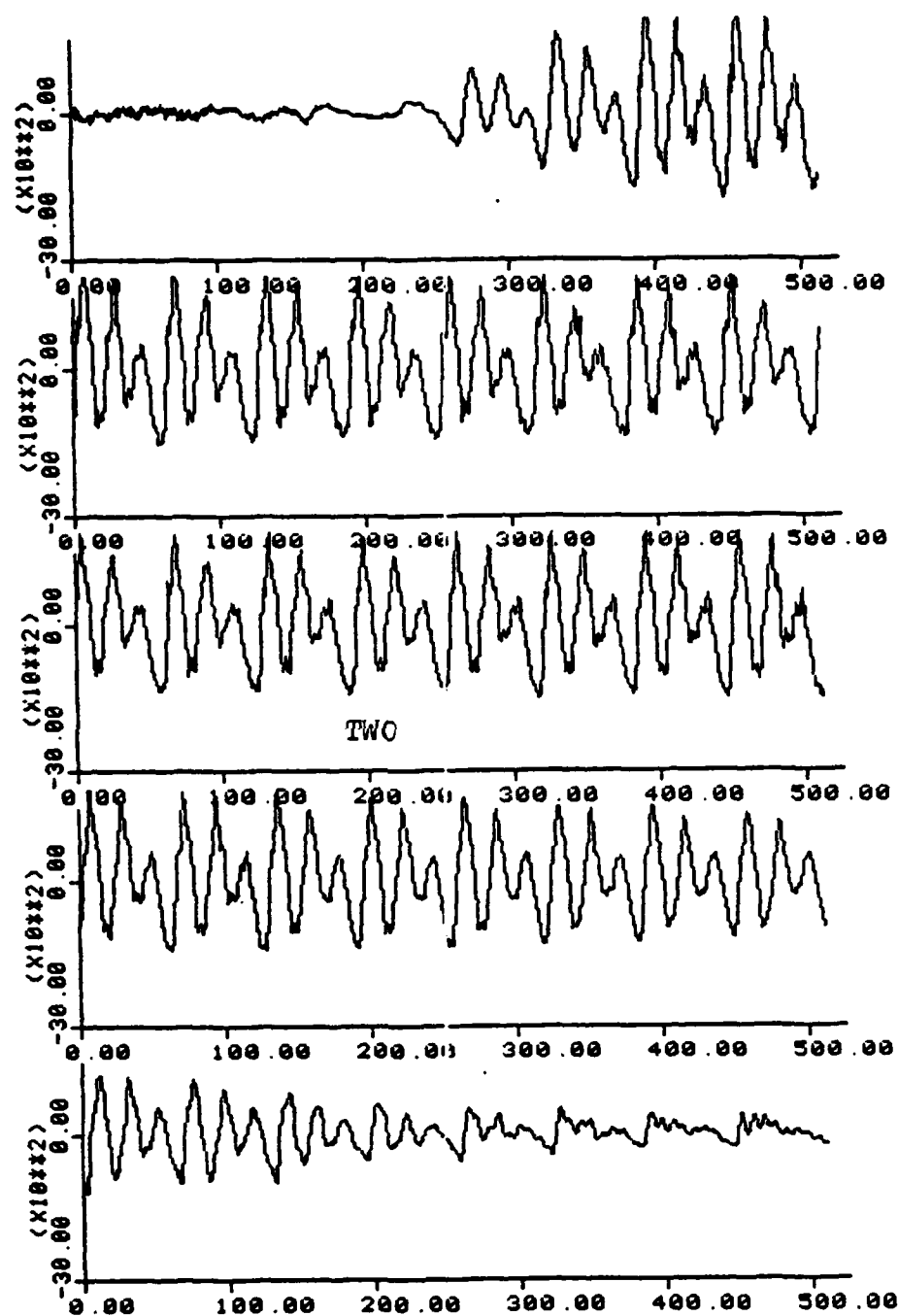


Figure IV-11.4 "ONE...TWO...THREE...FCUR"

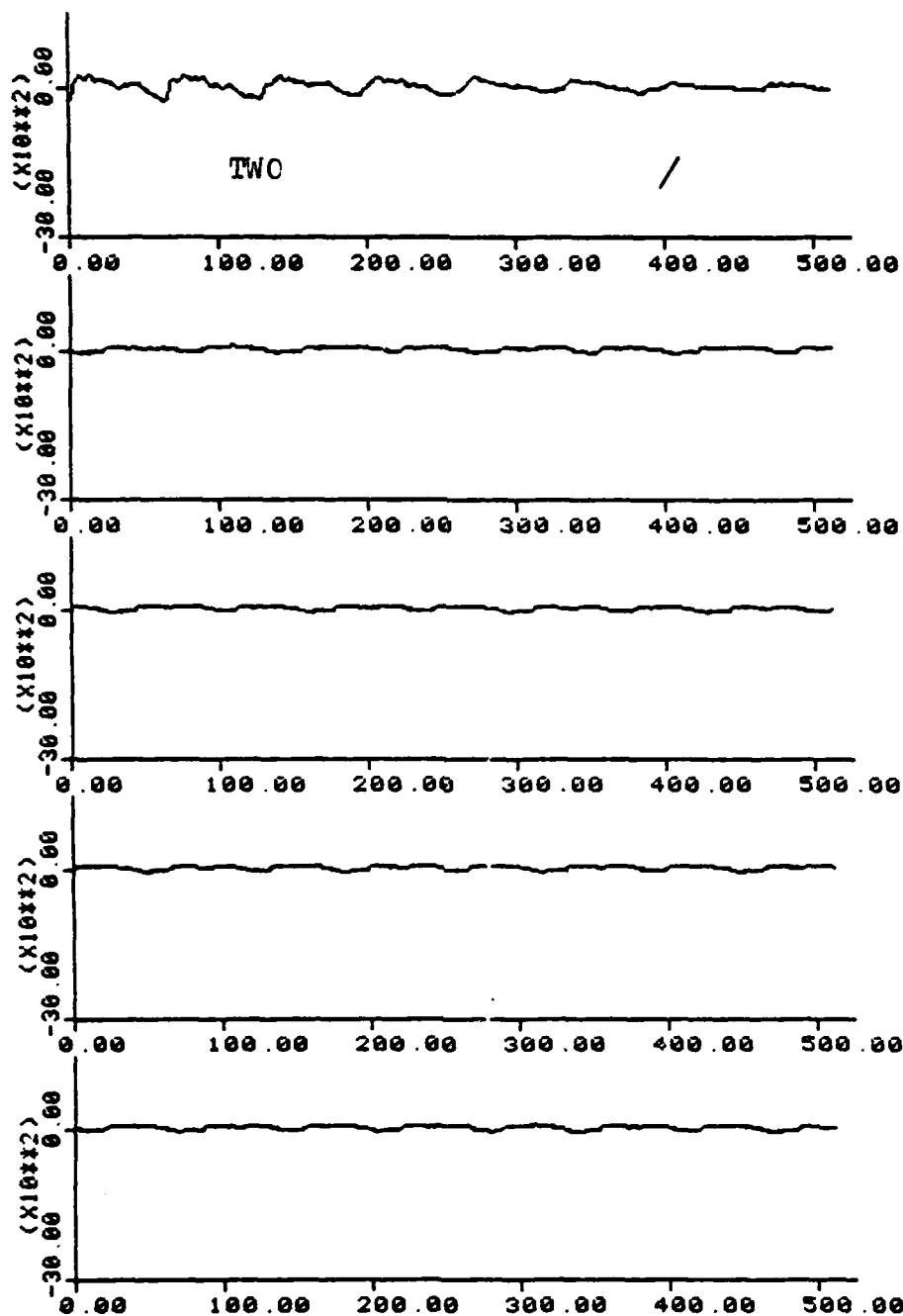


Figure IV-11.5 "ONE...TWO...THREE...FOUR"

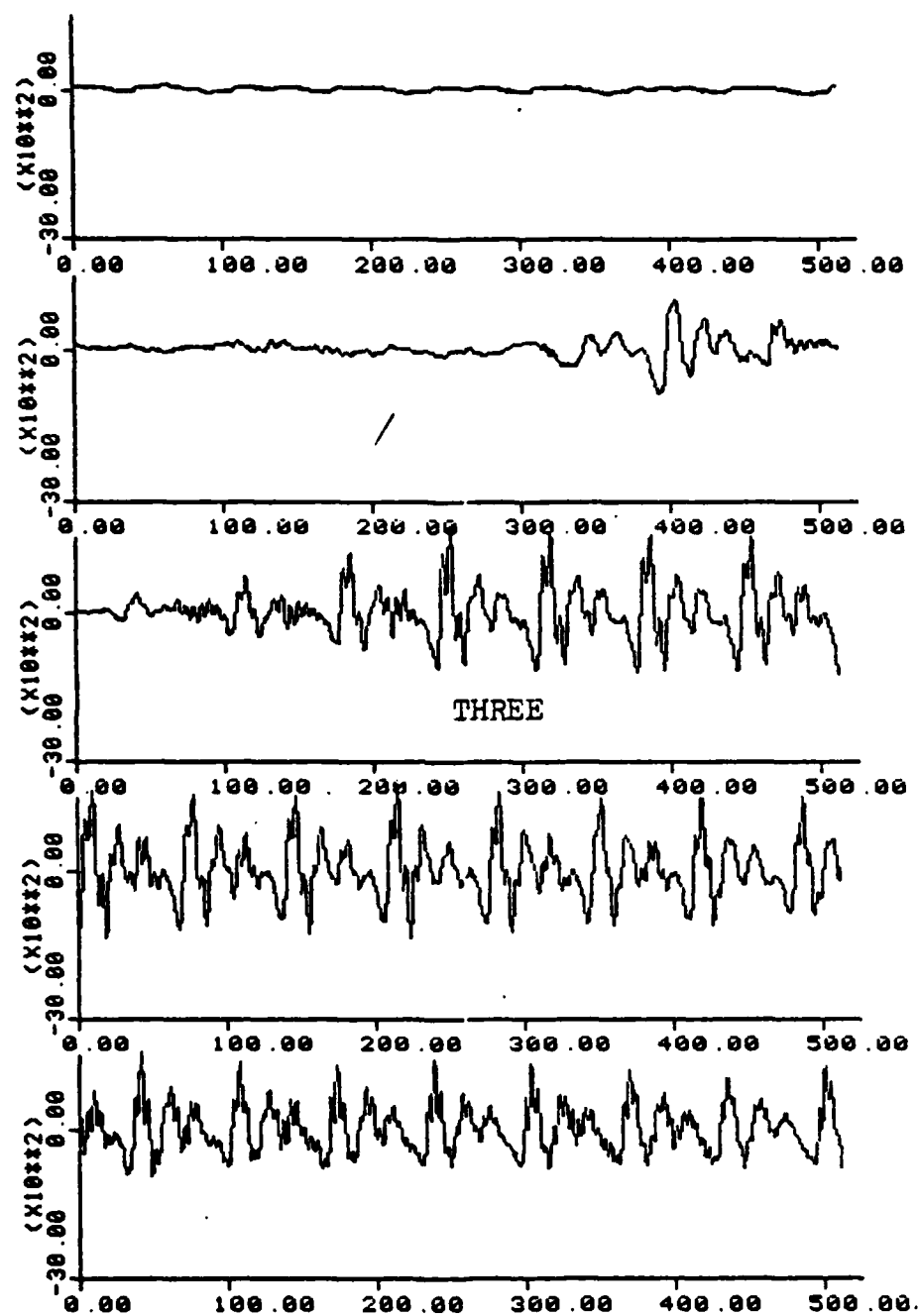


Figure IV-11.6 "ONE...TWO...THREE...FOUR"

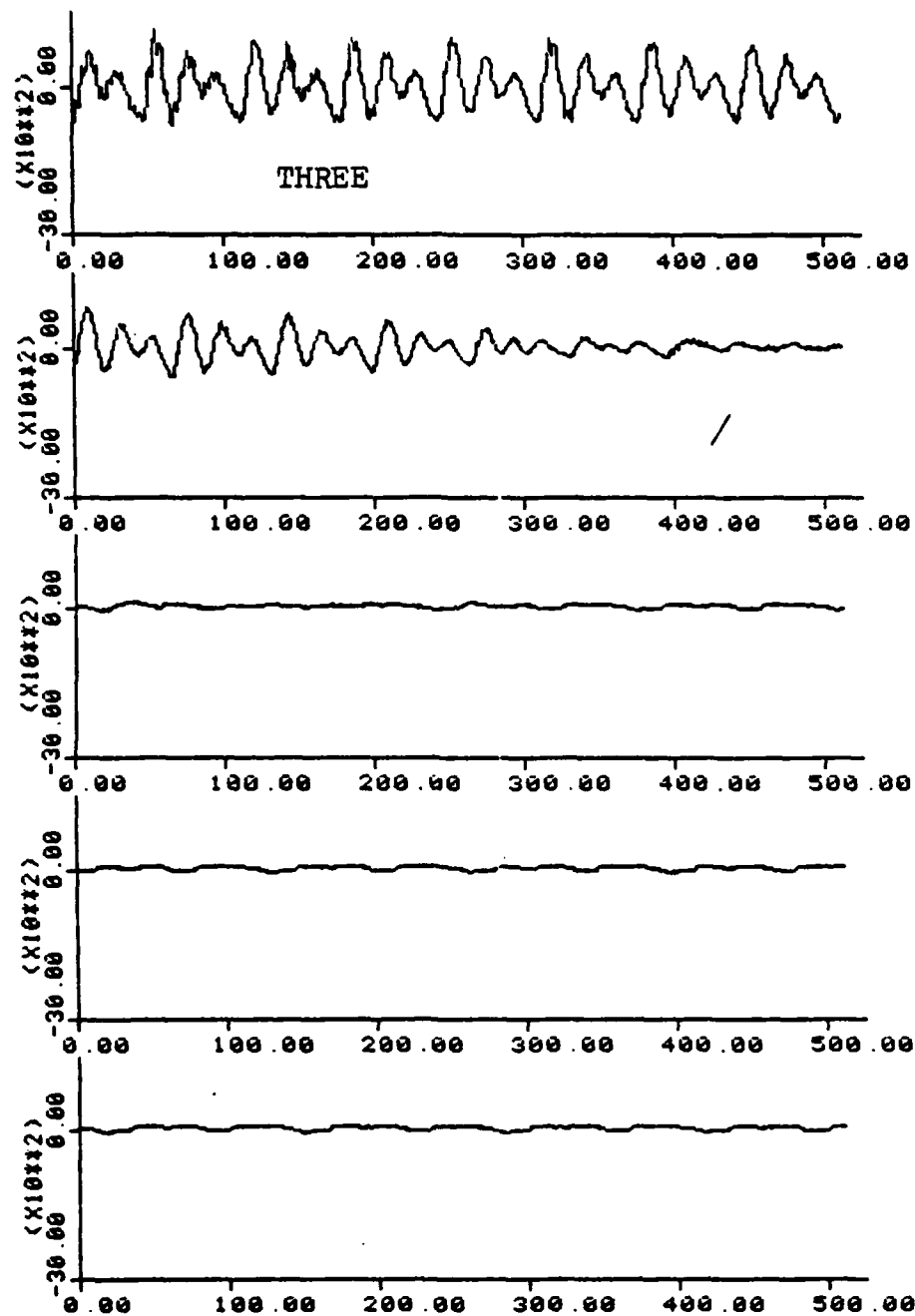


Figure IV-11.7 "CNE...TWO...THREE...FOUR"

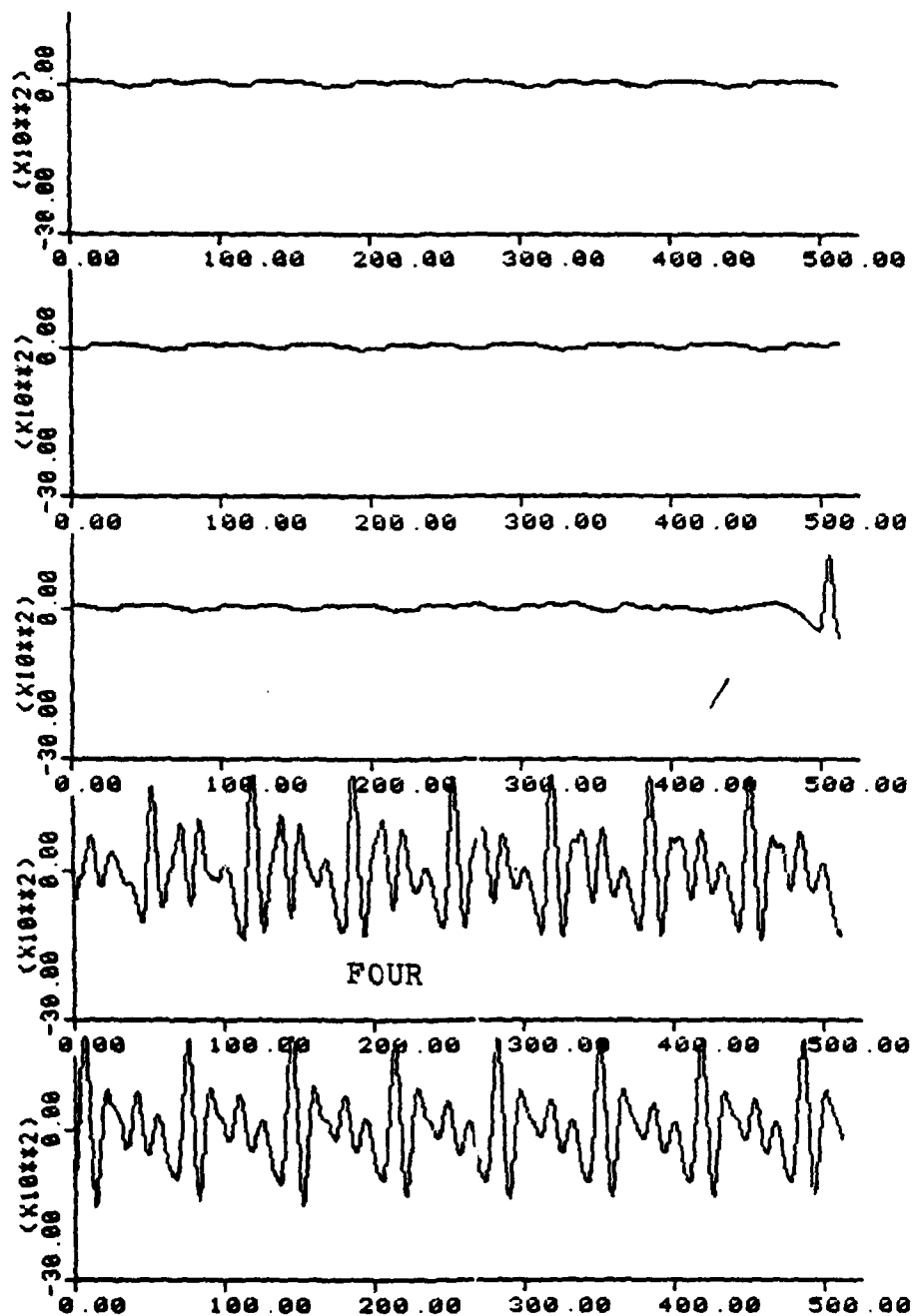


Figure IV-11.8 "ONE...TWO...THREE...FOUR"

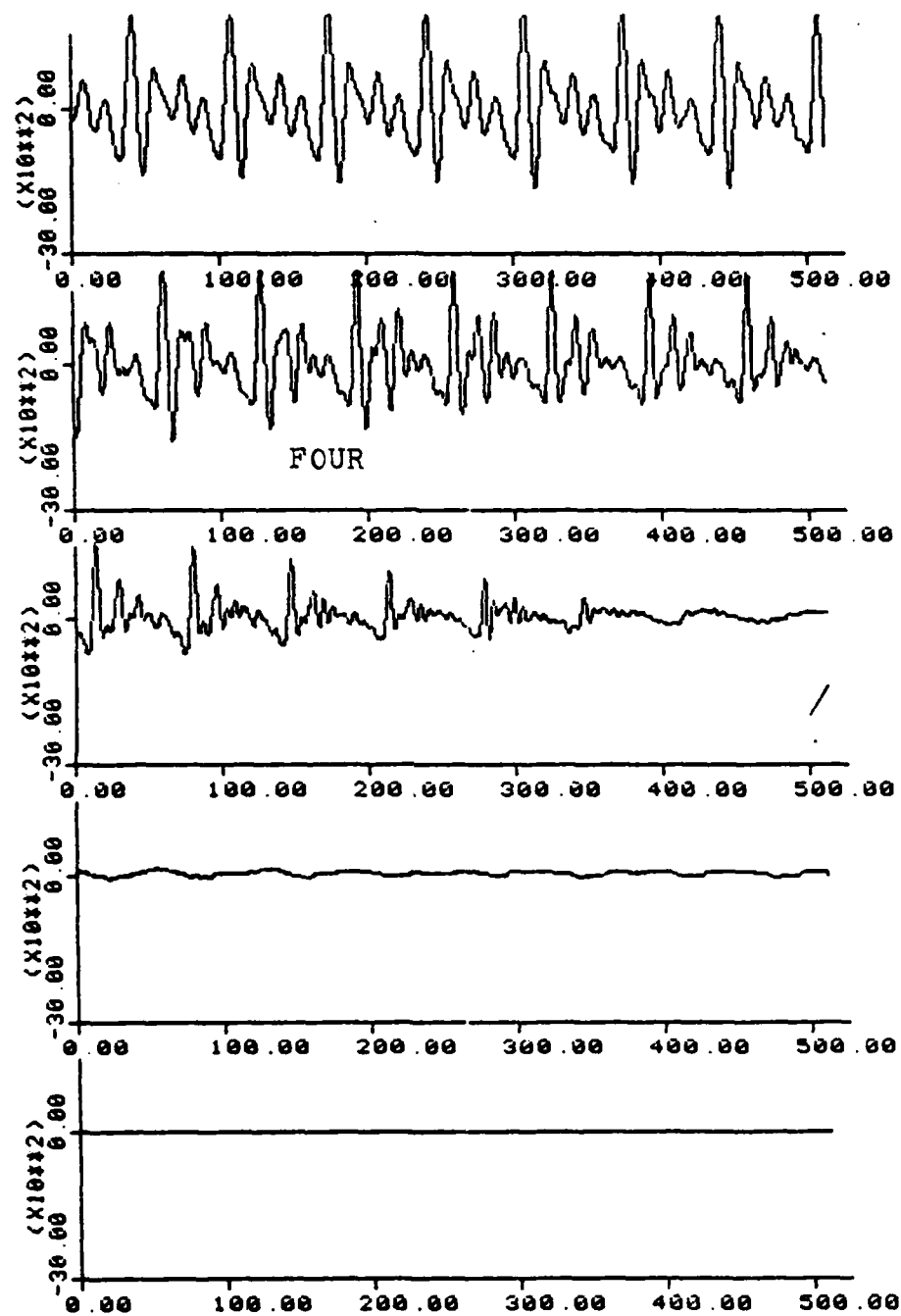


Figure IV-11.9 "CNE...TWO...THREE...FOUR"

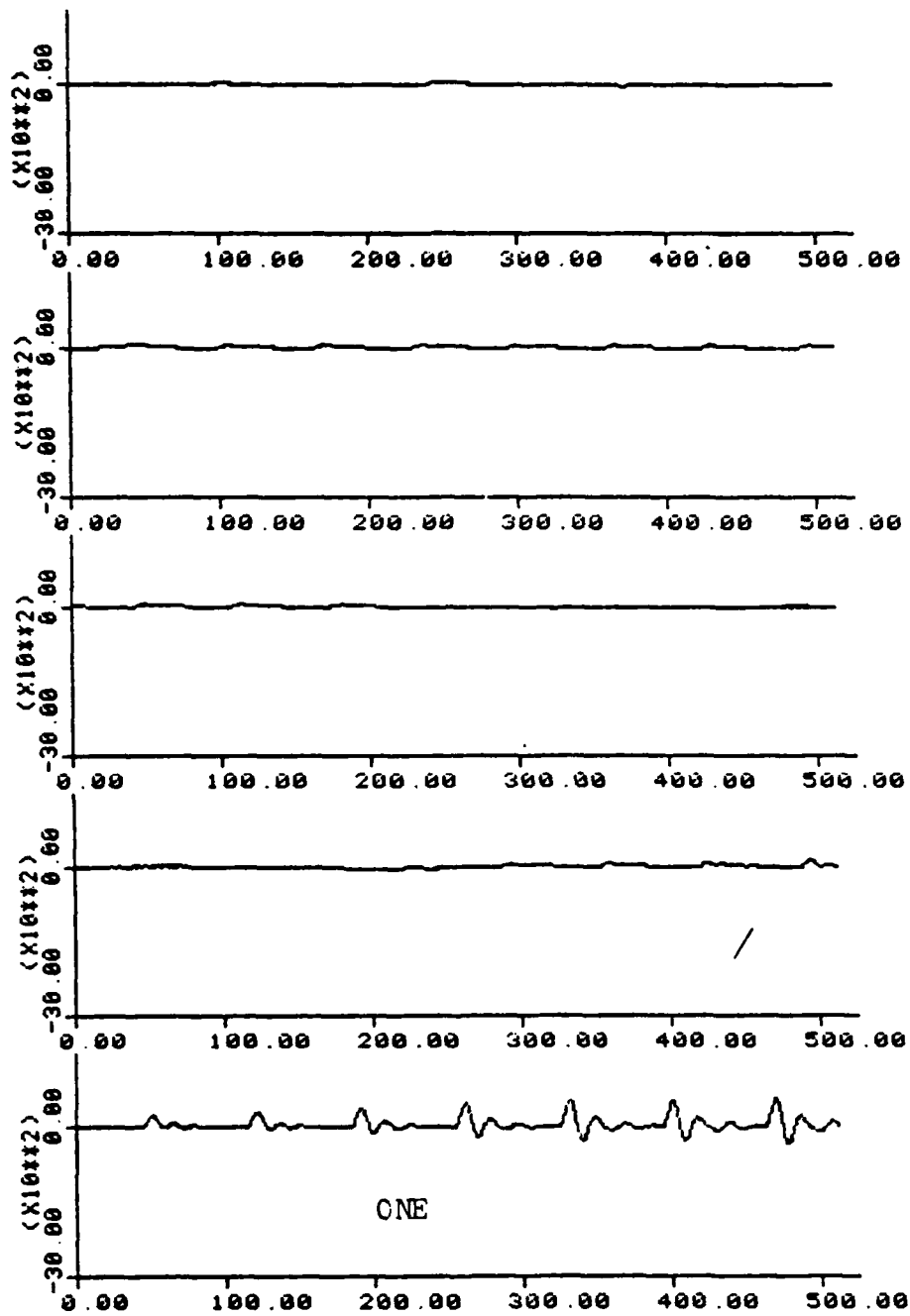


Figure IV-12.1 "CNE...TWC...THREE...FOUR"

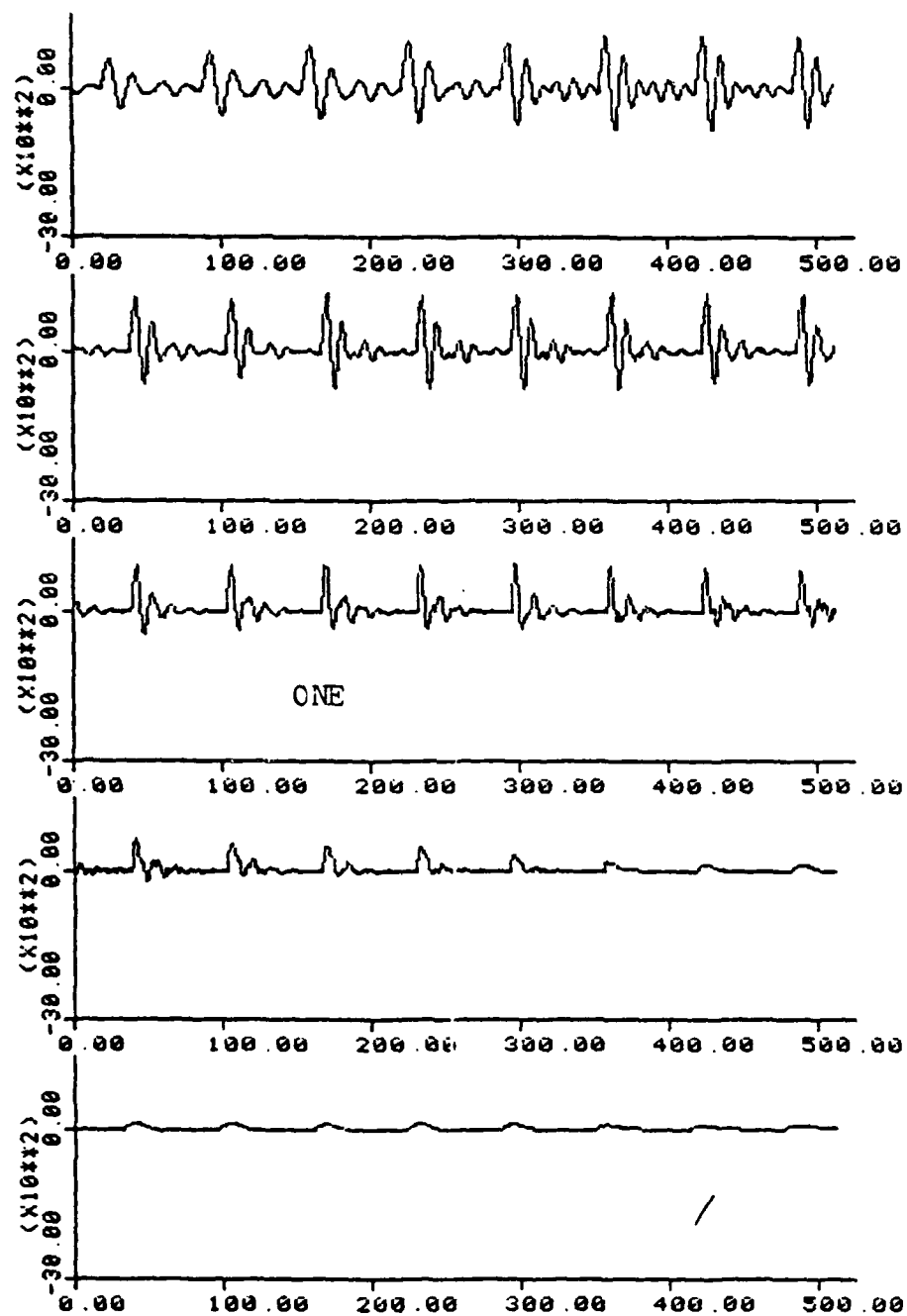


Figure IV-12.2 "ONE...TWC...THREE...FOUR"

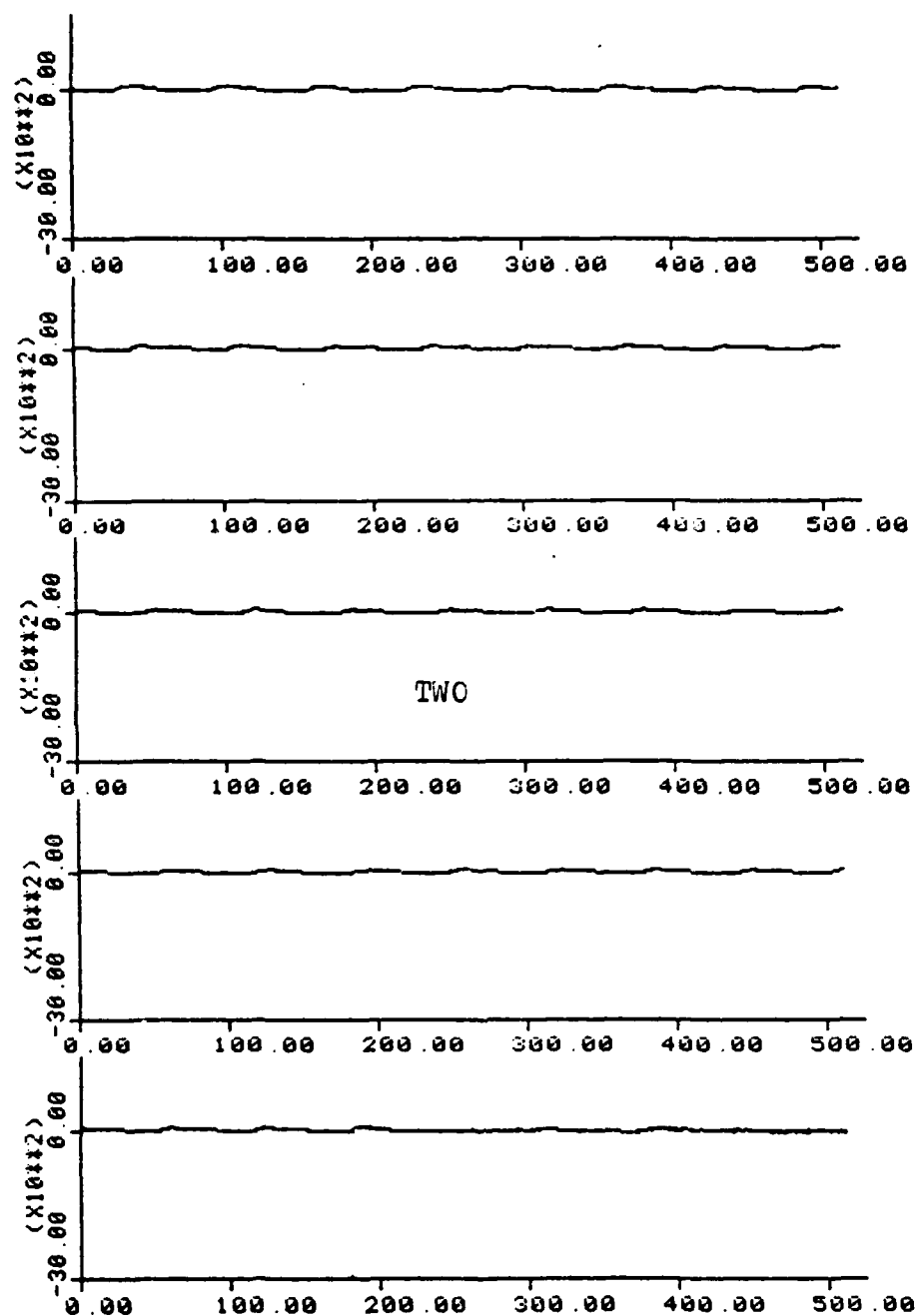


Figure IV-12.3 "ONE...TWO...THREE...FCUR"

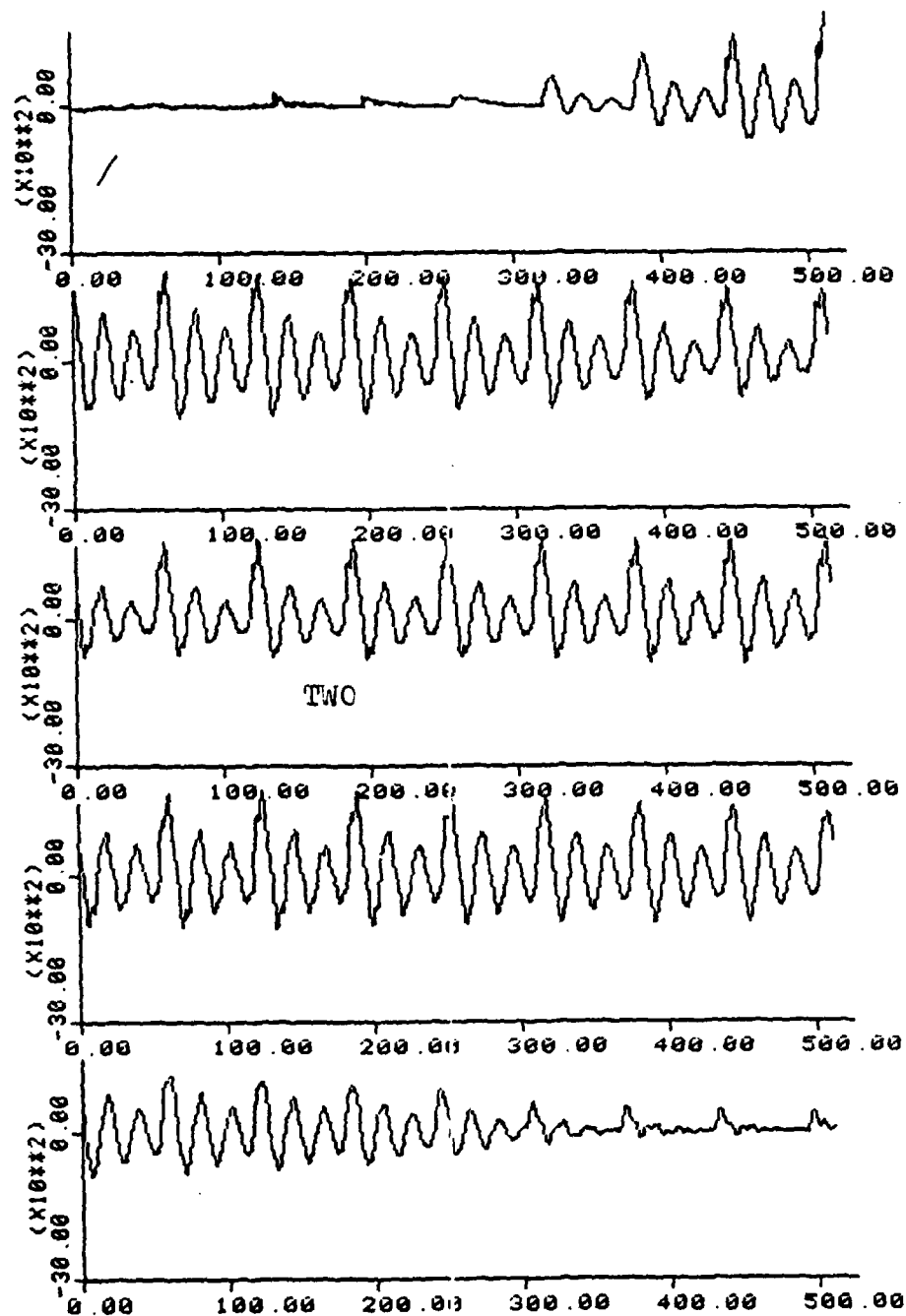


Figure IV-12.4 "ONE...TWO...THREE...FOUR"

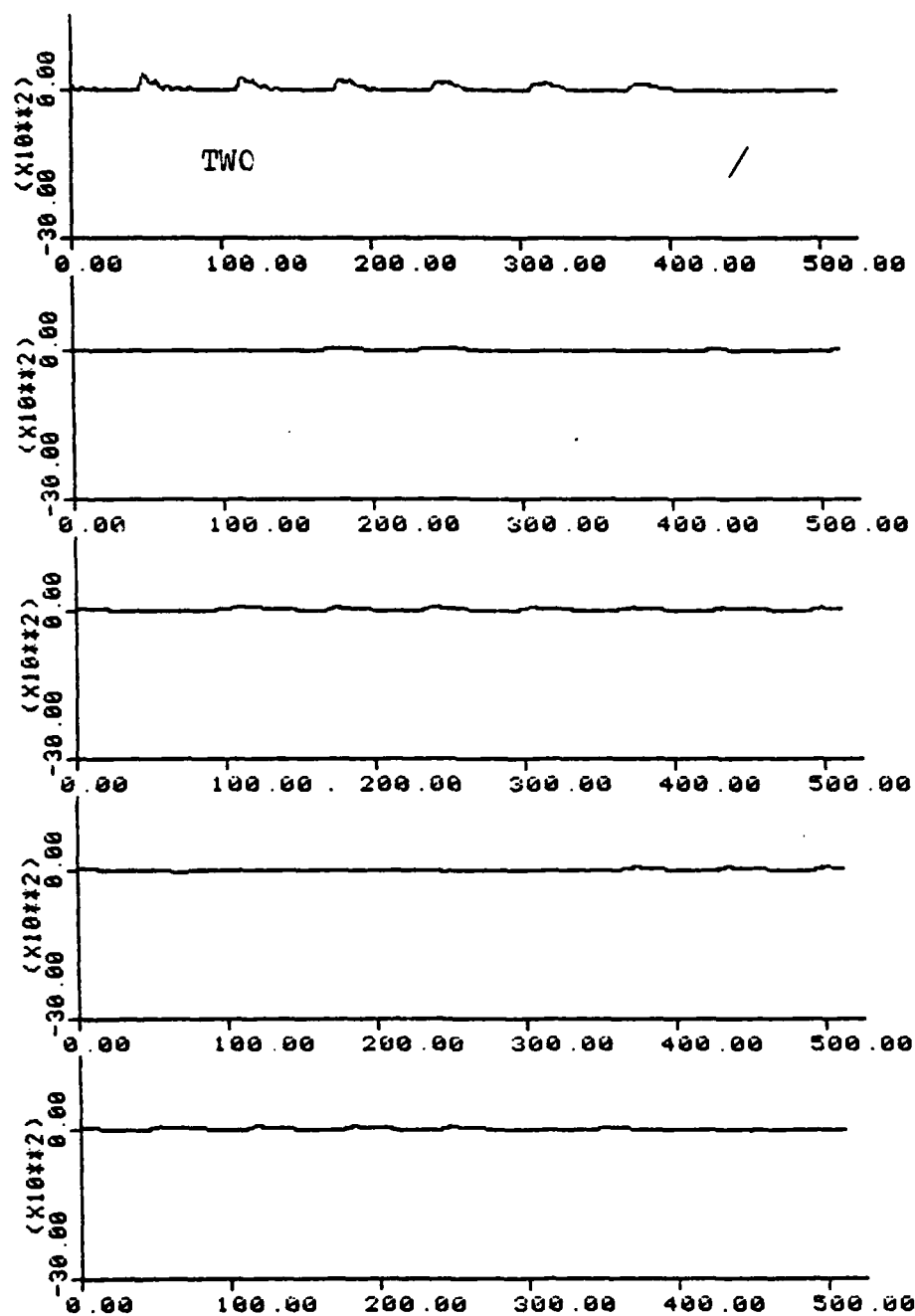


Figure IV-12.5 "ONE...TWO...THREE...FCUR"

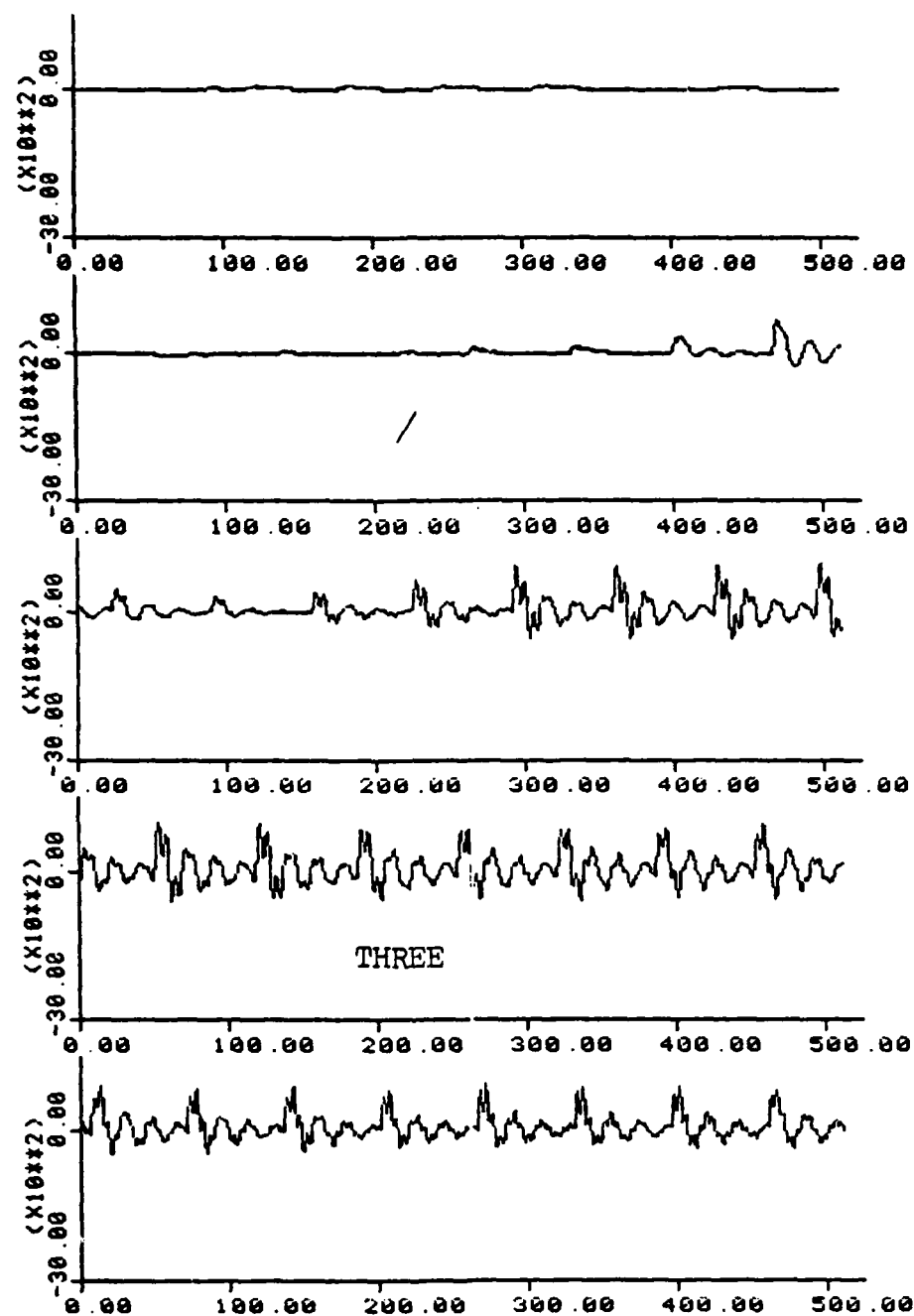


Figure IV-12.6 "CNE...TWO...THREE...FCUR"

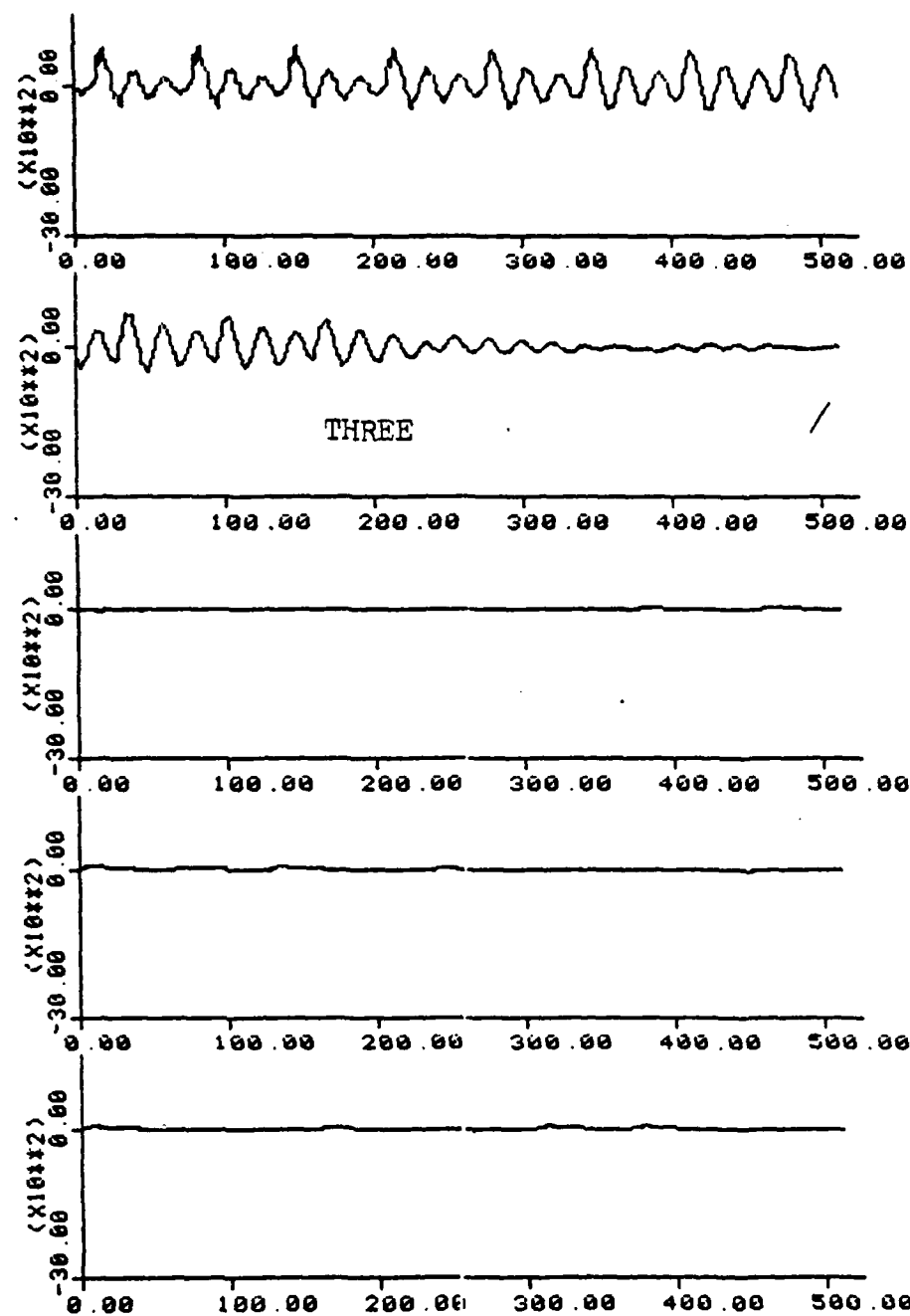


Figure IV-12.7 "ONE...TWO...THREE...FOUR"

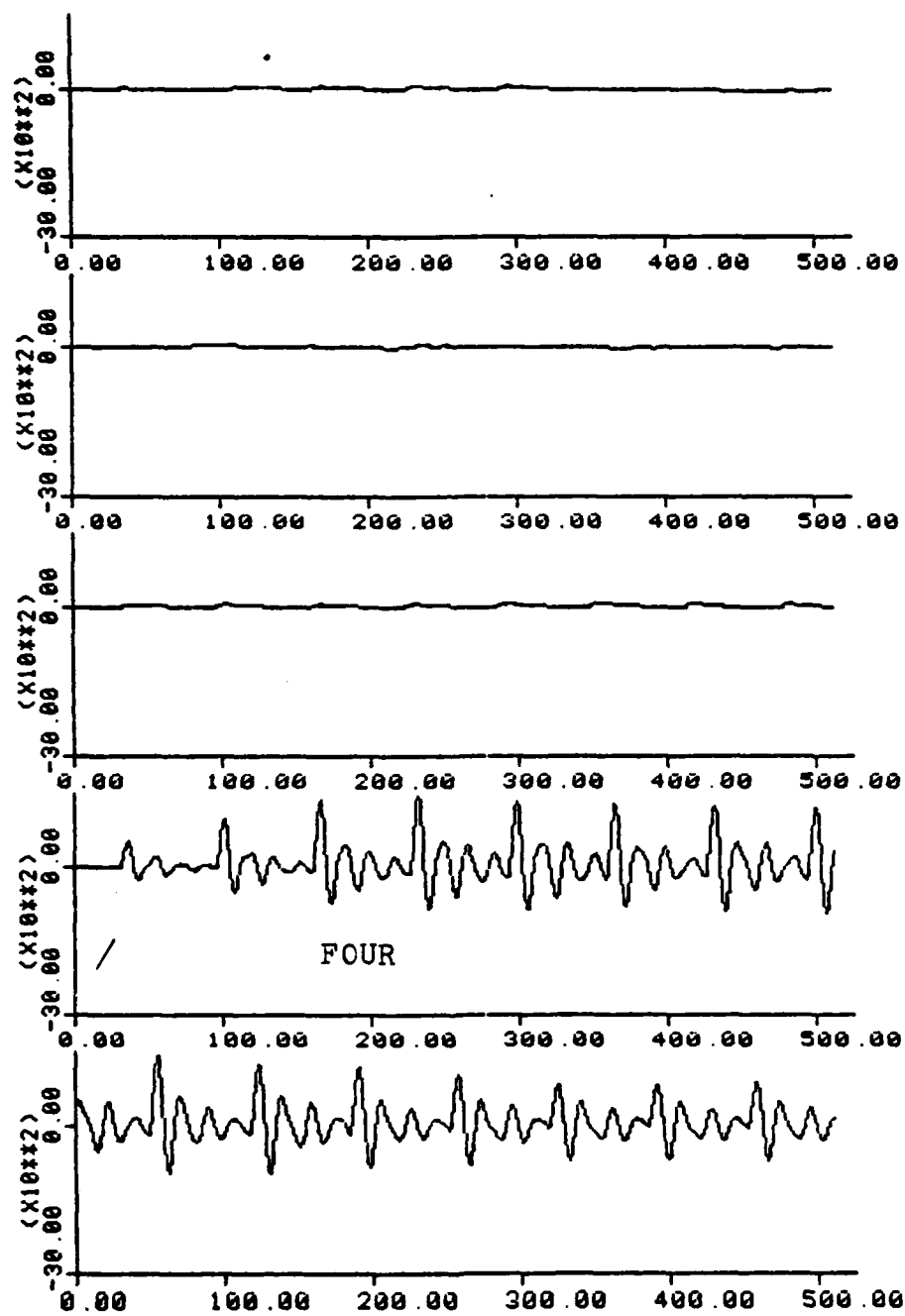


Figure IV-12.8 "ONE...TWO...THREE...FOUR"

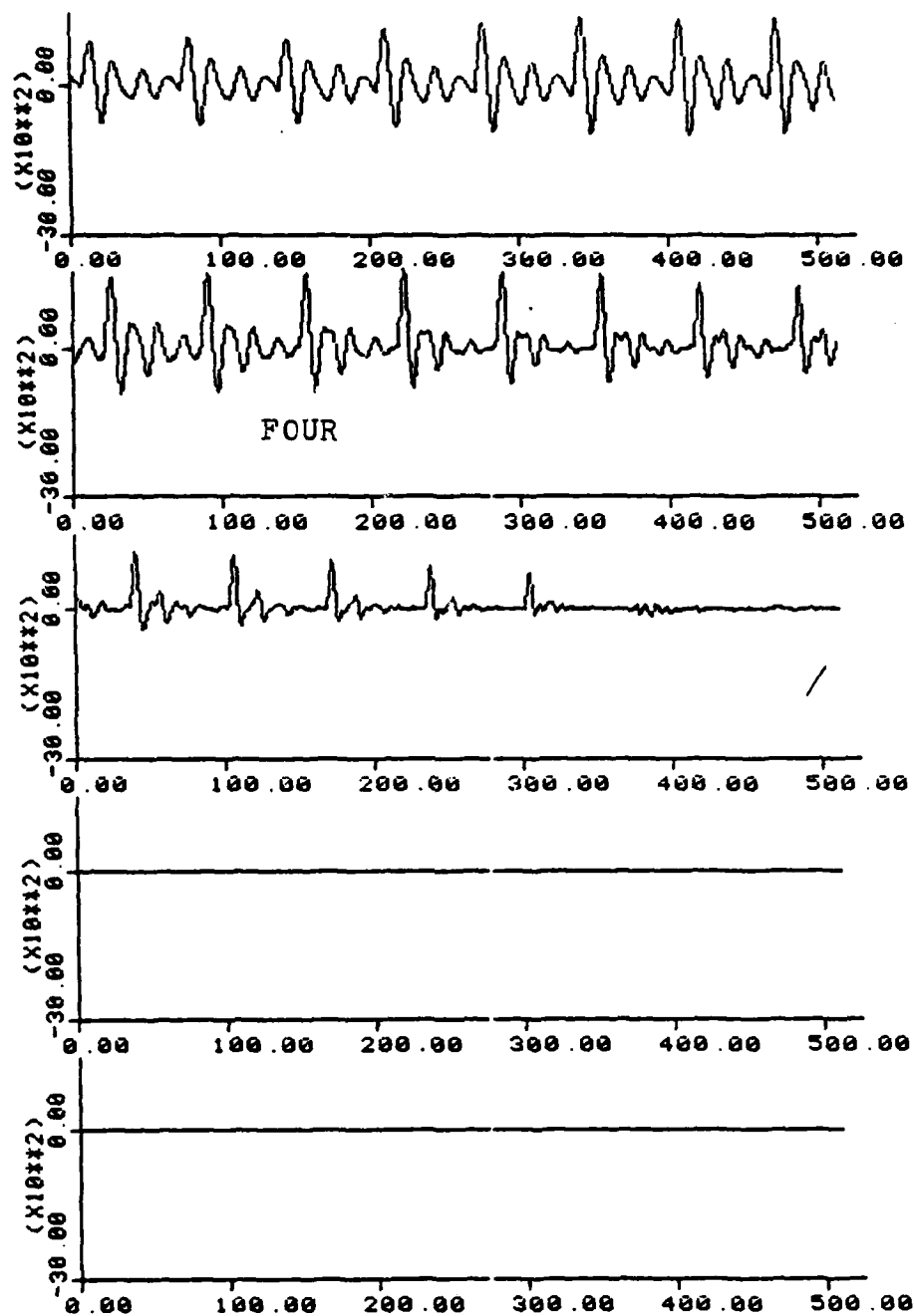


Figure IV-12.9 "CNE...TWO...THREE...FOUR"

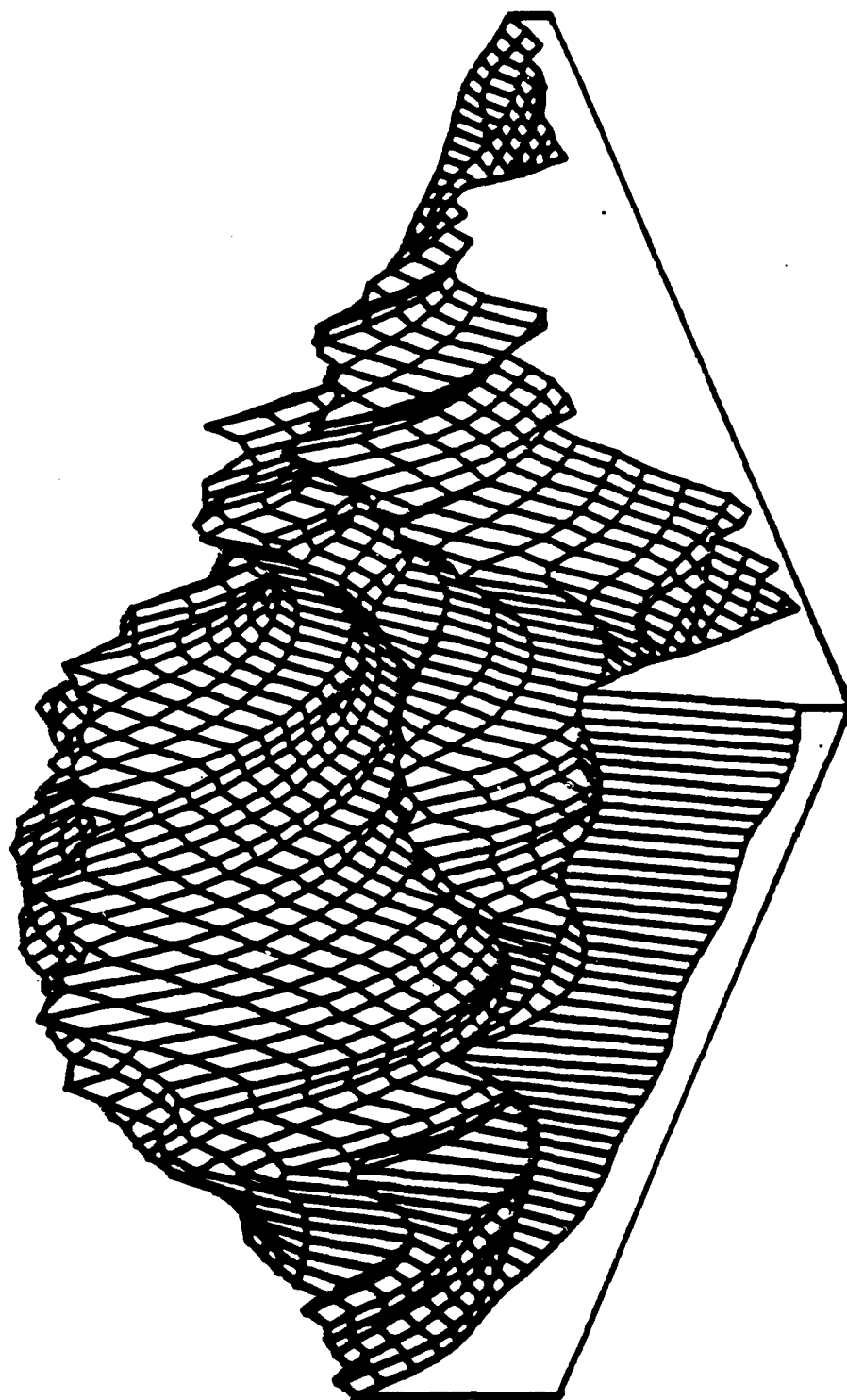


Figure IV-13 "ONE"

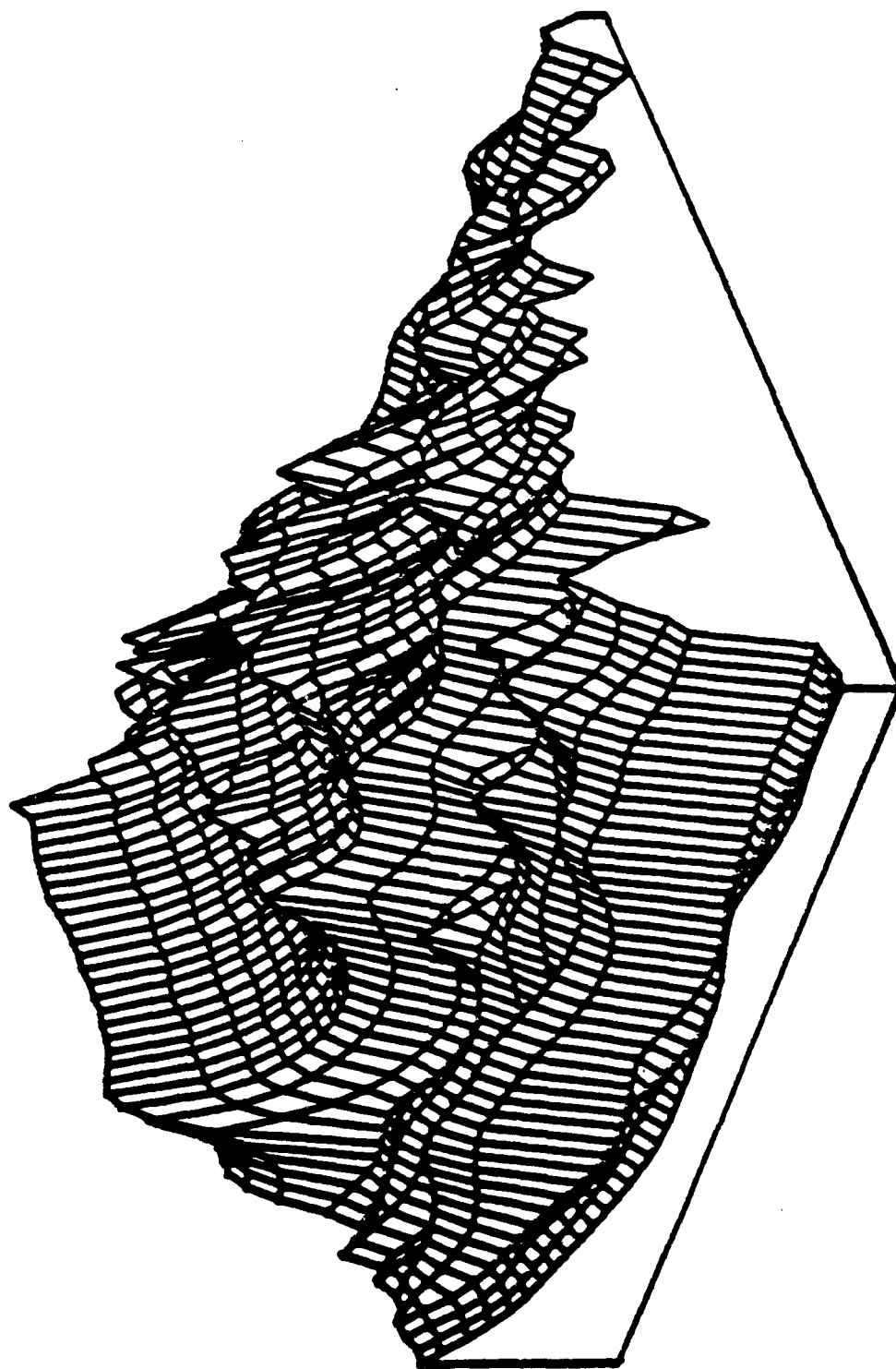


Figure IV-14 "TWO"

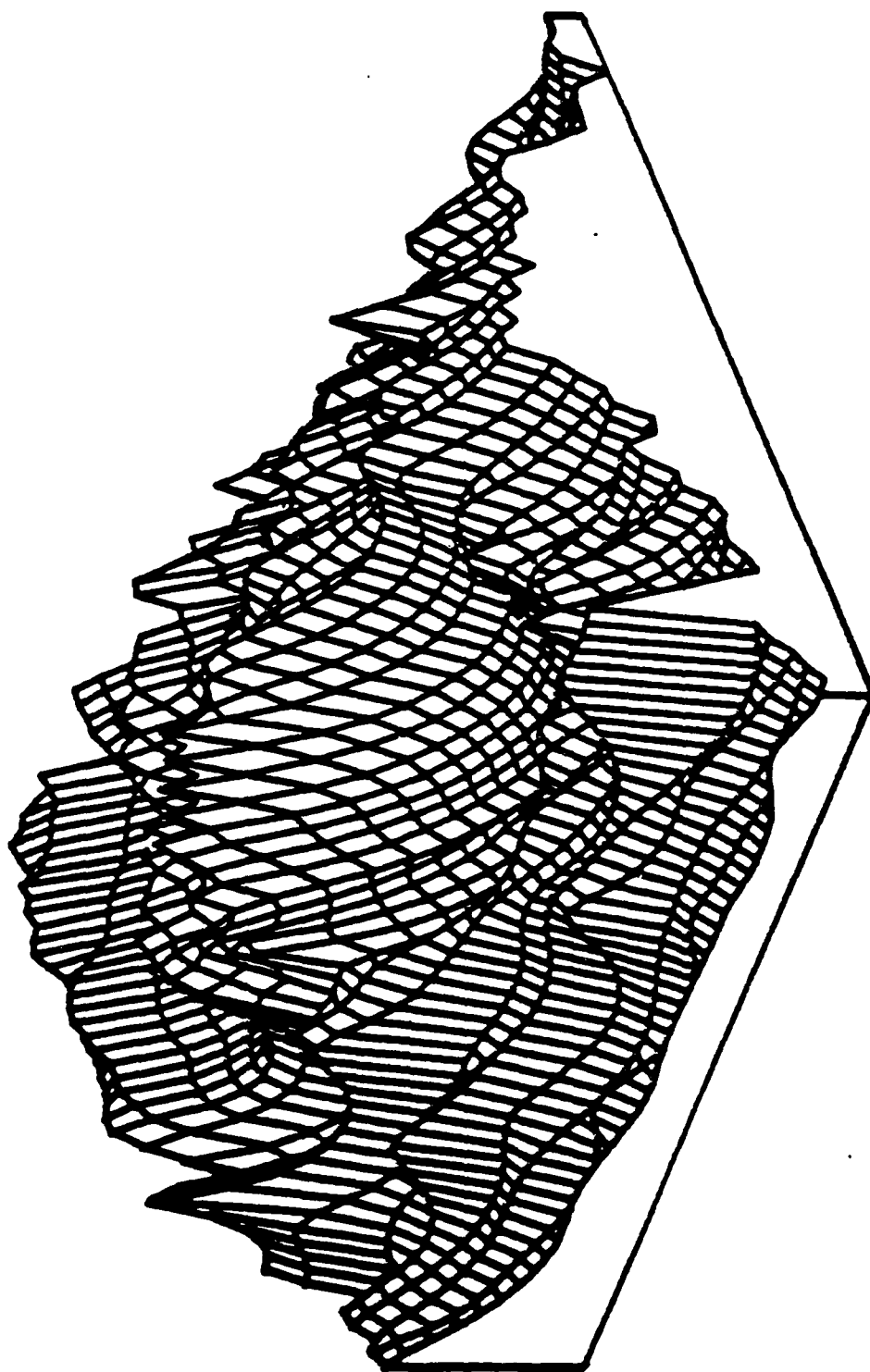


Figure IV-15 "THREE"

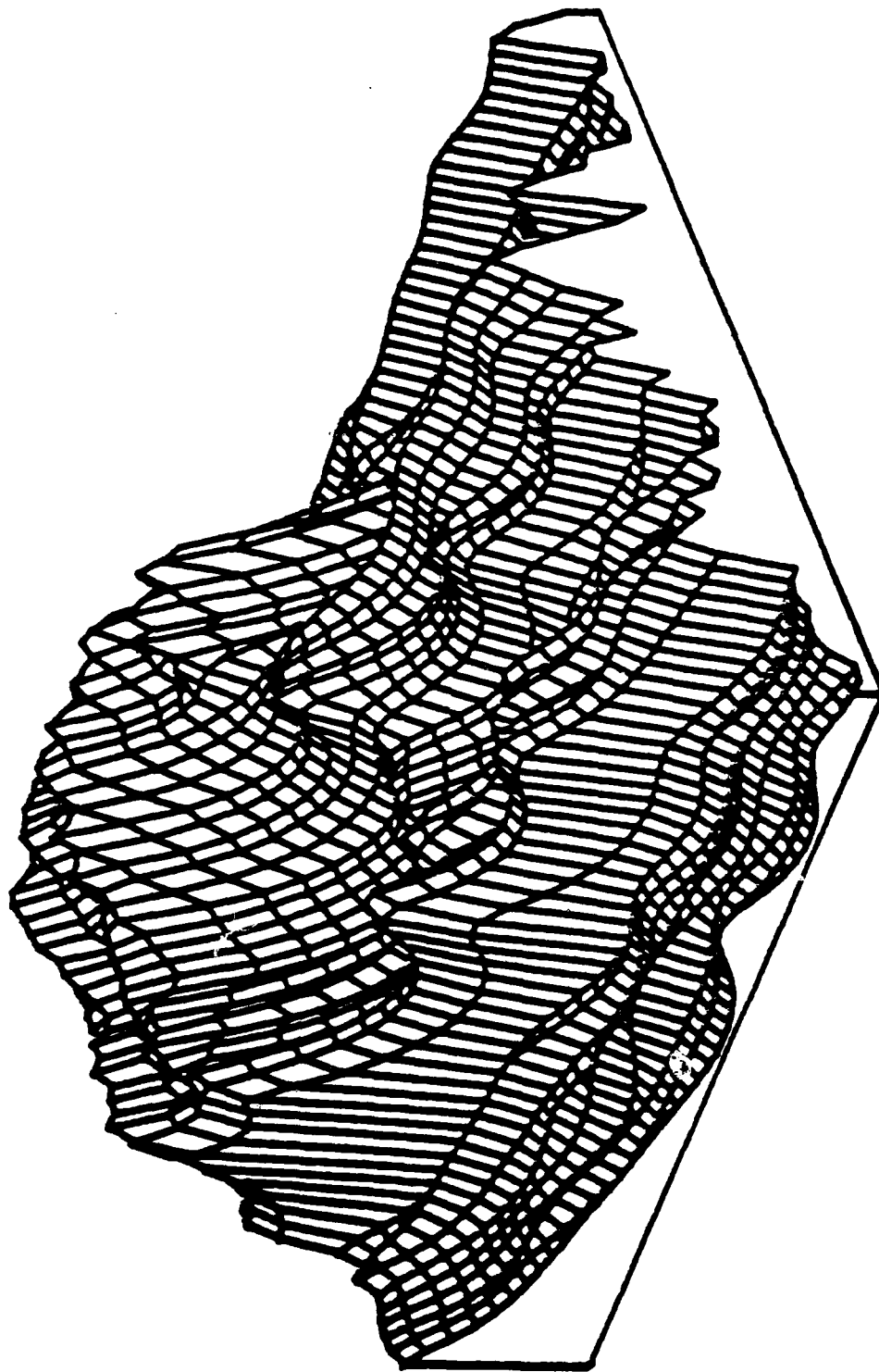
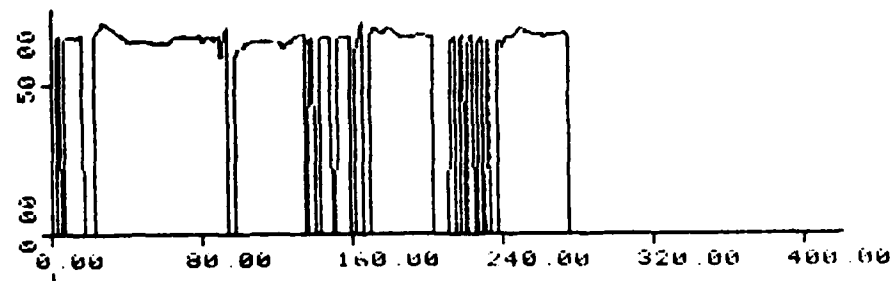
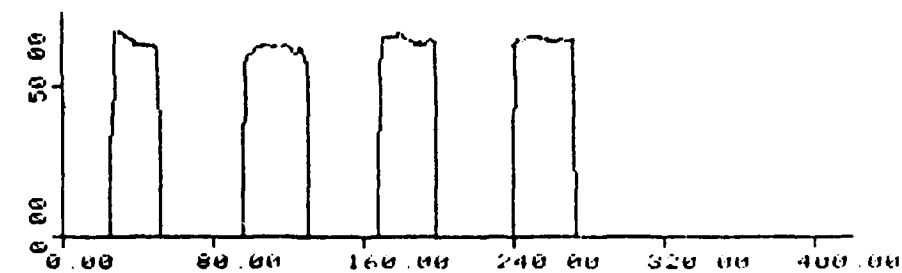


Figure IV-16 "FOUR"



(a) without noise



(b) with noise added

Figure IV-17 Pitch plot of "ONE...TWC...THREE...FCUR"

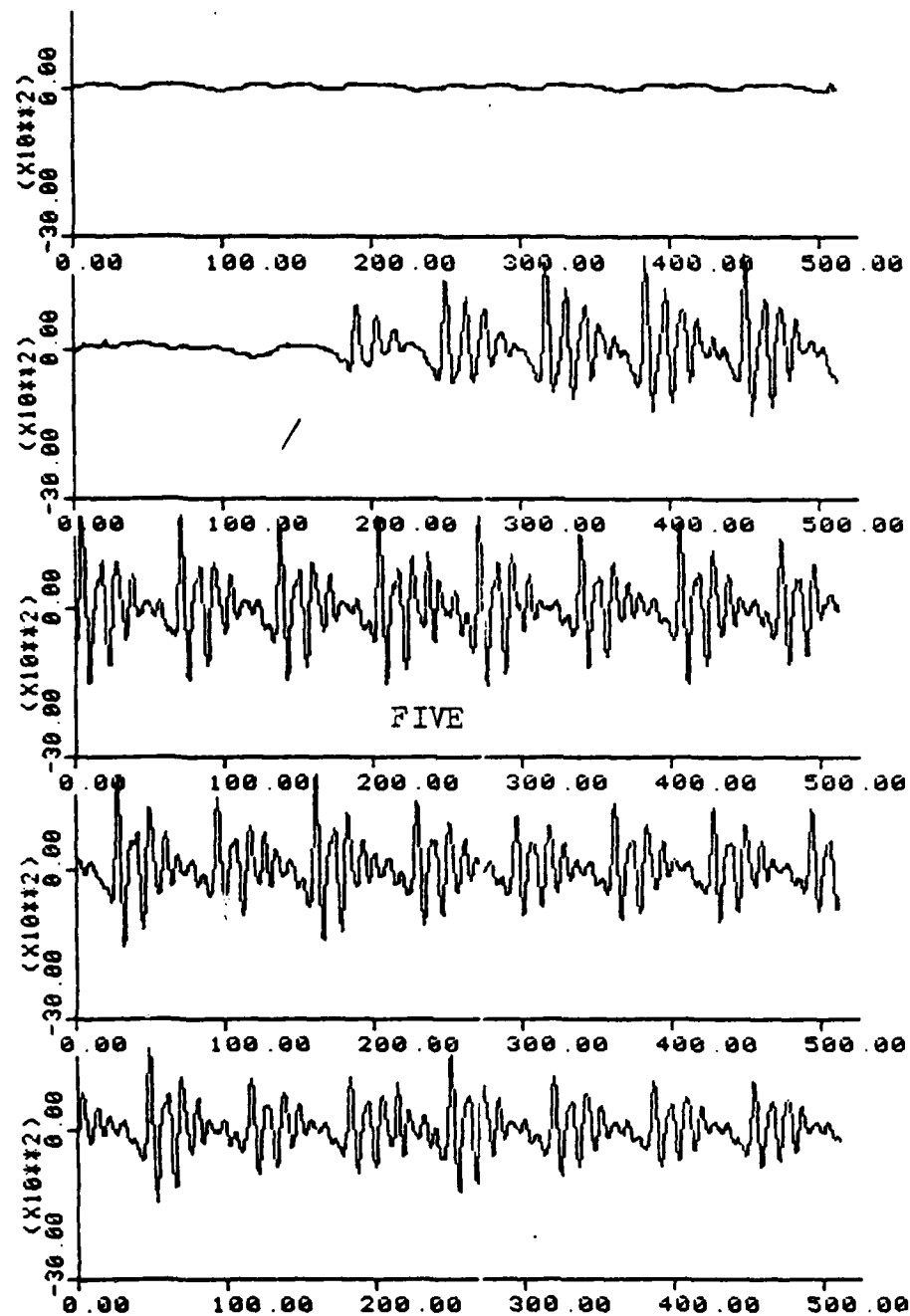


Figure IV-18.1 "FIVE...SIX...SEVEN...EIGHT"

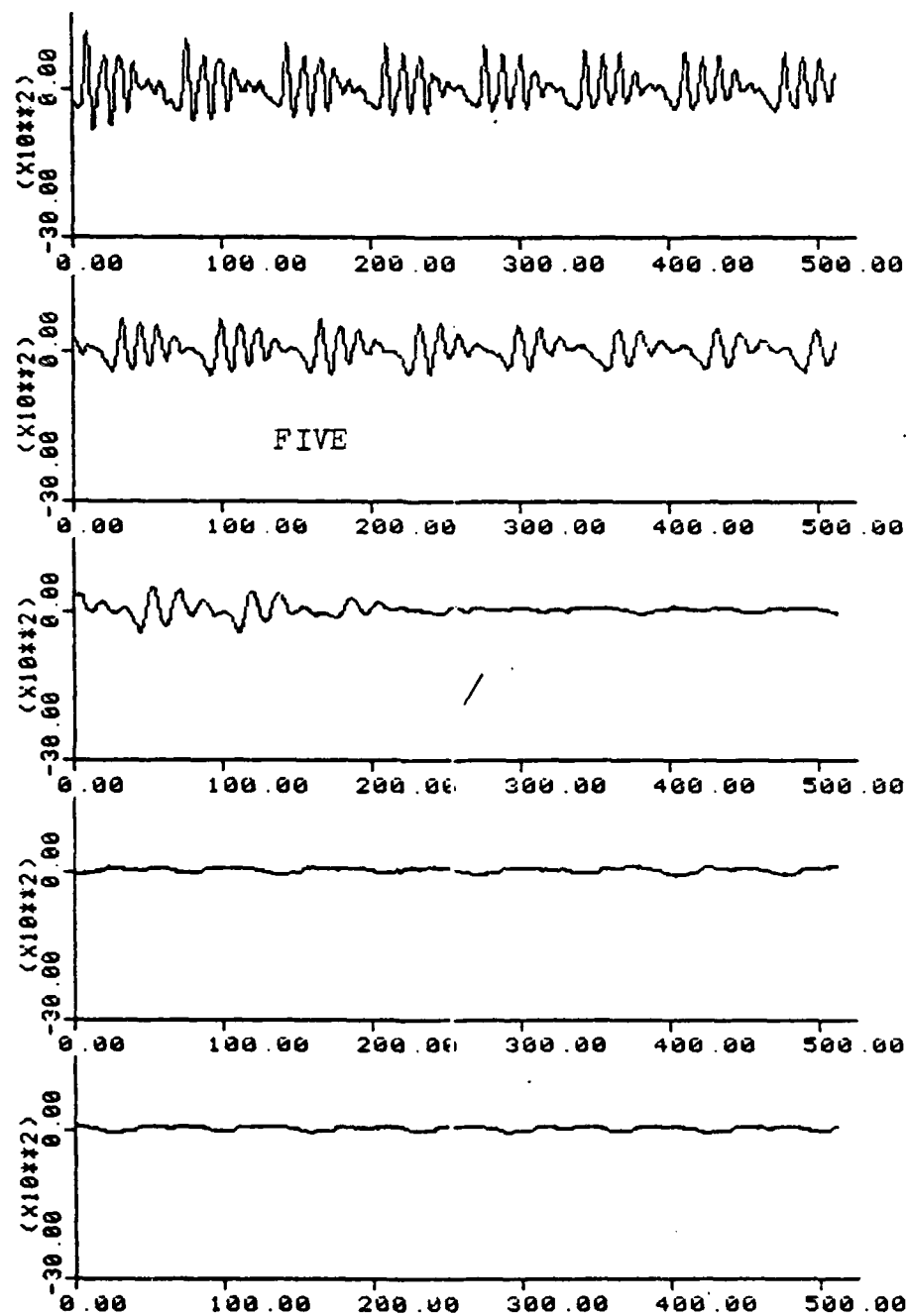


Figure IV-18.2 "FIVE...SIX..SEVEN...EIGHT"

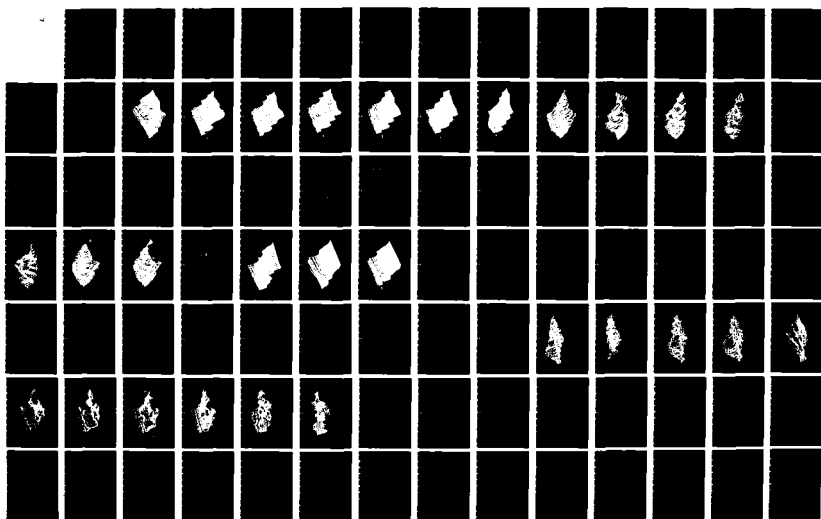
AD-A138 008

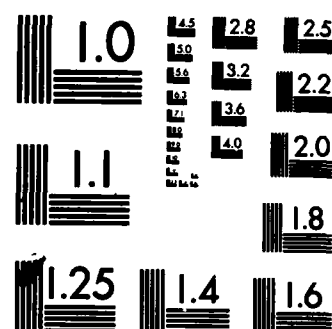
A RECURSIVE LINEAR PREDICTIVE VOCODER(U) AIR FORCE INST 3/4
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
W A JANSSEN DEC 83 AFIT/GE/EE/83D-33

UNCLASSIFIED

F/G 17/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

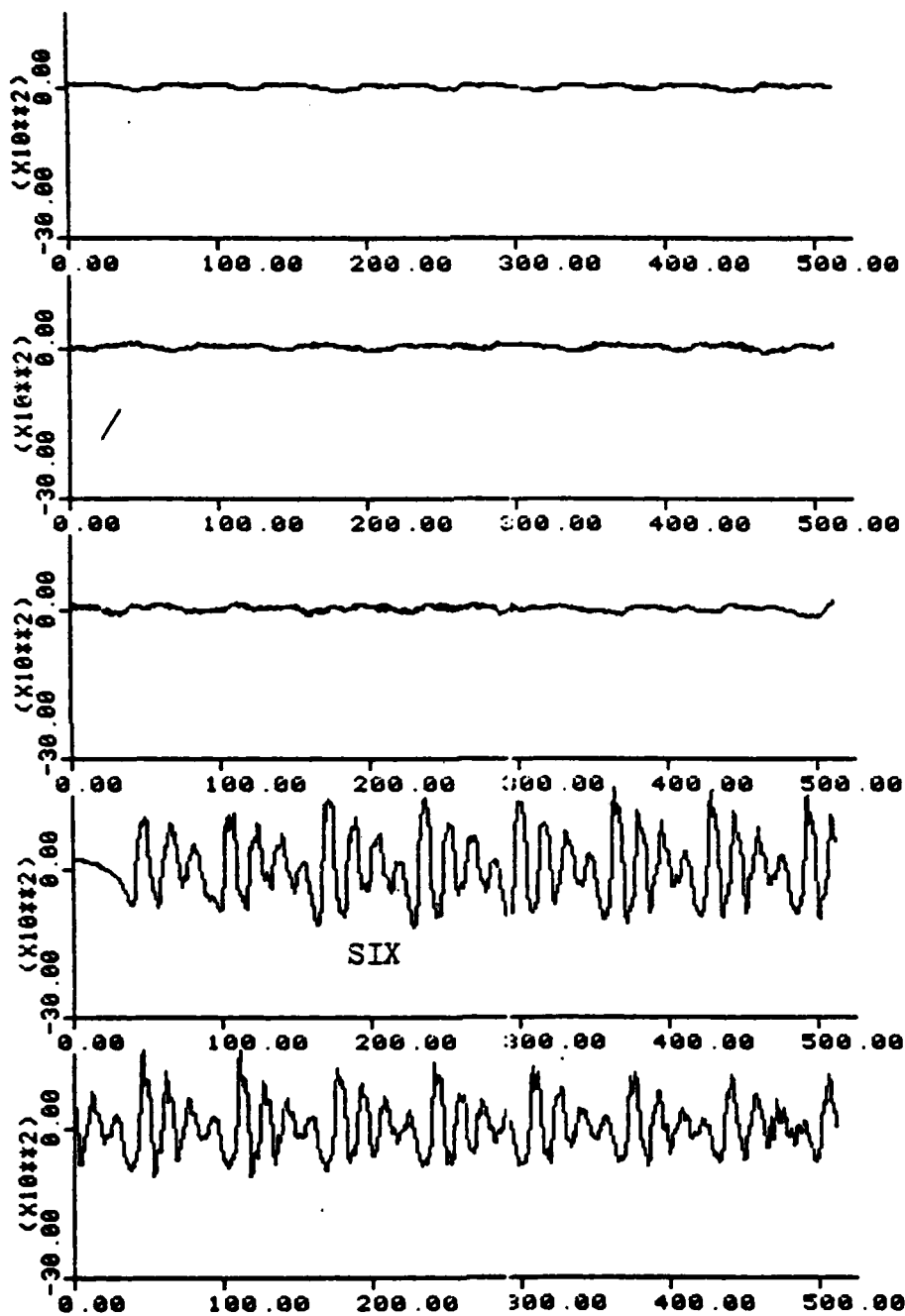


Figure IV-18.3 "FIVE...SIX...SEVEN...EIGHT"

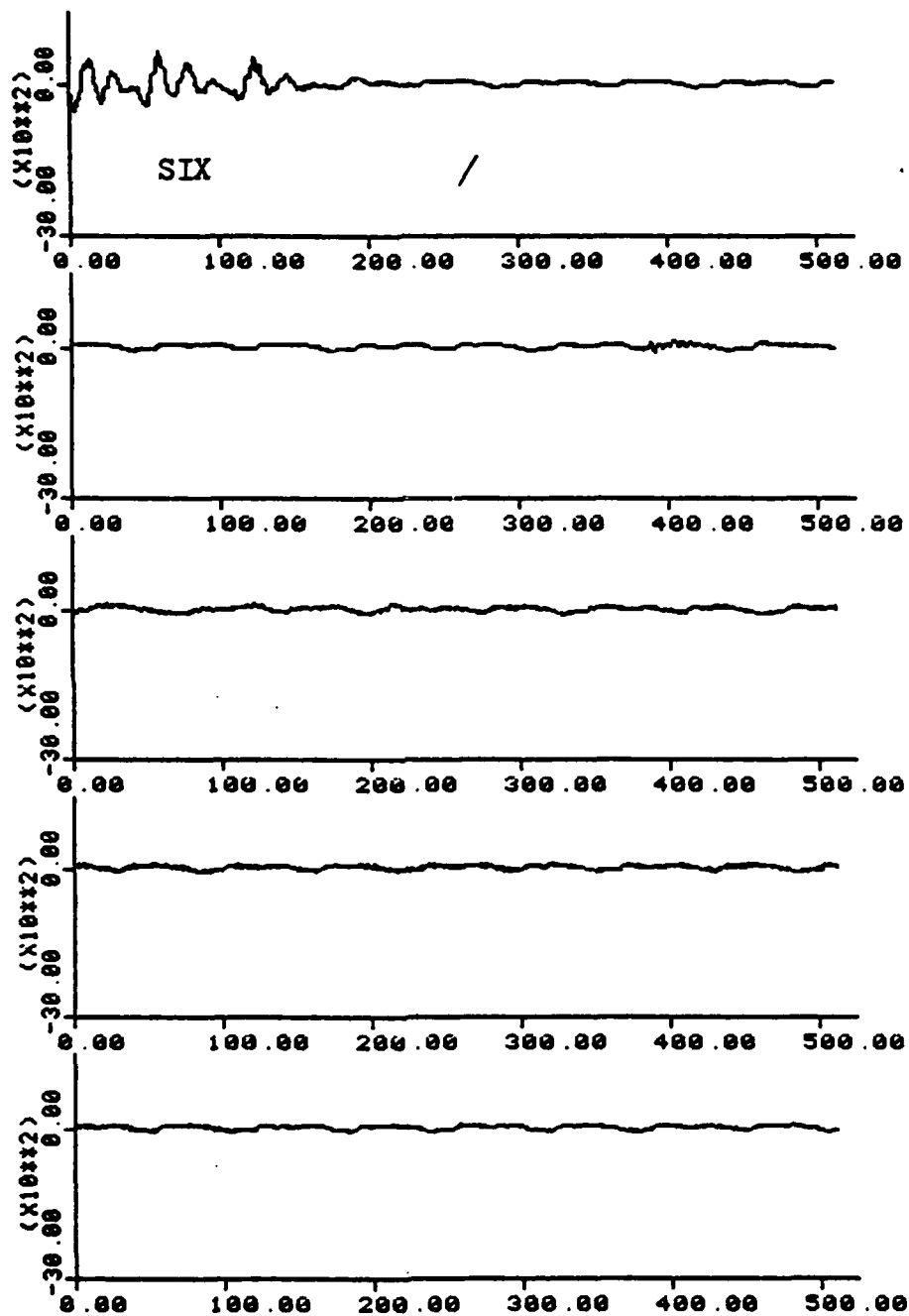


Figure IV-18.4 "FIVE...SIX...SEVEN...EIGHT"

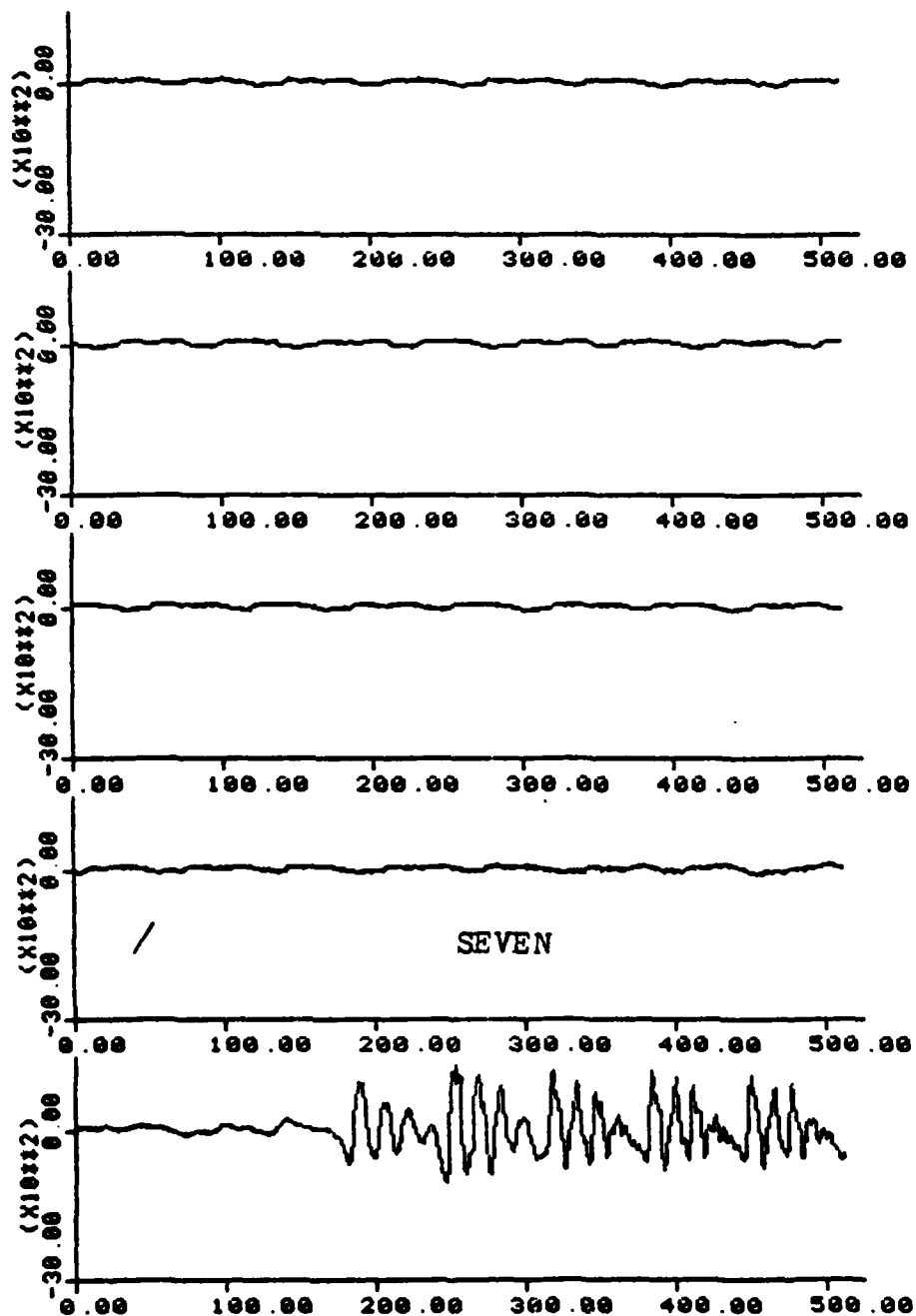


Figure IV-18.5 "FIVE...SIX...SEVEN...EIGHT"

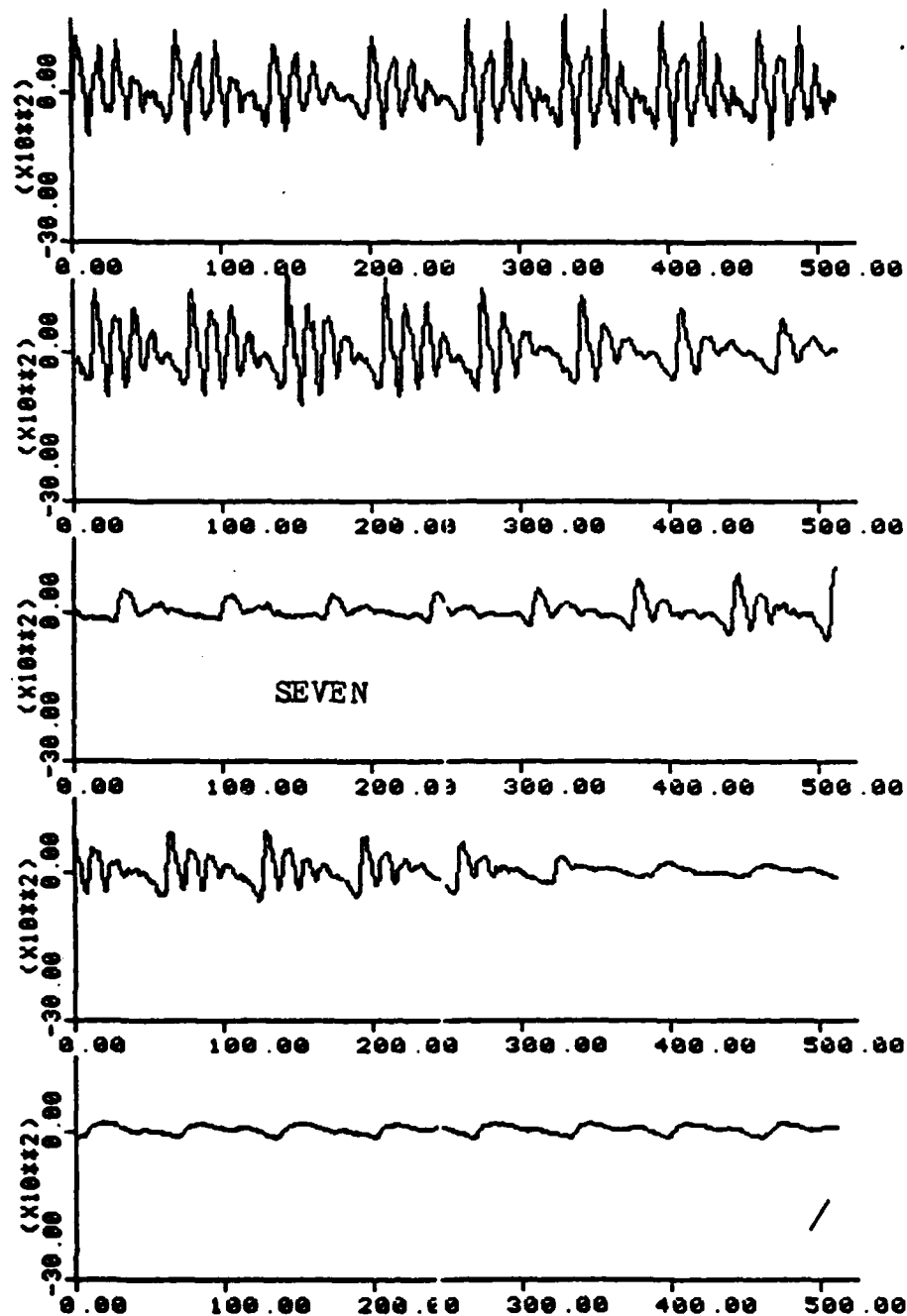


Figure IV-18.6 "FIVE...SIX...SEVEN...EIGHT"

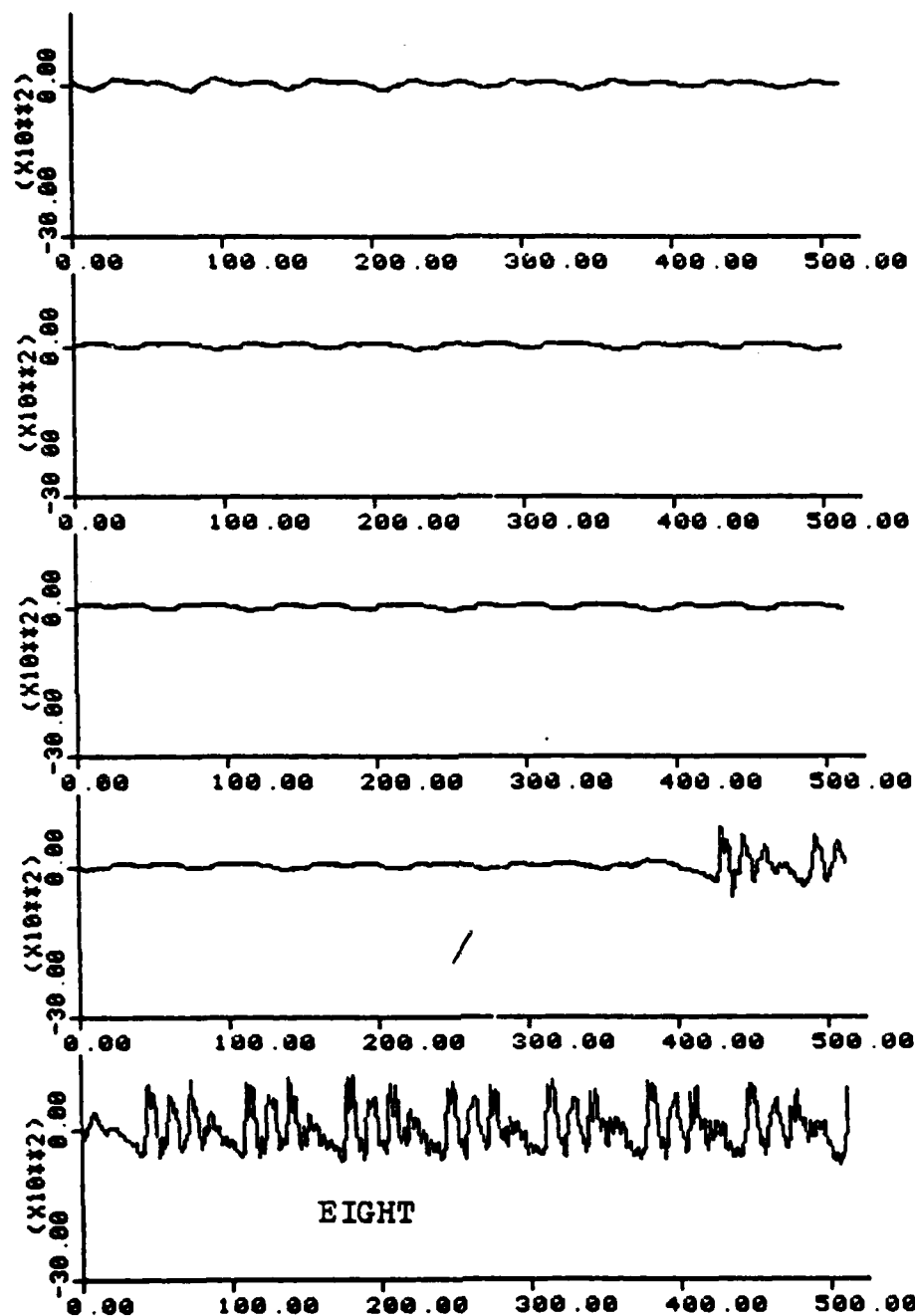


Figure IV-18.7 "FIVE...SIX...SEVEN...EIGHT"

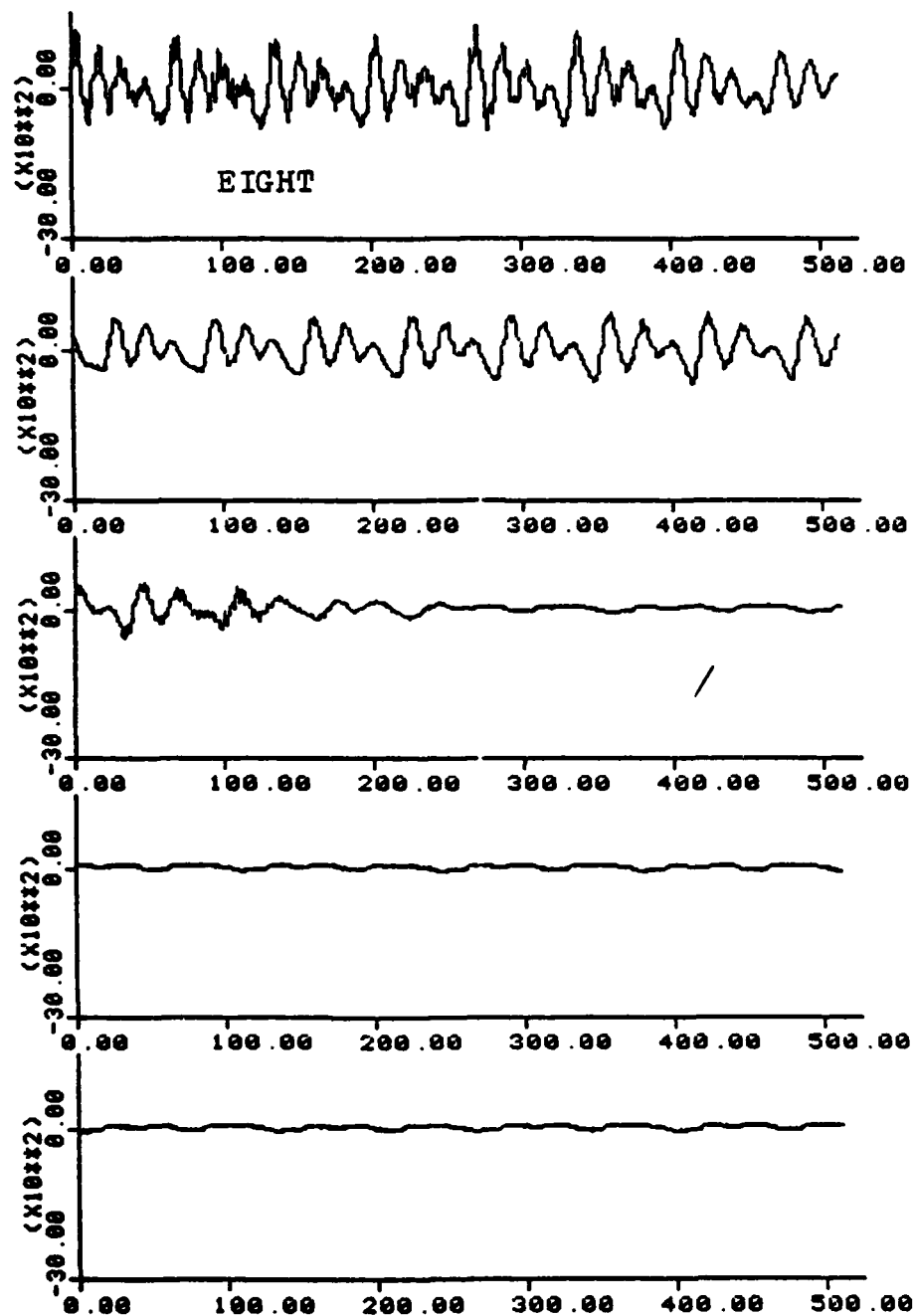


Figure IV-18.8 "FIVE...SIX...SEVEN...EIGHT"

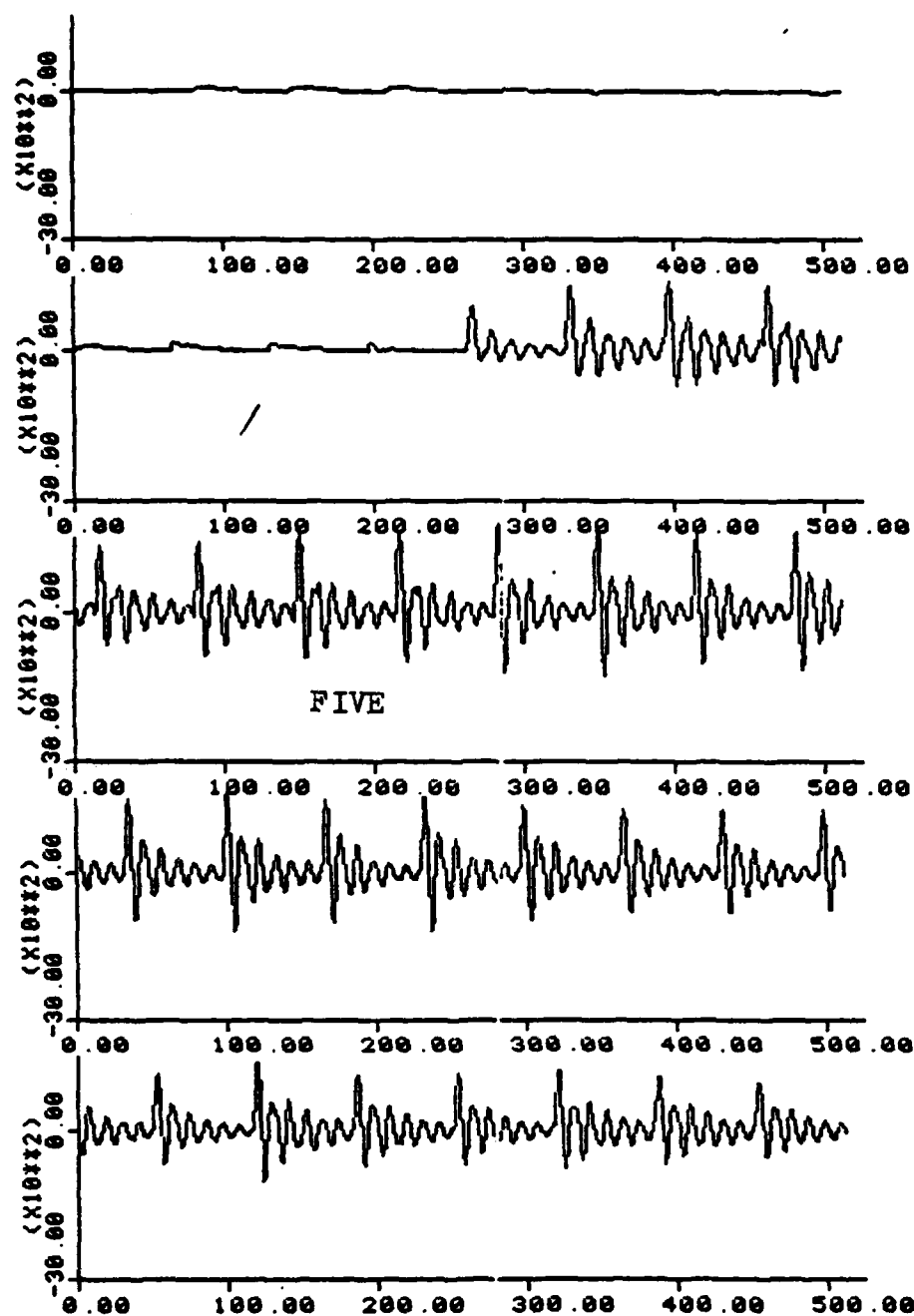


Figure IV-19.1 "FIVE...SIX...SEVEN...EIGHT"

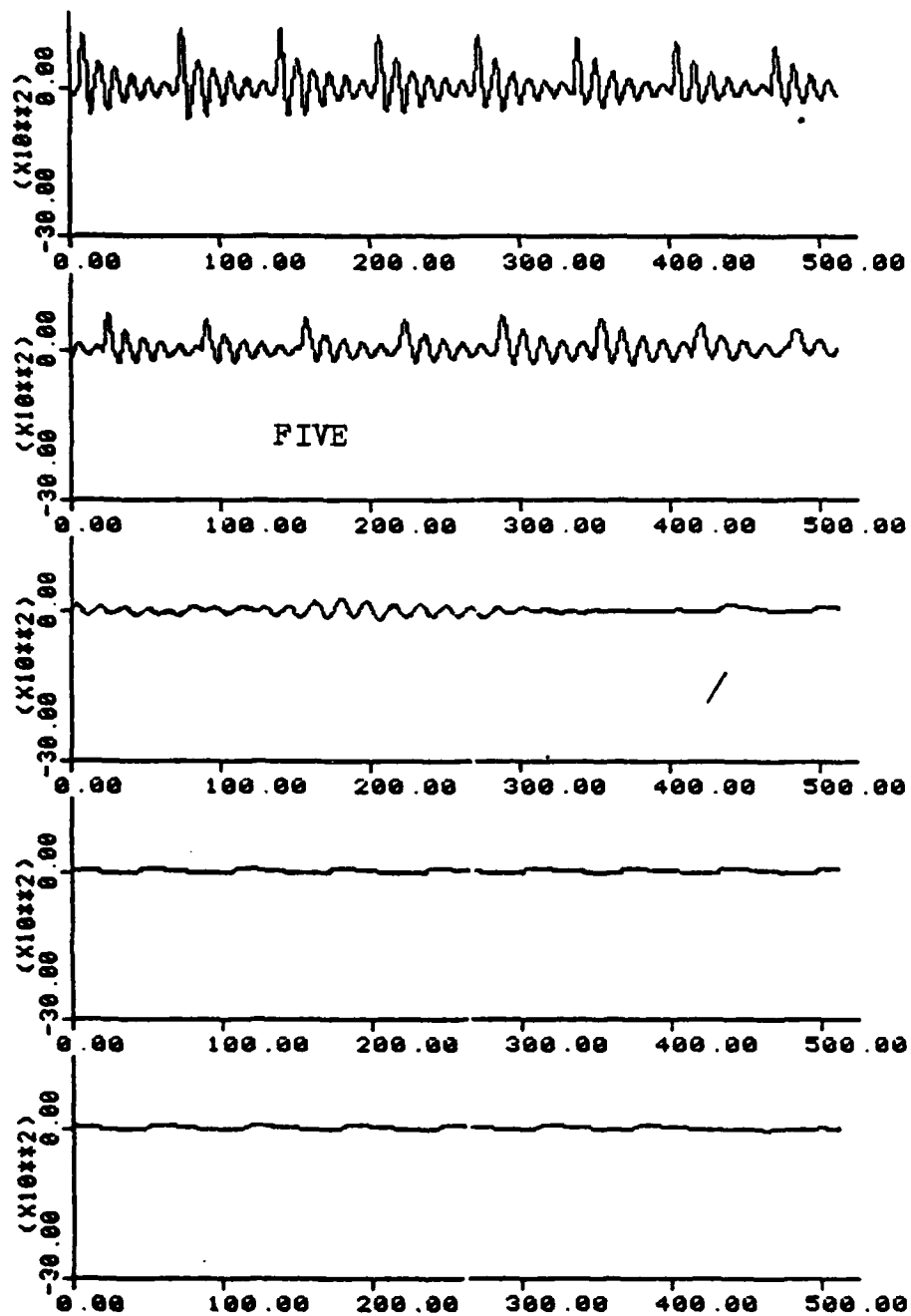


Figure IV-19.2 "FIVE...SIX...SEVEN...EIGHT"

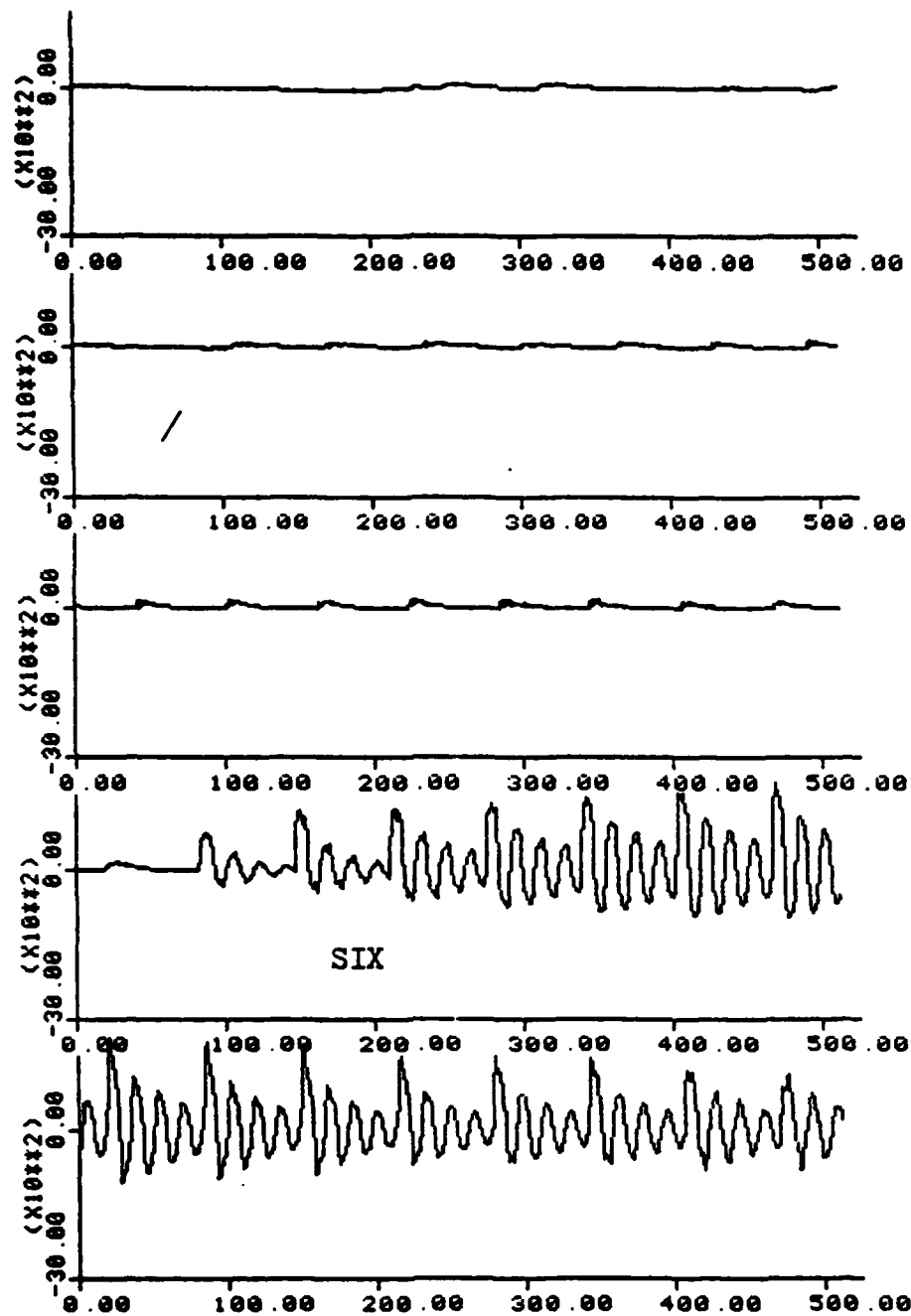


Figure IV-19.3 "FIVE...SIX...SEVEN...EIGHT"

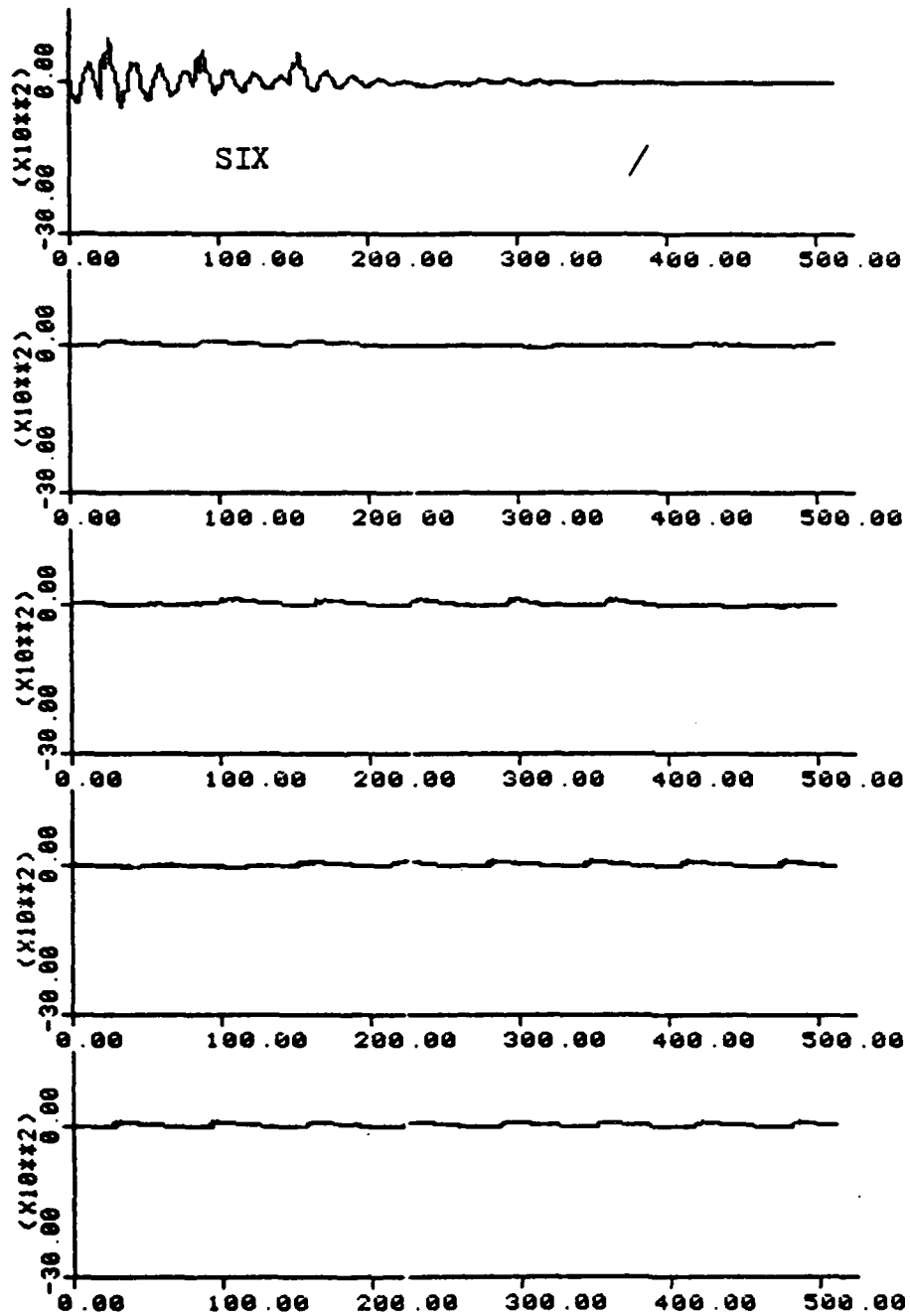


Figure IV-19.4 "FIVE...SIX...SEVEN...EIGHT"

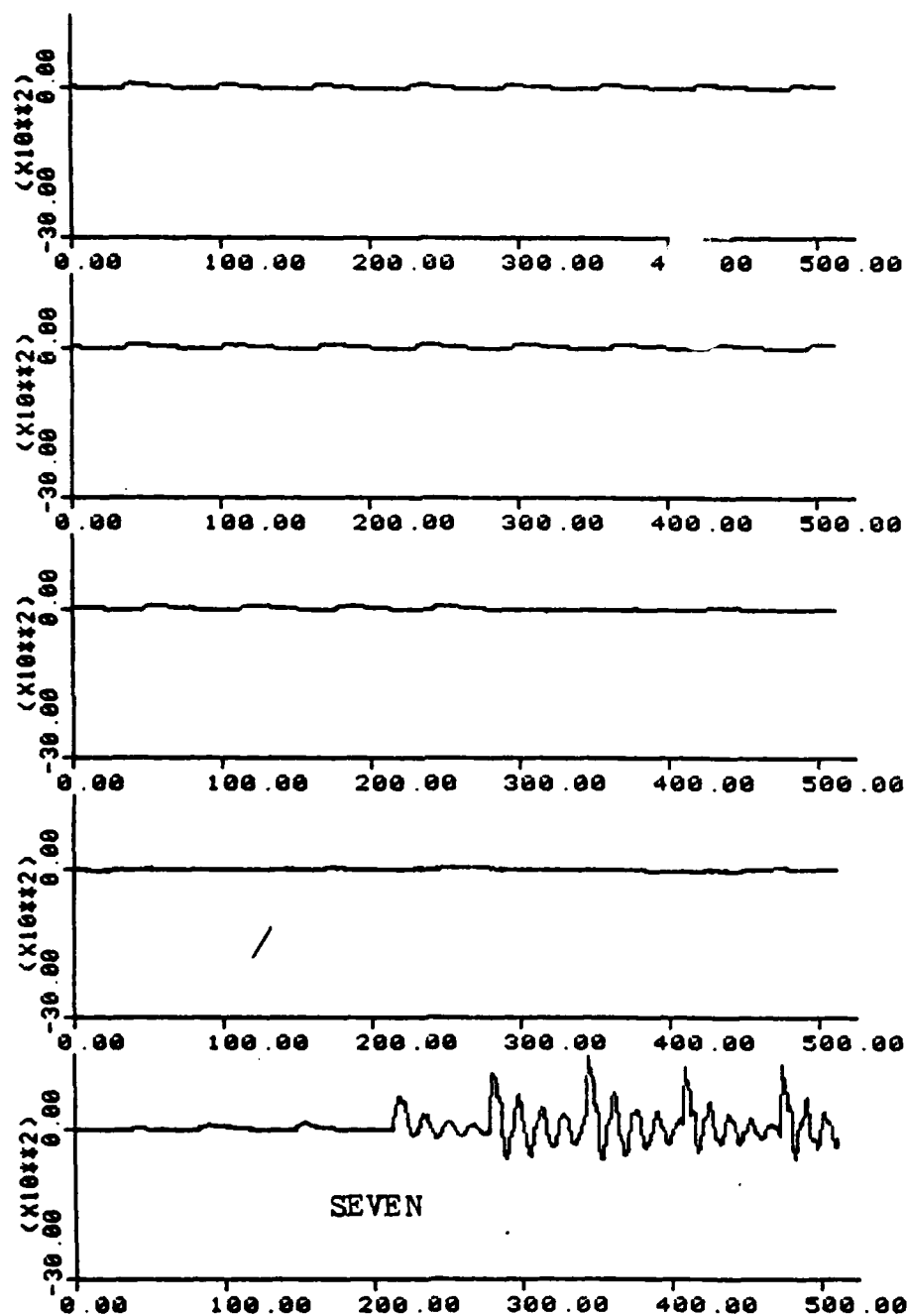


Figure IV-19.5 "FIVE...SIX...SEVEN...EIGHT"

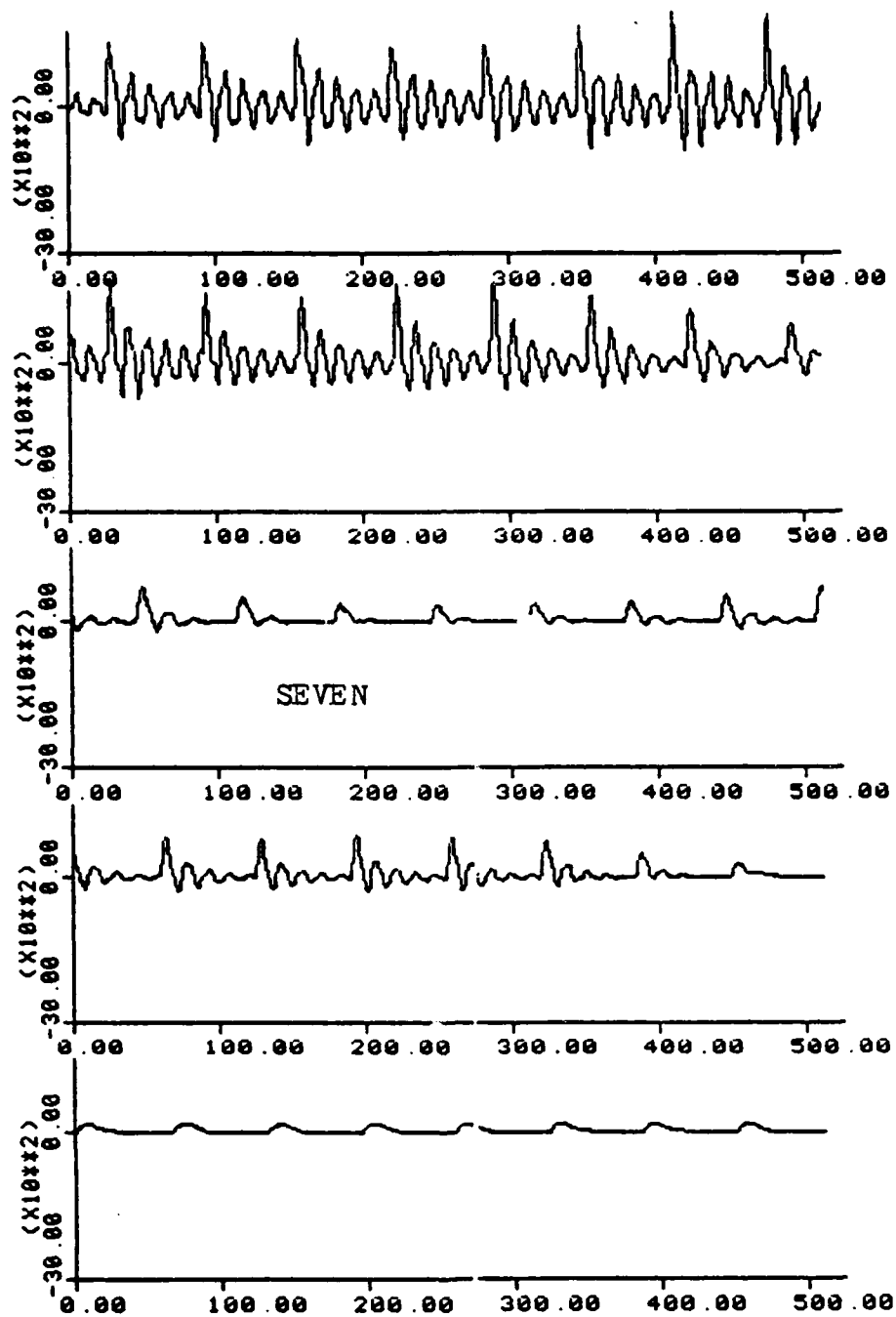


Figure IV-19.6 "FIVE...SIX...SEVEN...EIGHT"

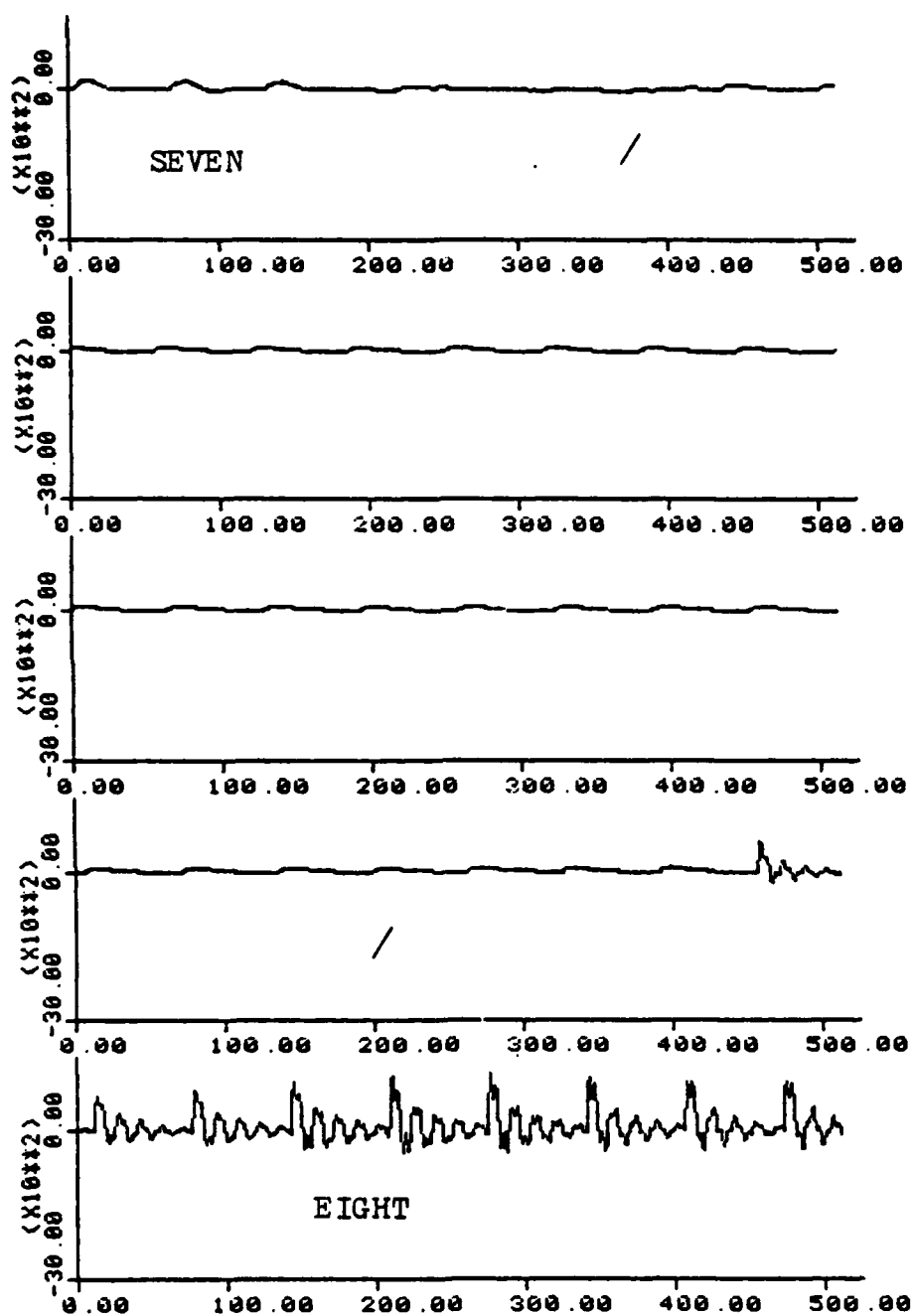


Figure IV-19.7 "FIVE...SIX...SEVEN...EIGHT"

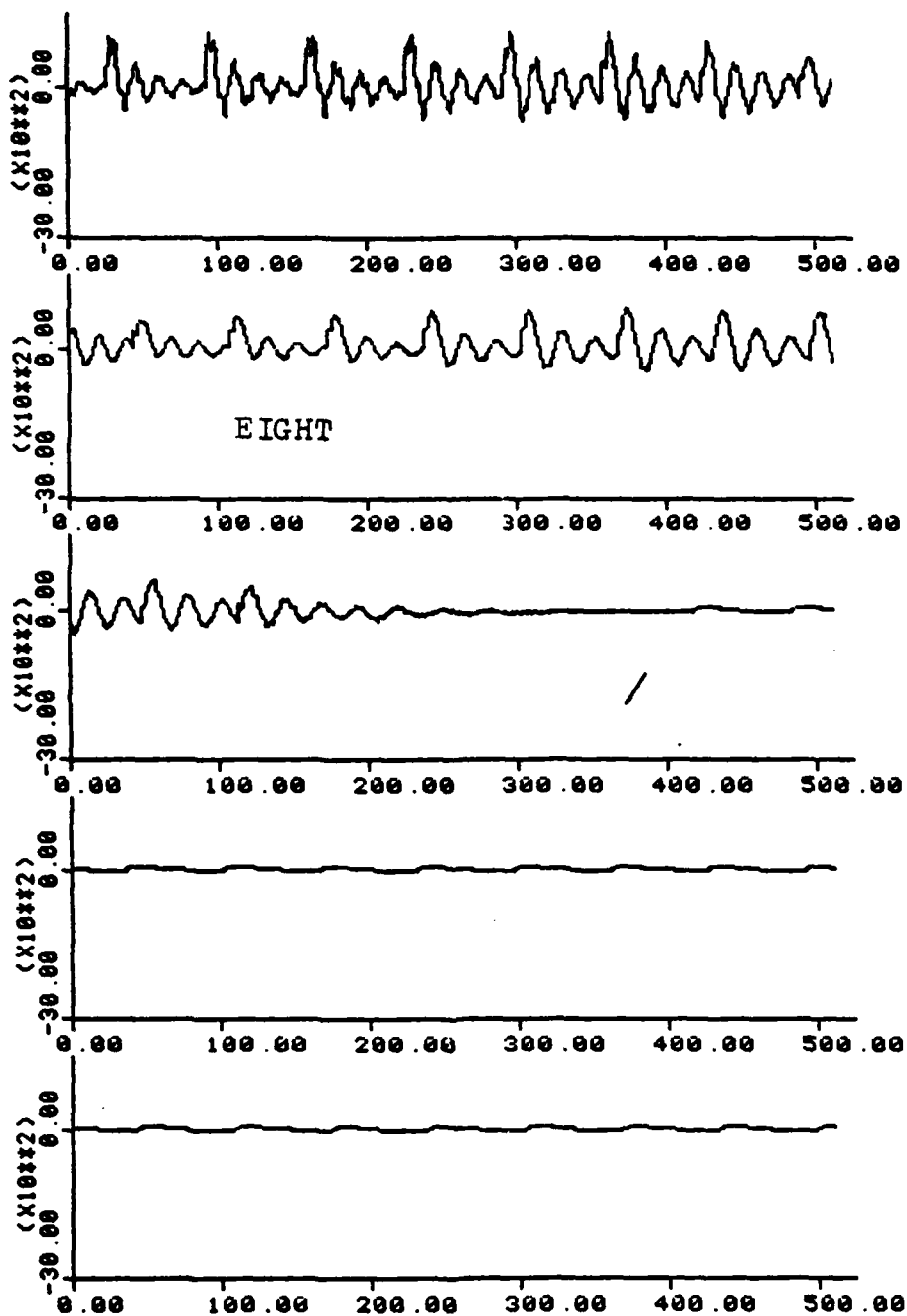


Figure IV-19.8 "FIVE...SIX...SEVEN...EIGHT"

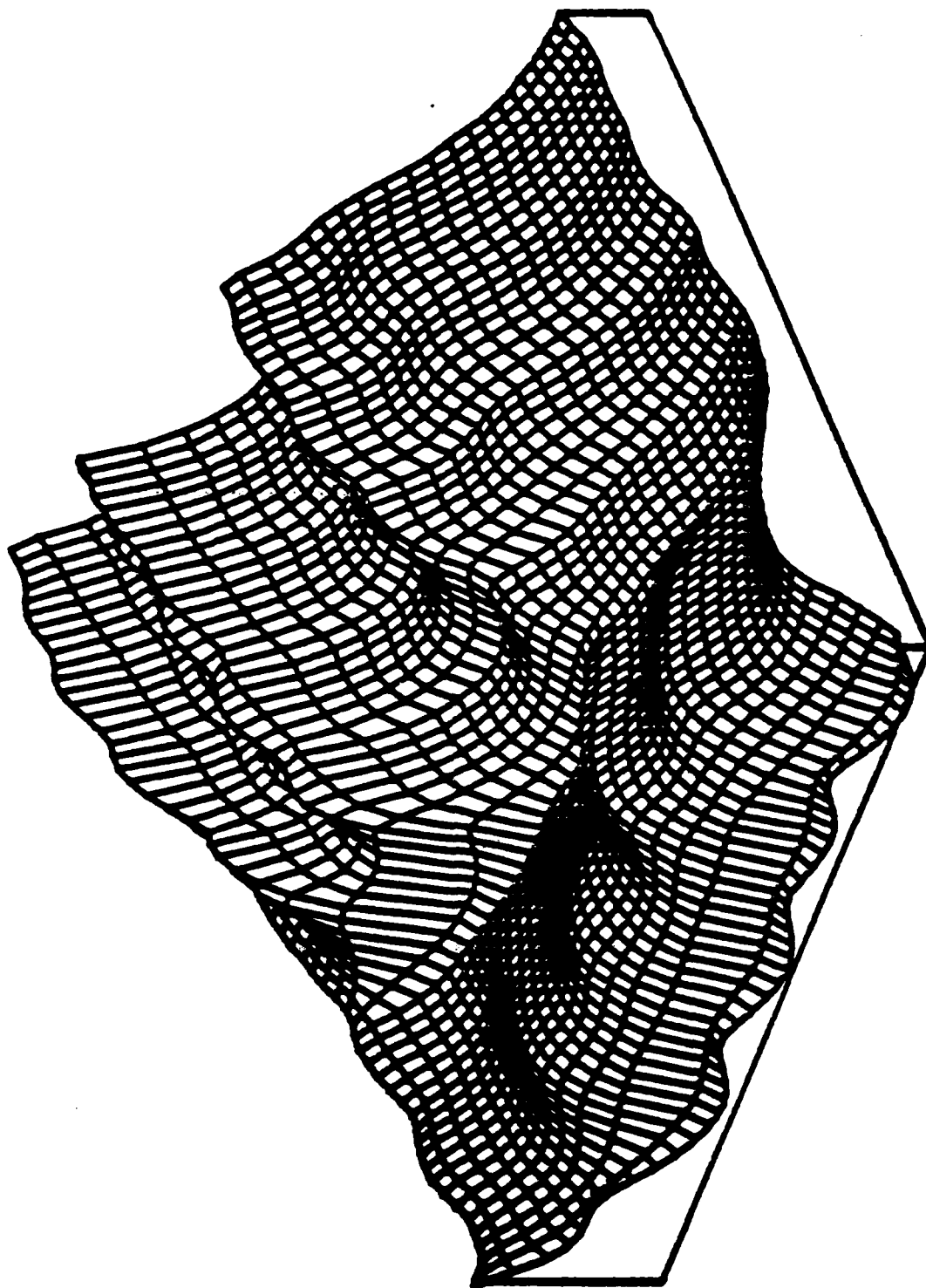


Figure IV-20.1.2.a "FIVE"

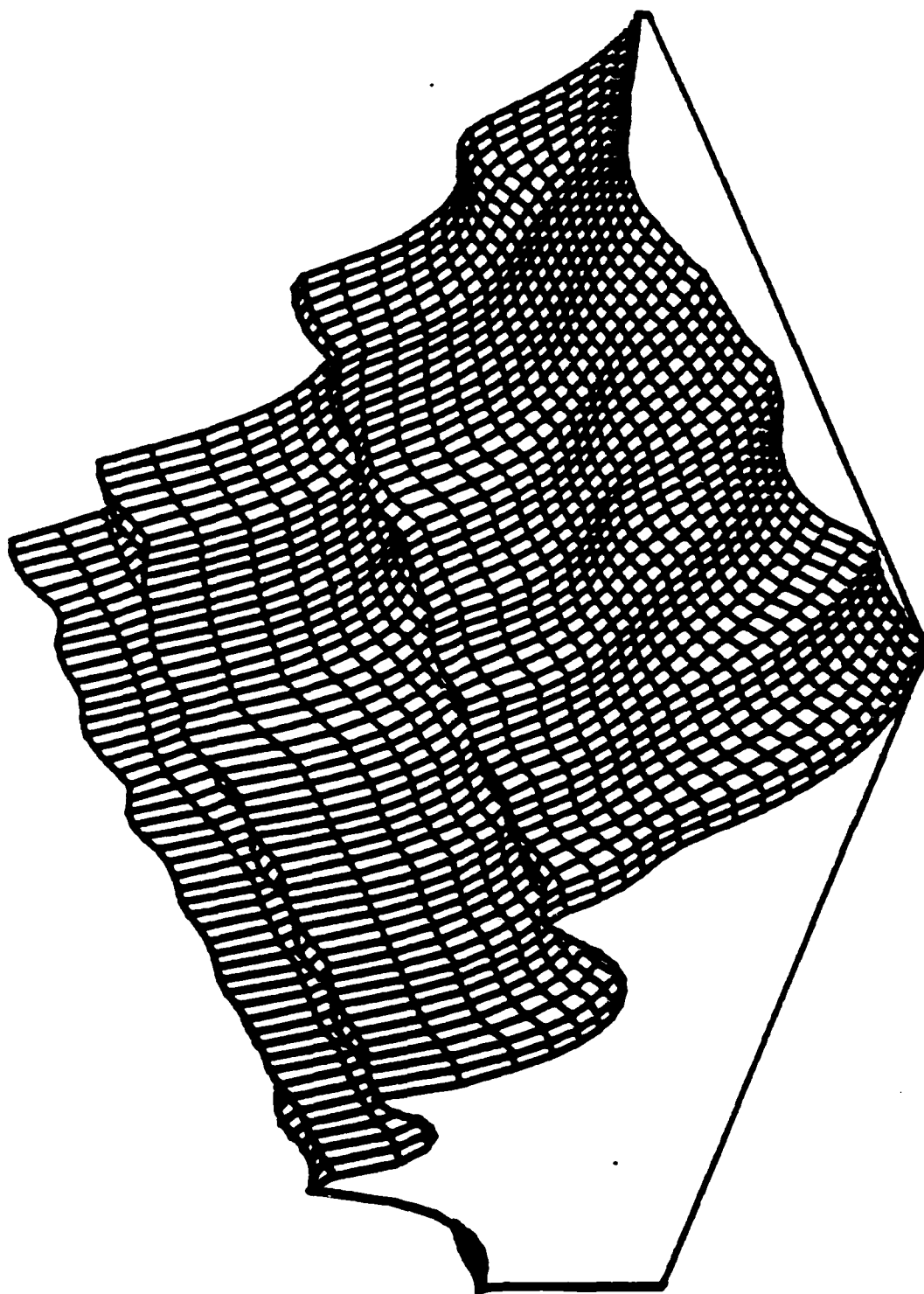


Figure IV-20.1.3.a "FIVE"

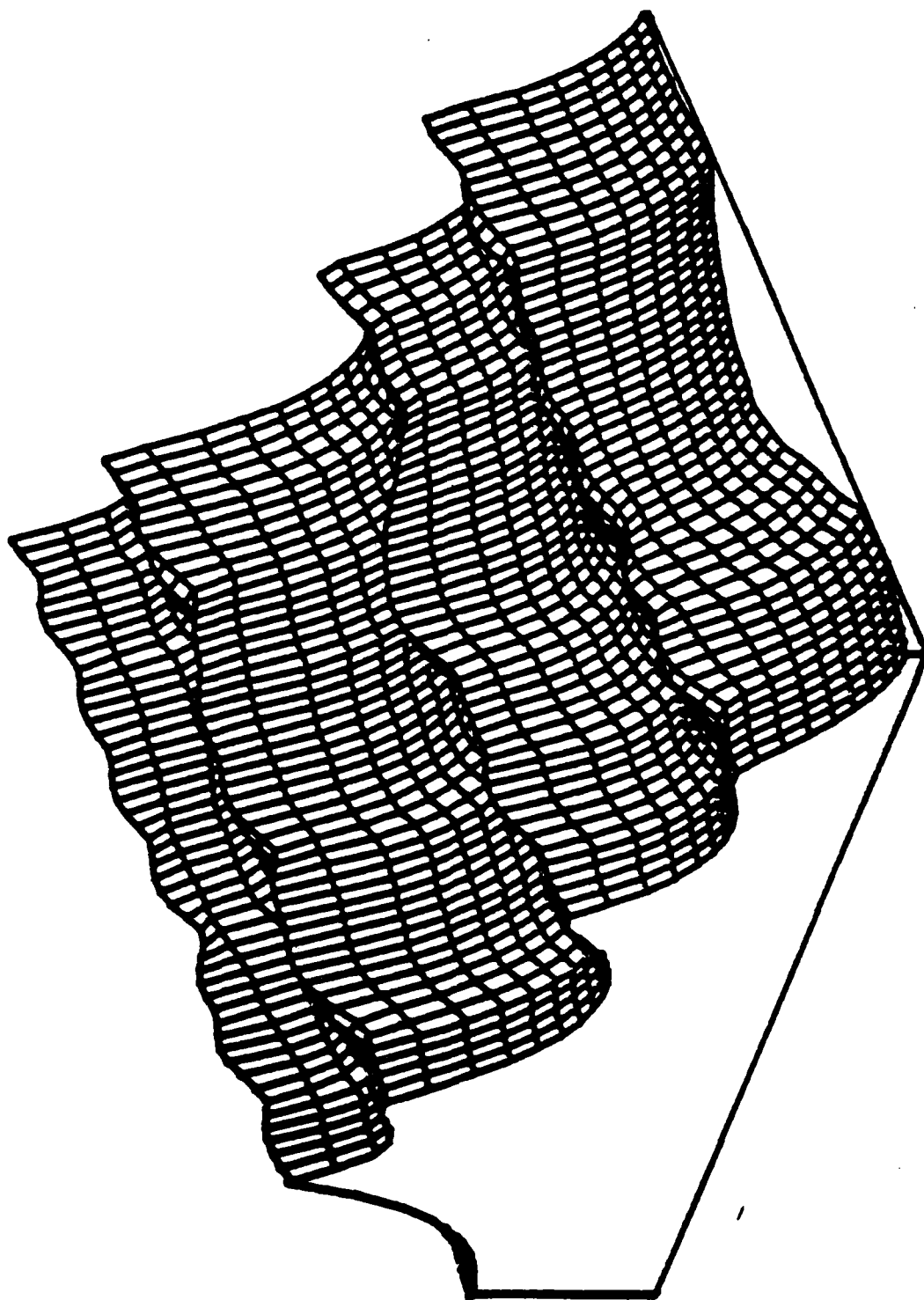


Figure IV-20.1.4.a "FIVE"

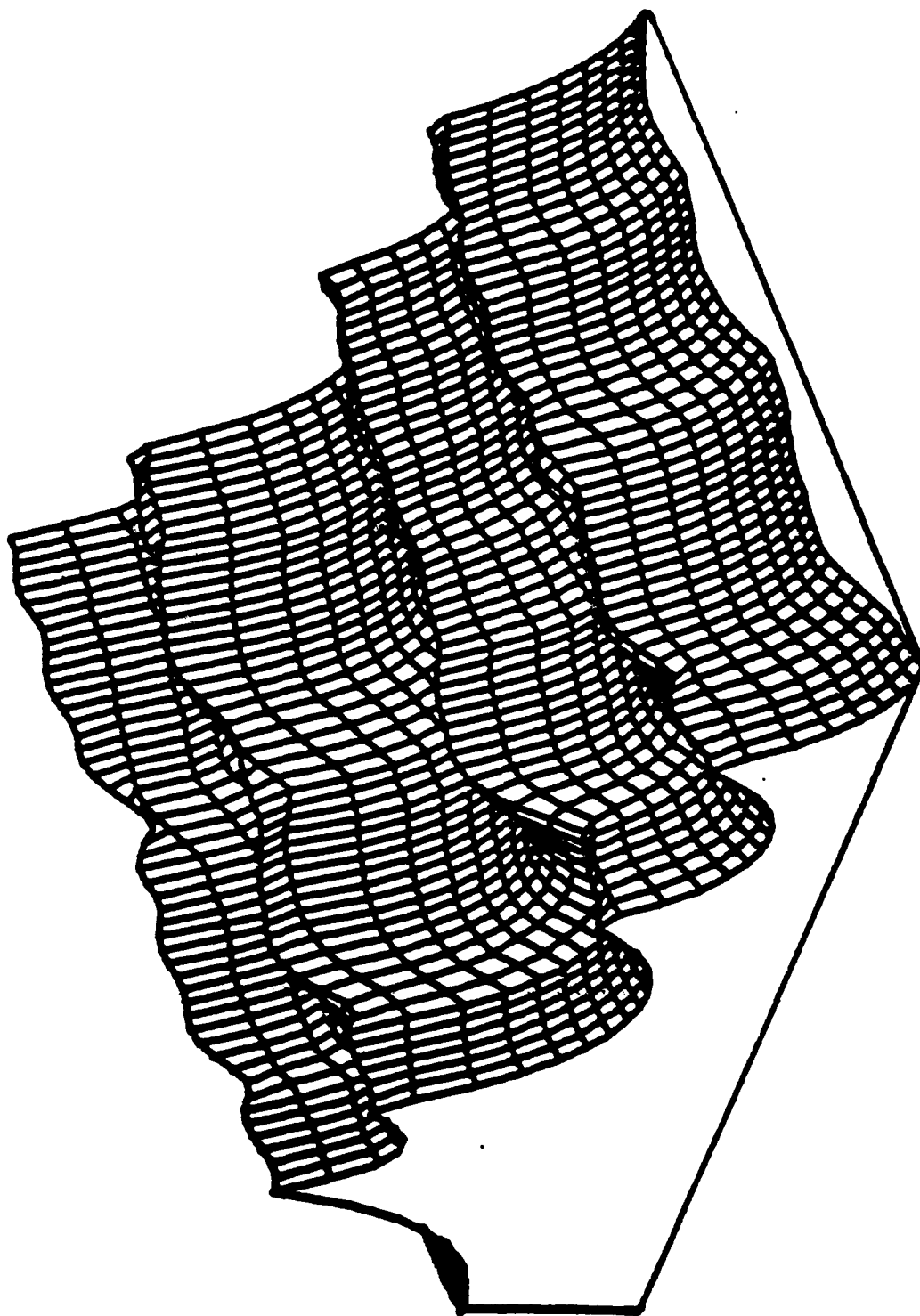


Figure IV-20.1.5.a "FIVE"

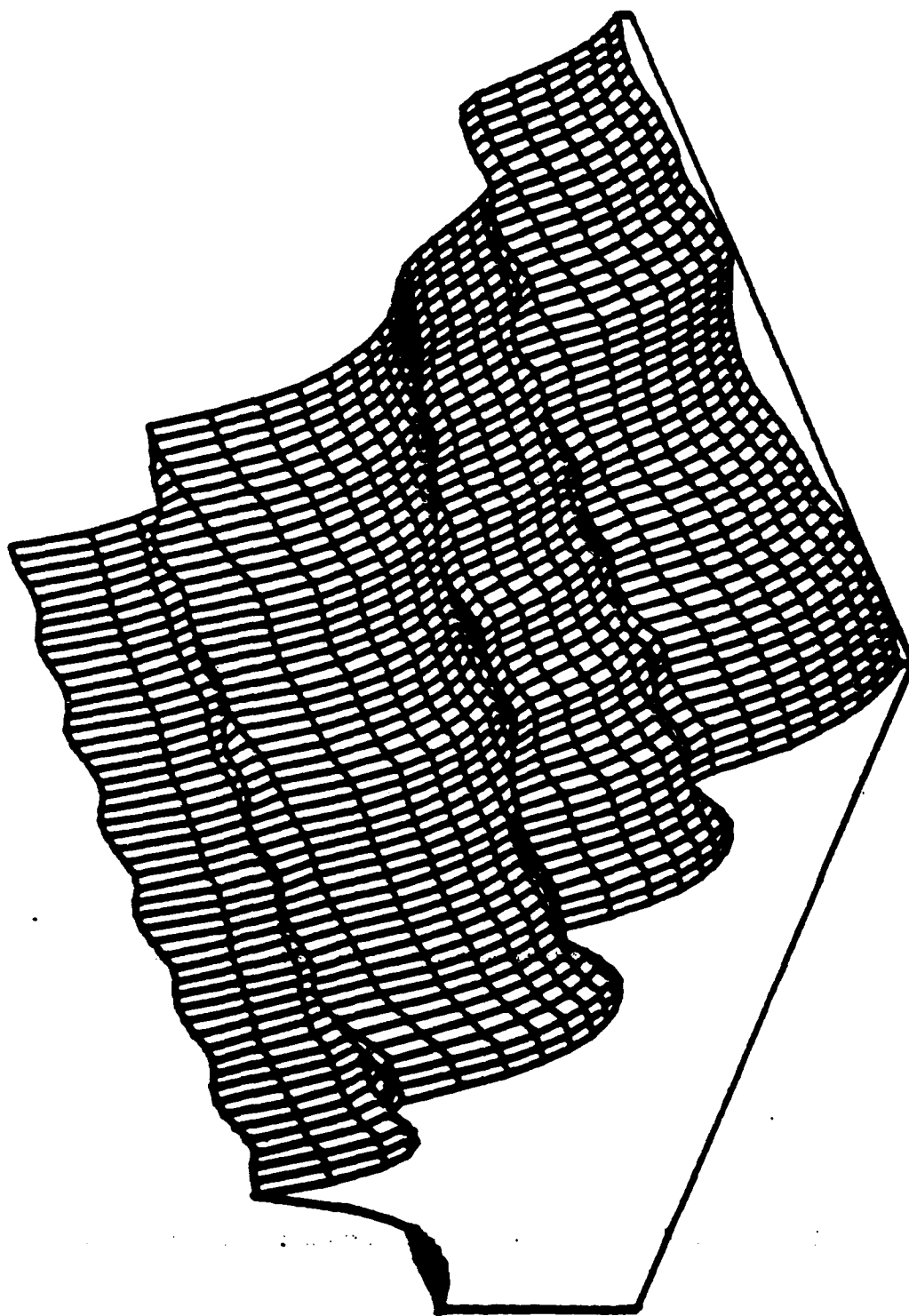


Figure IV-20.2.1.a "FIVE"

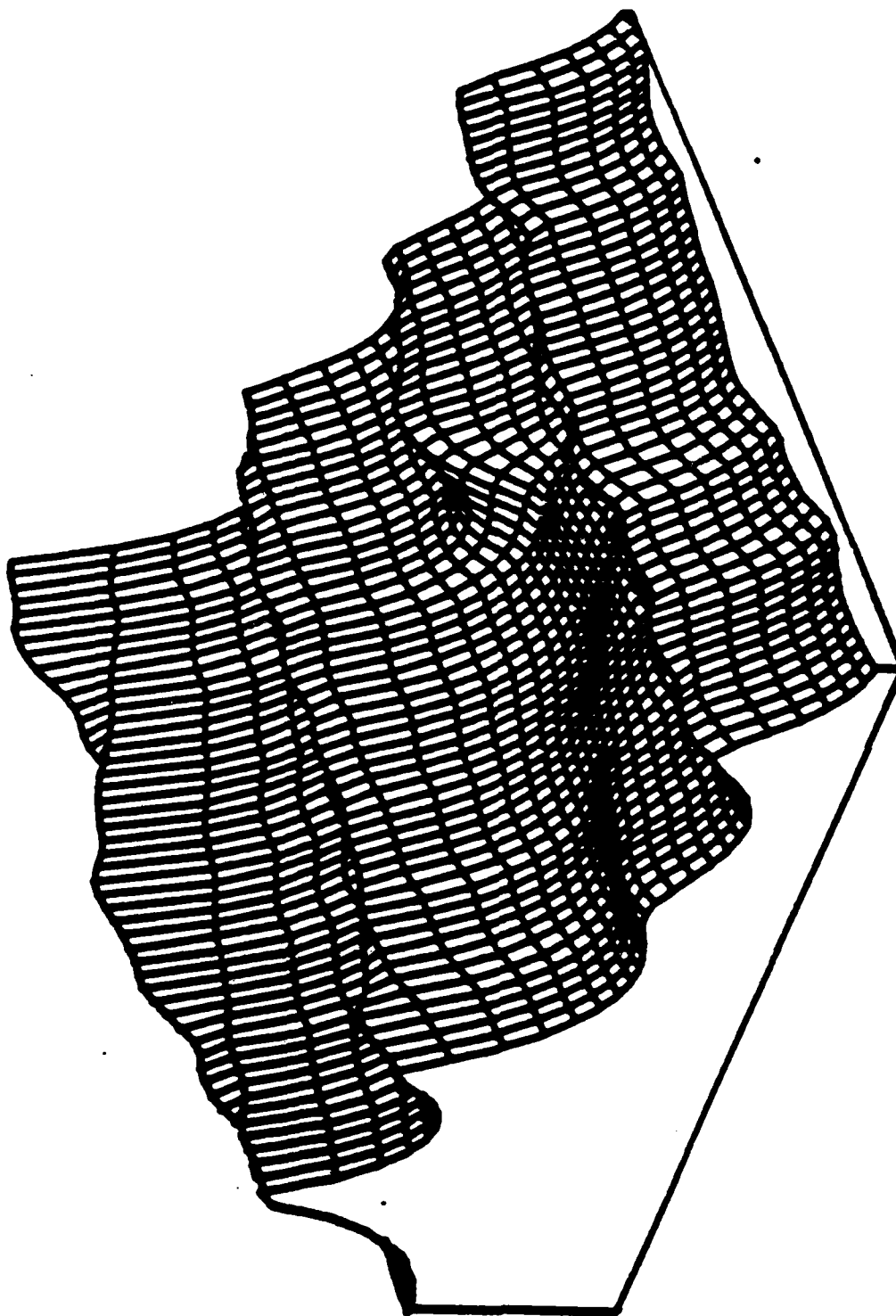


Figure IV-20.2.2.a "FIVE"

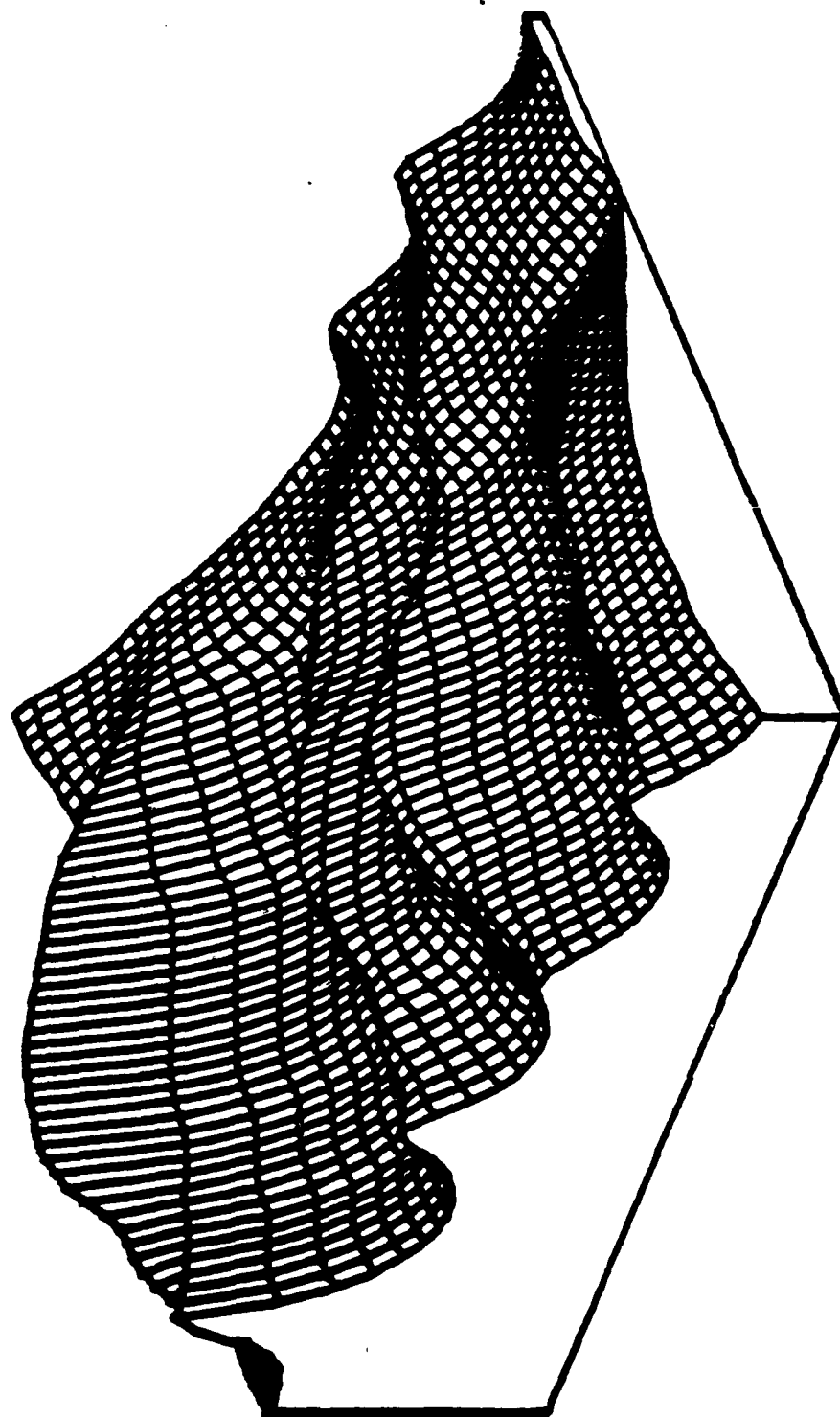


Figure IV-20.2.3.a "FIVE"

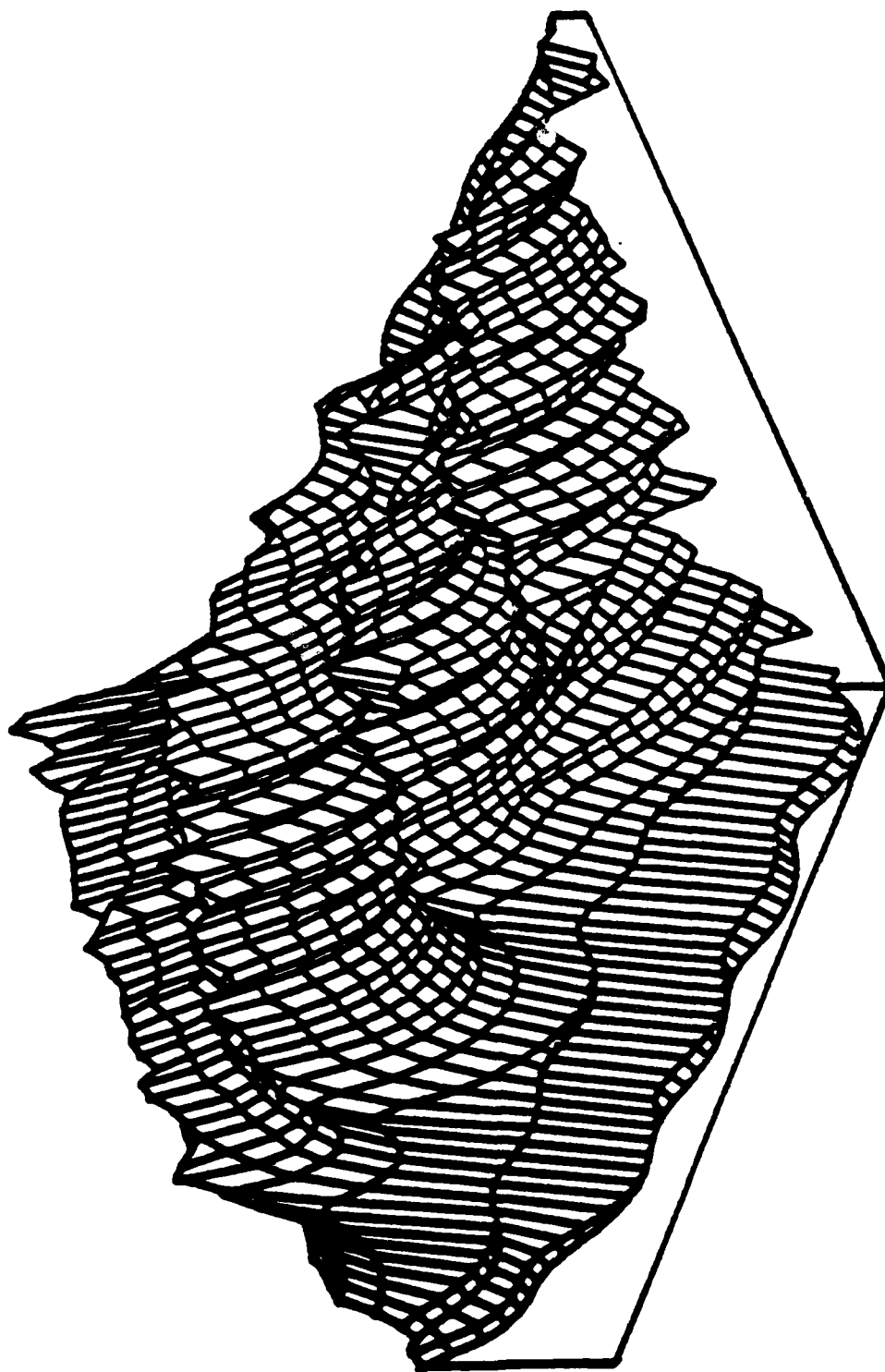


Figure IV-21.b "FIVE"

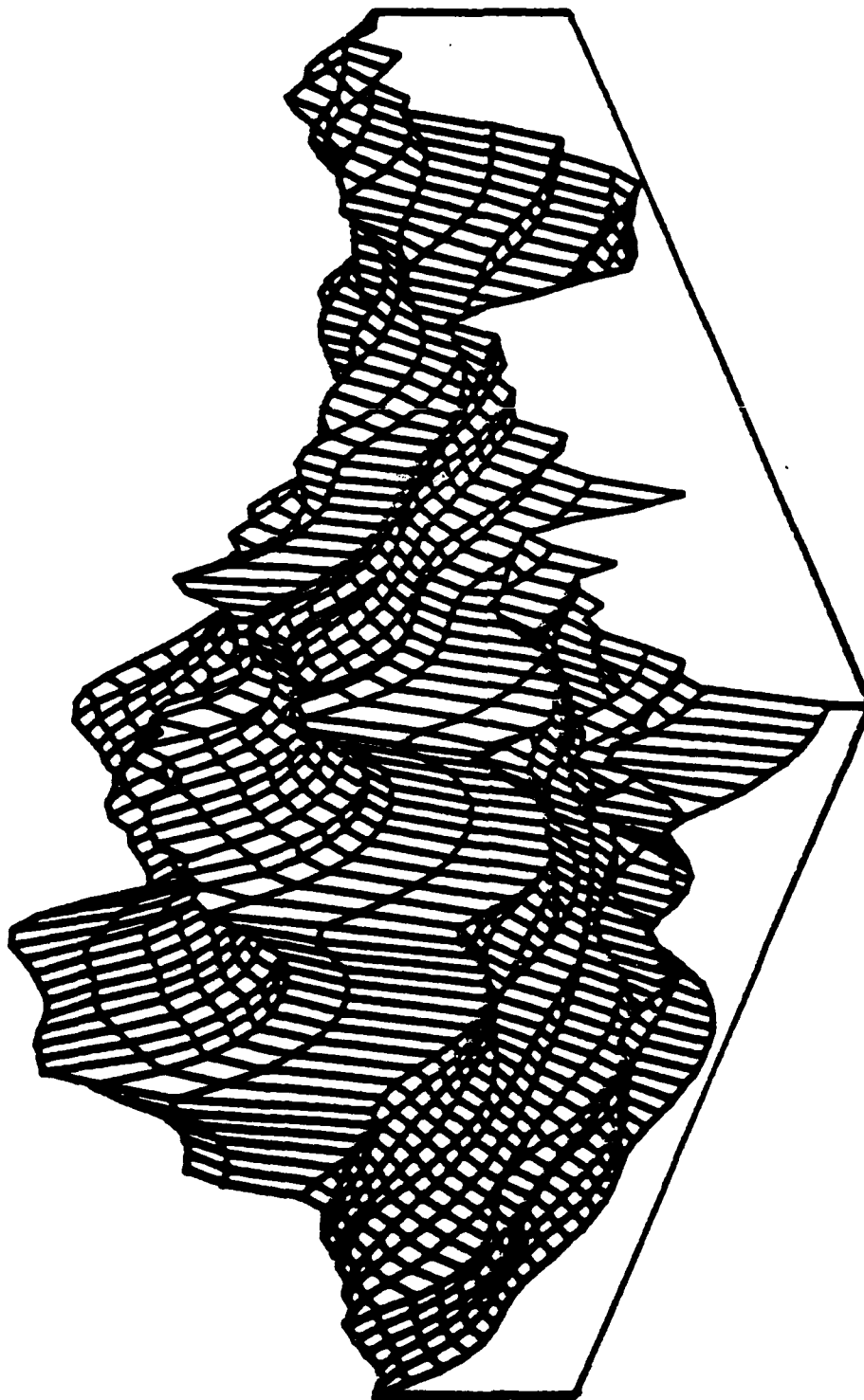


Figure IV-22.b "SIX"

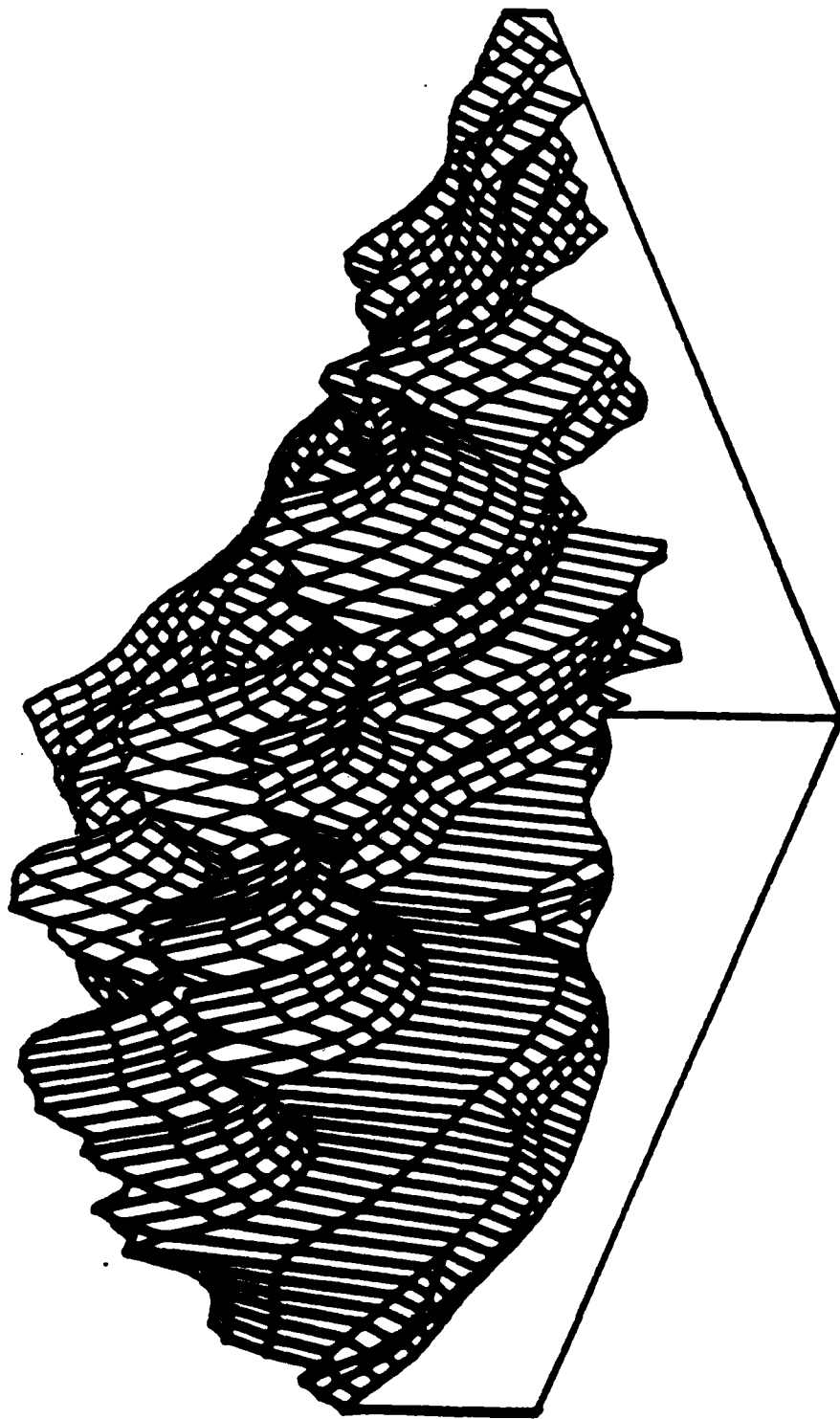


Figure IV-23.b "SEVEN"

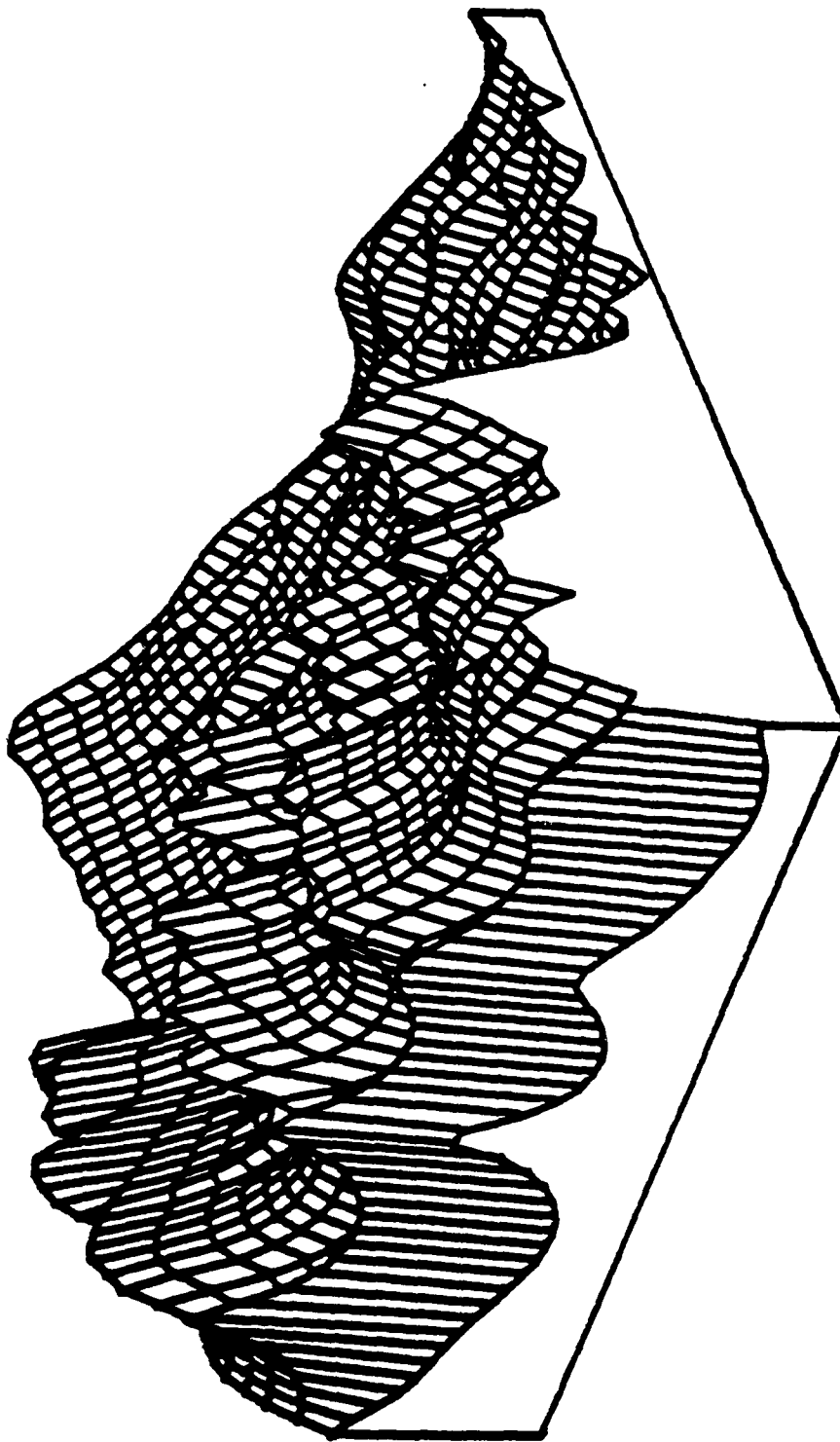
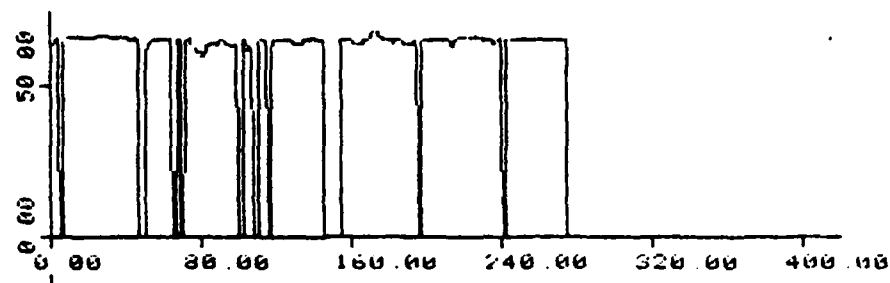
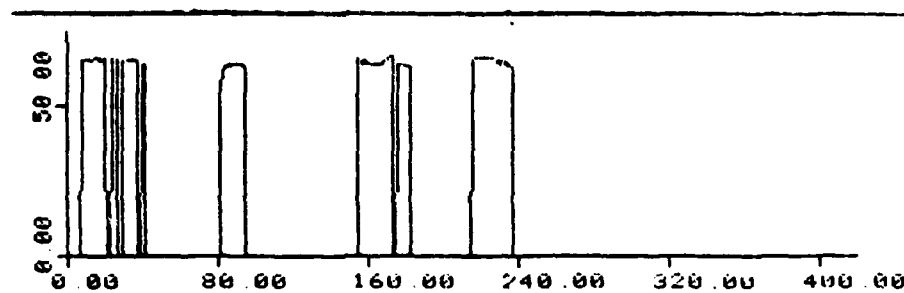


Figure IV-24.b "EIGHT"



(a) without noise



(b) with noise added

Figure IV-25 Pitch Plot of "FIVE...SIX...SEVEN...EIGHT"

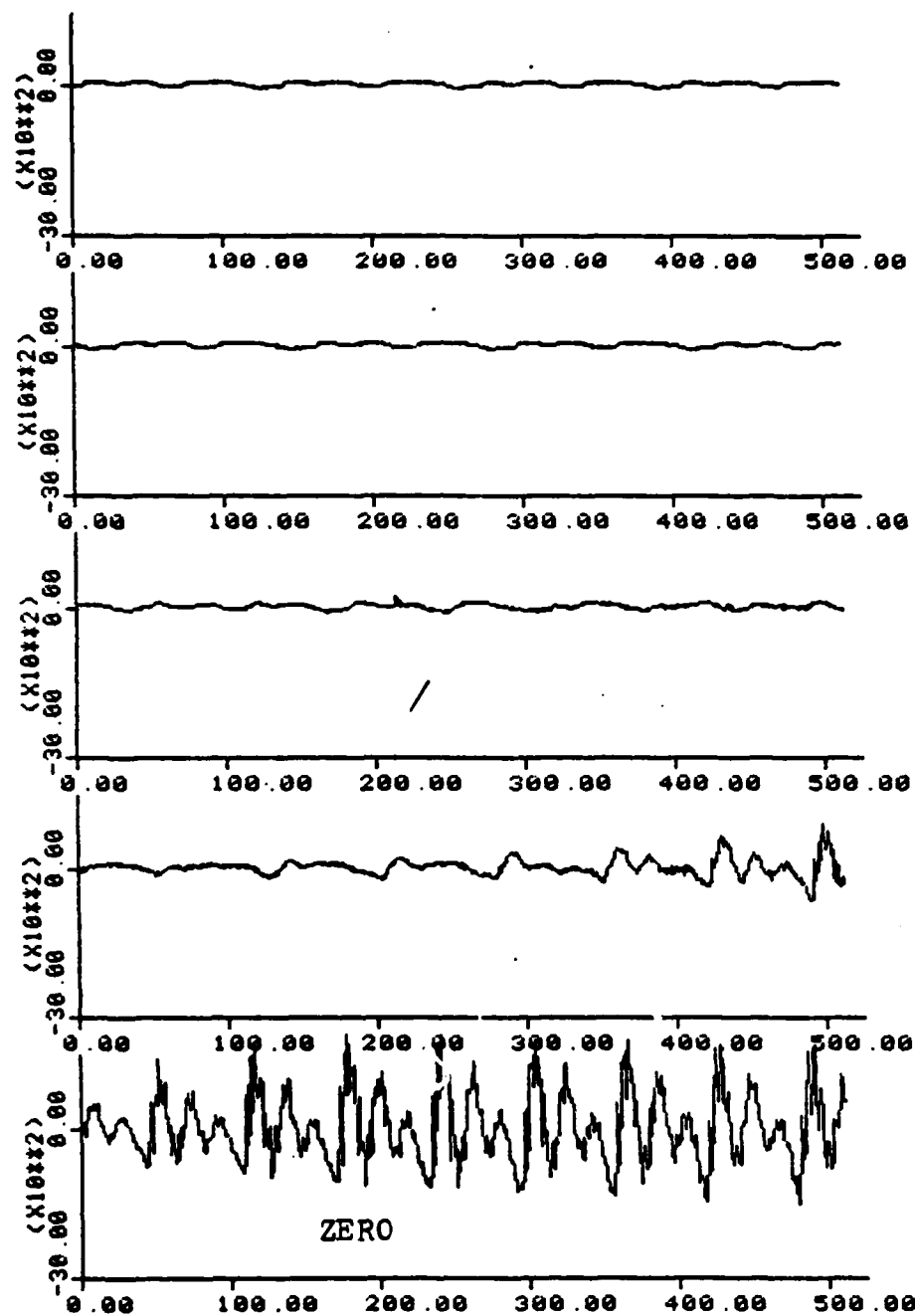


Figure IV-26.1 Original "ZERO...NINE...TEN"

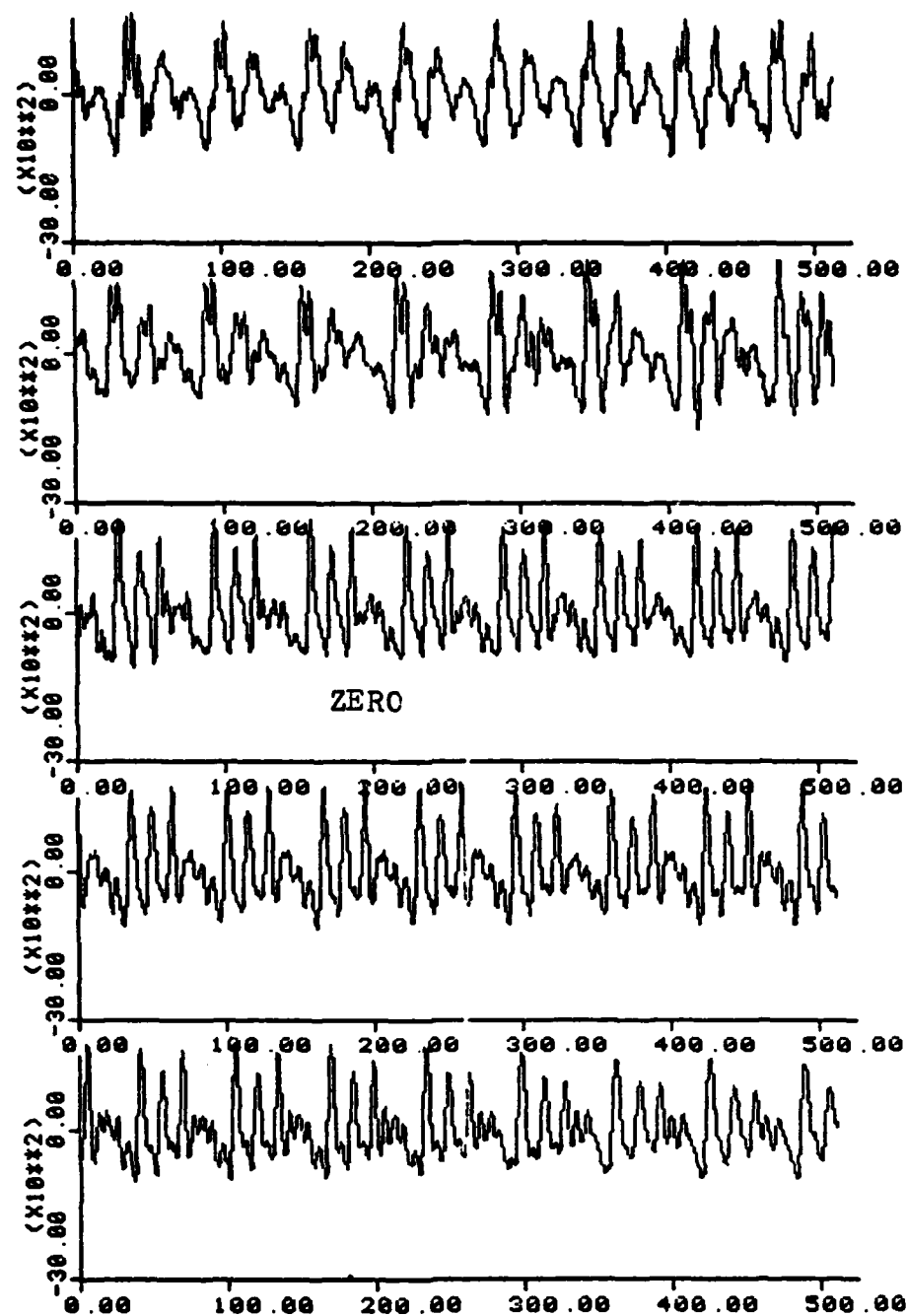


Figure IV-26.2 "ZERO...NINE...TEN"

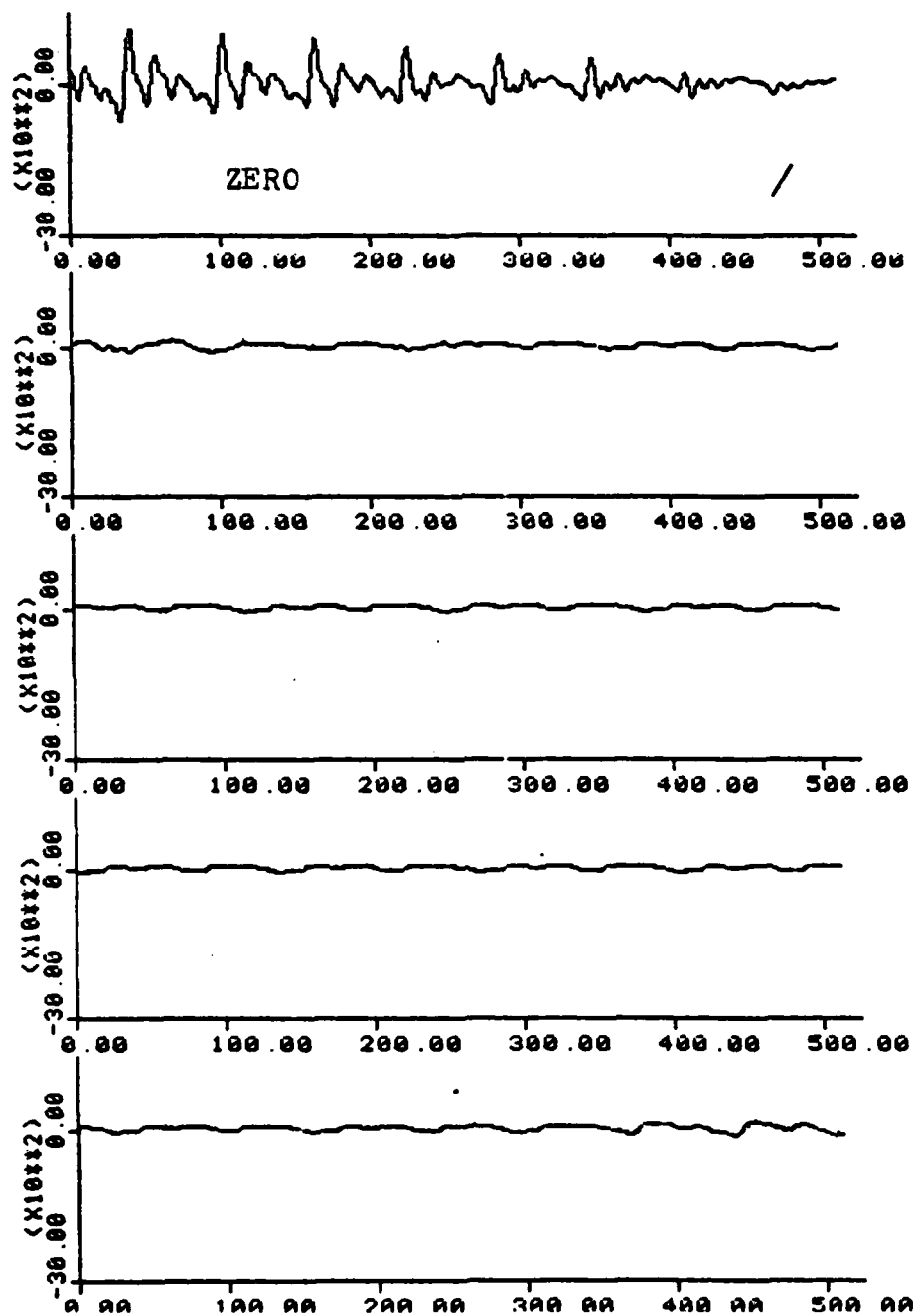


Figure IV-26.3 "ZERO...NINE...TEN"

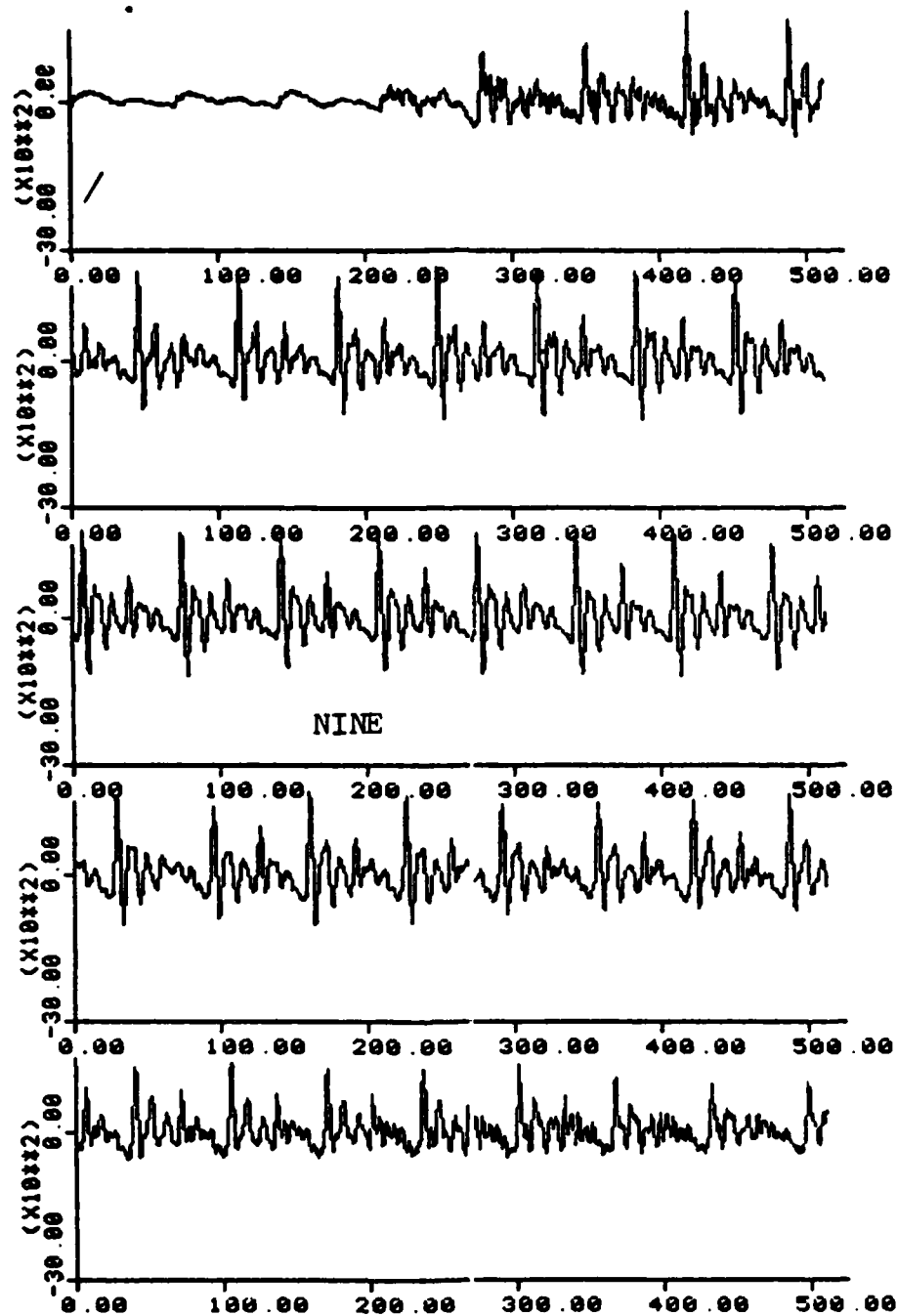


Figure IV-26.4 "ZERC...NINE...TEN"

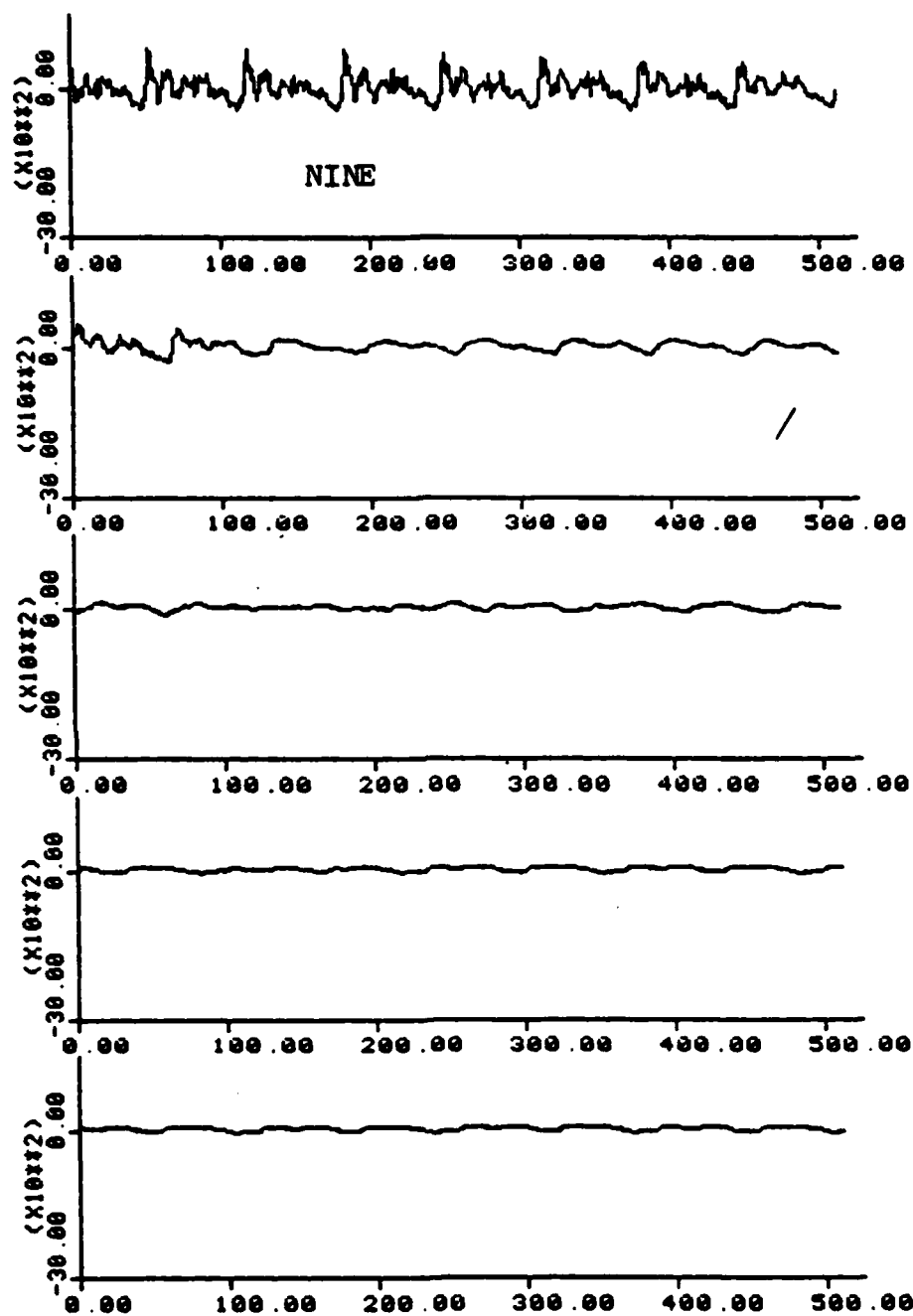


Figure IV-26.5 "ZERO...NINE...TEN"

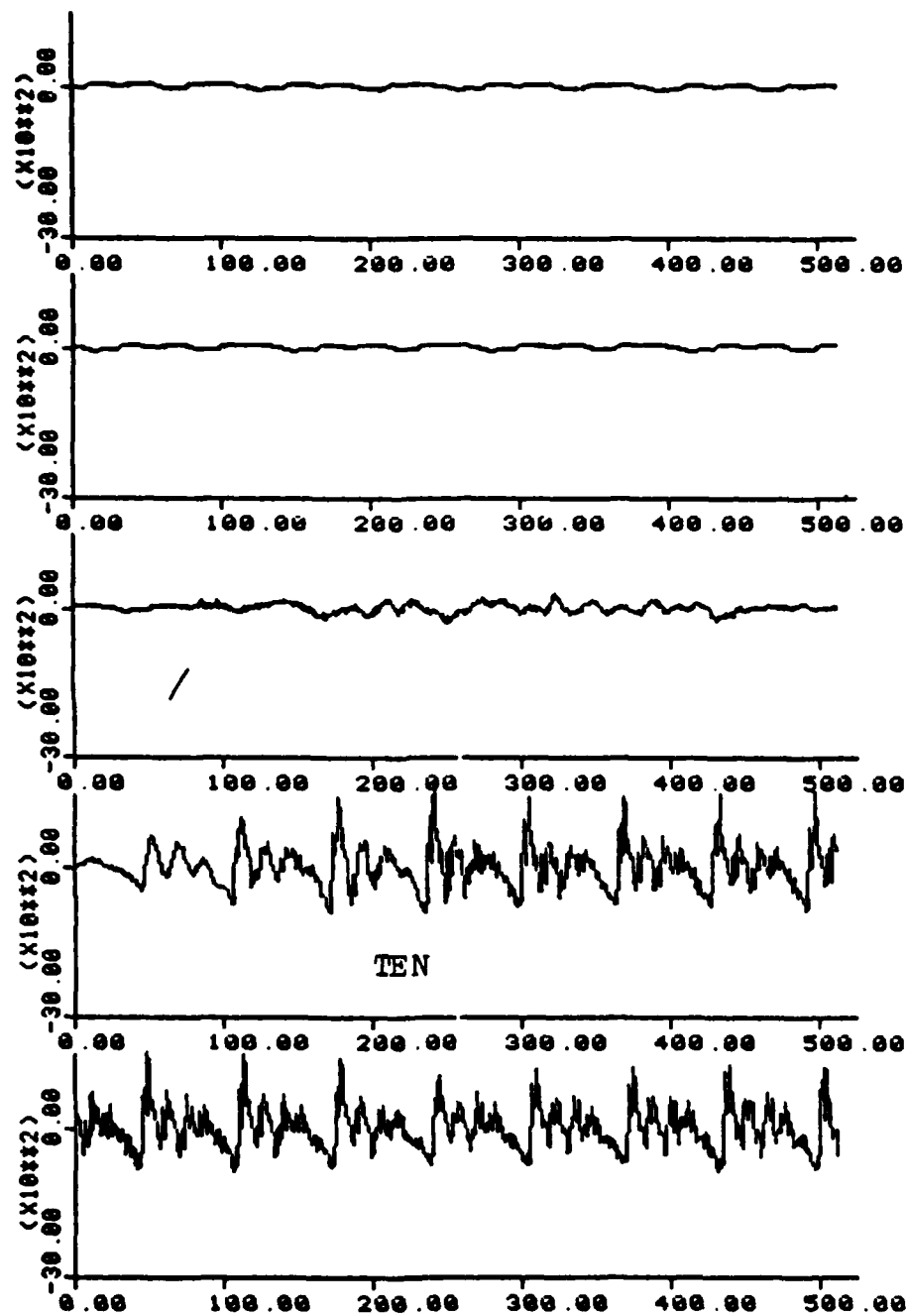


Figure IV-26.6 "ZERO...NINE...TEN"

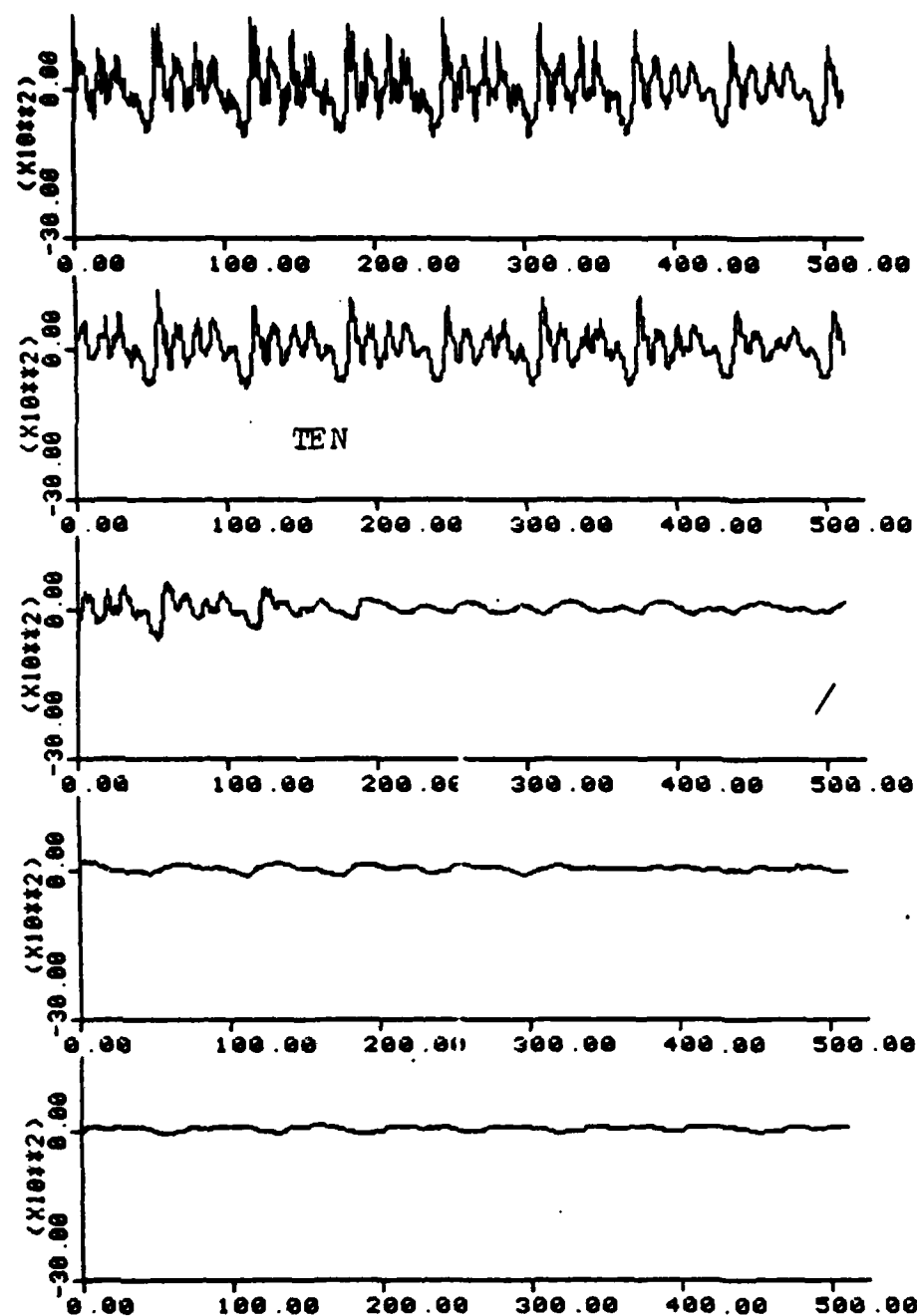


Figure IV-26.7 "ZERO...NINE...TEN"

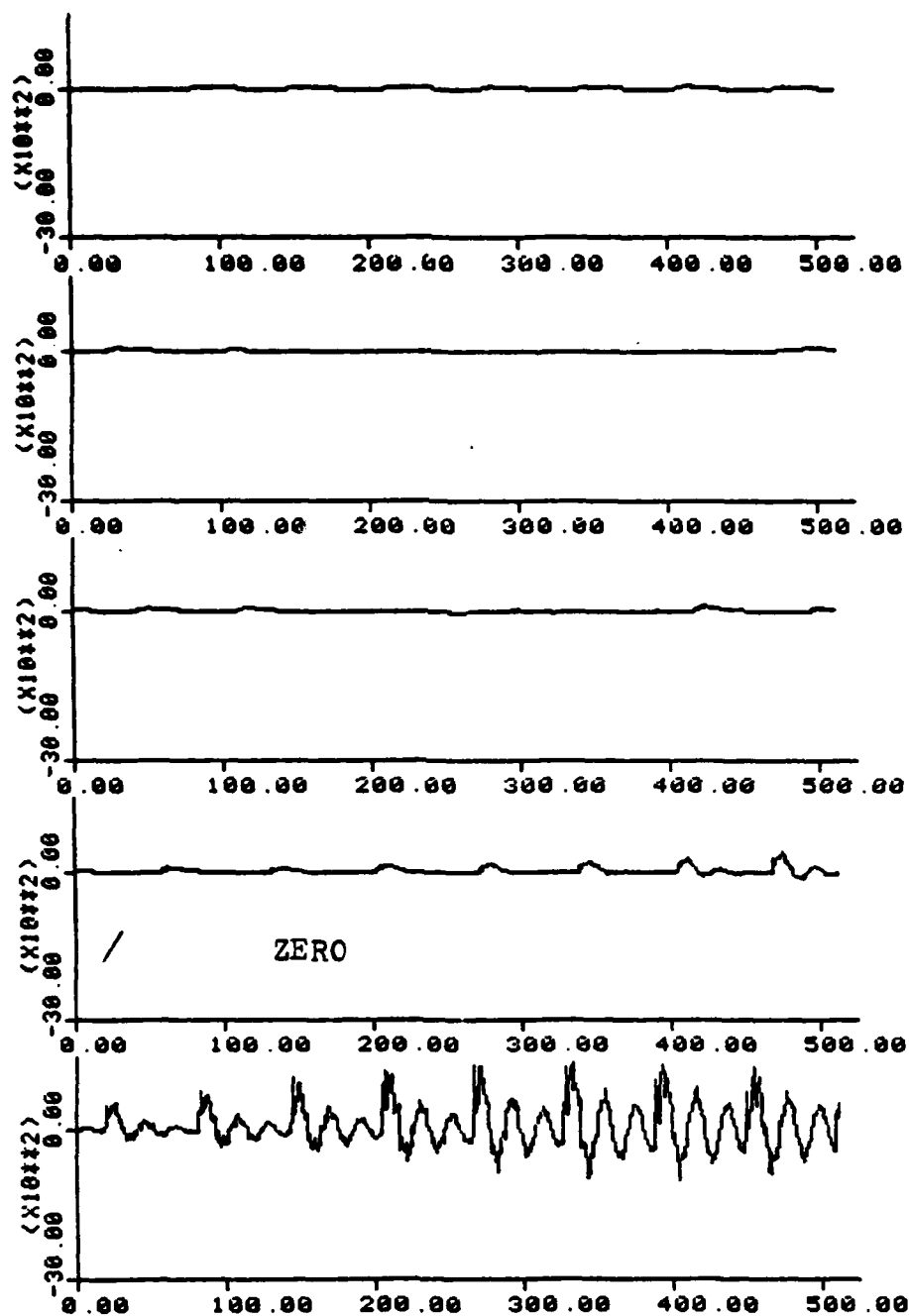


Figure IV-27.1 Vocoder Output of
"ZERO...NINE...TEN".

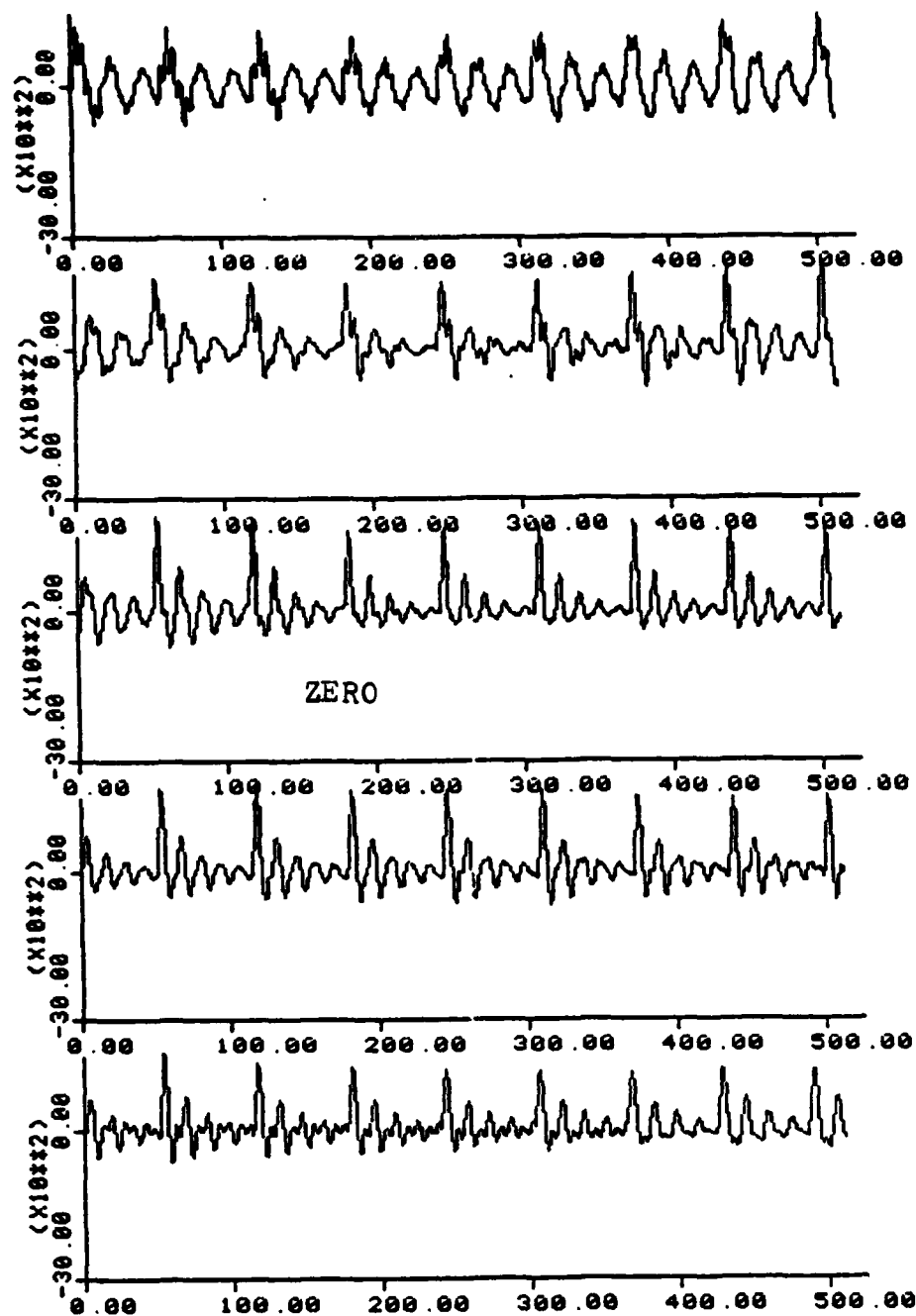


Figure IV-27.2 "ZERO...NINE...TEN"

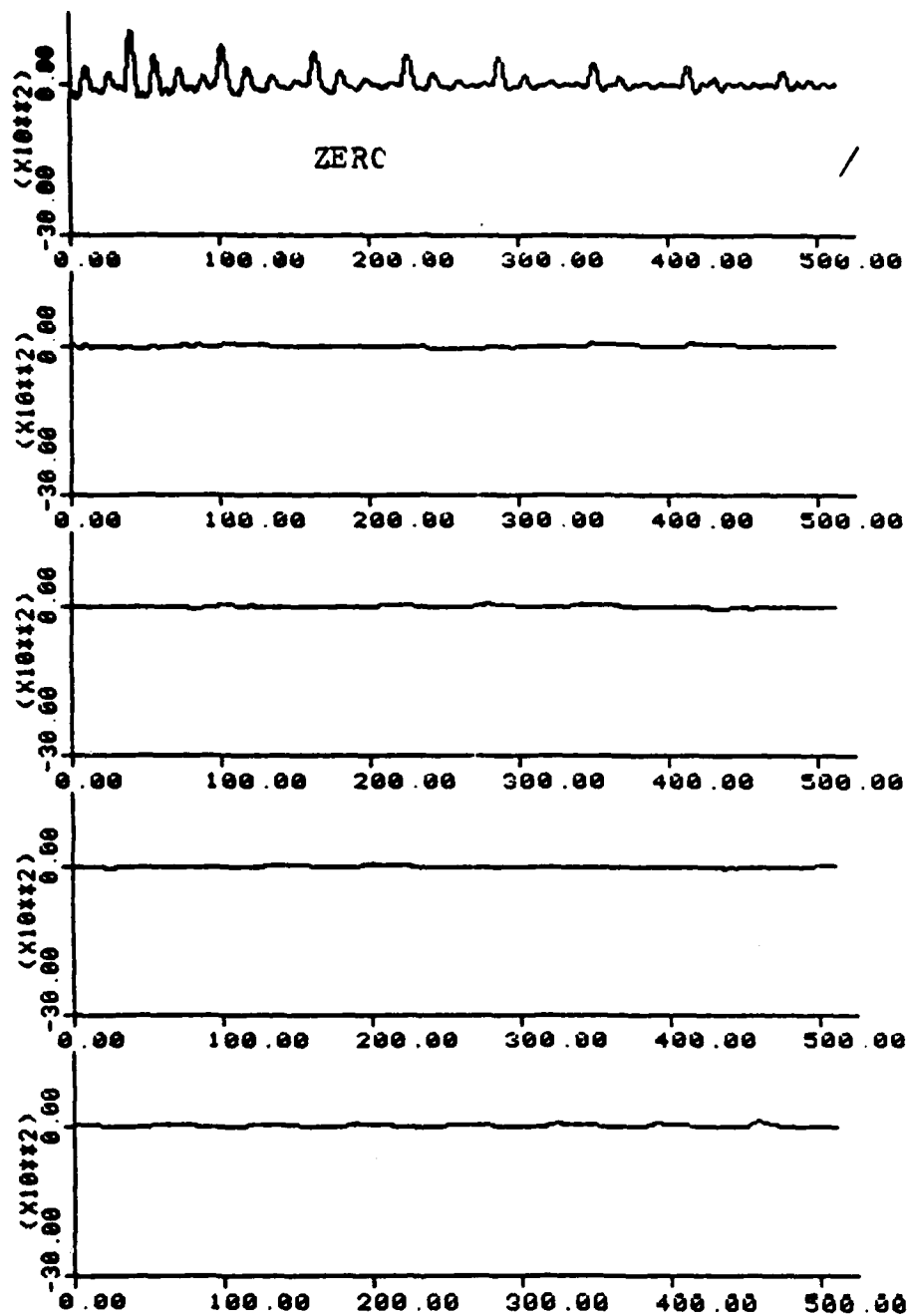


Figure IV-27.3 "ZERC...NINE...TEN"

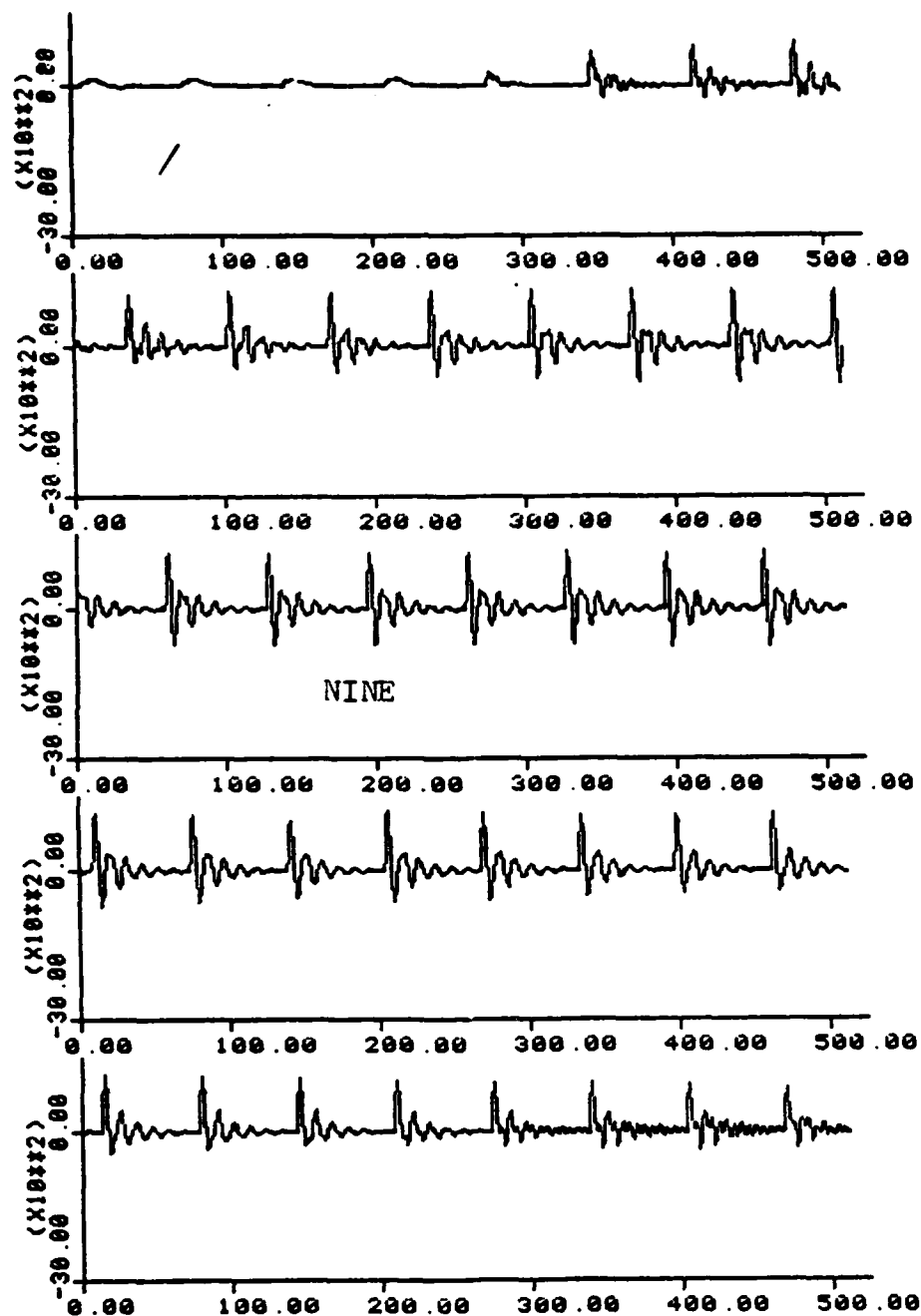


Figure IV-27.4 "ZERC...NINE...TEN"

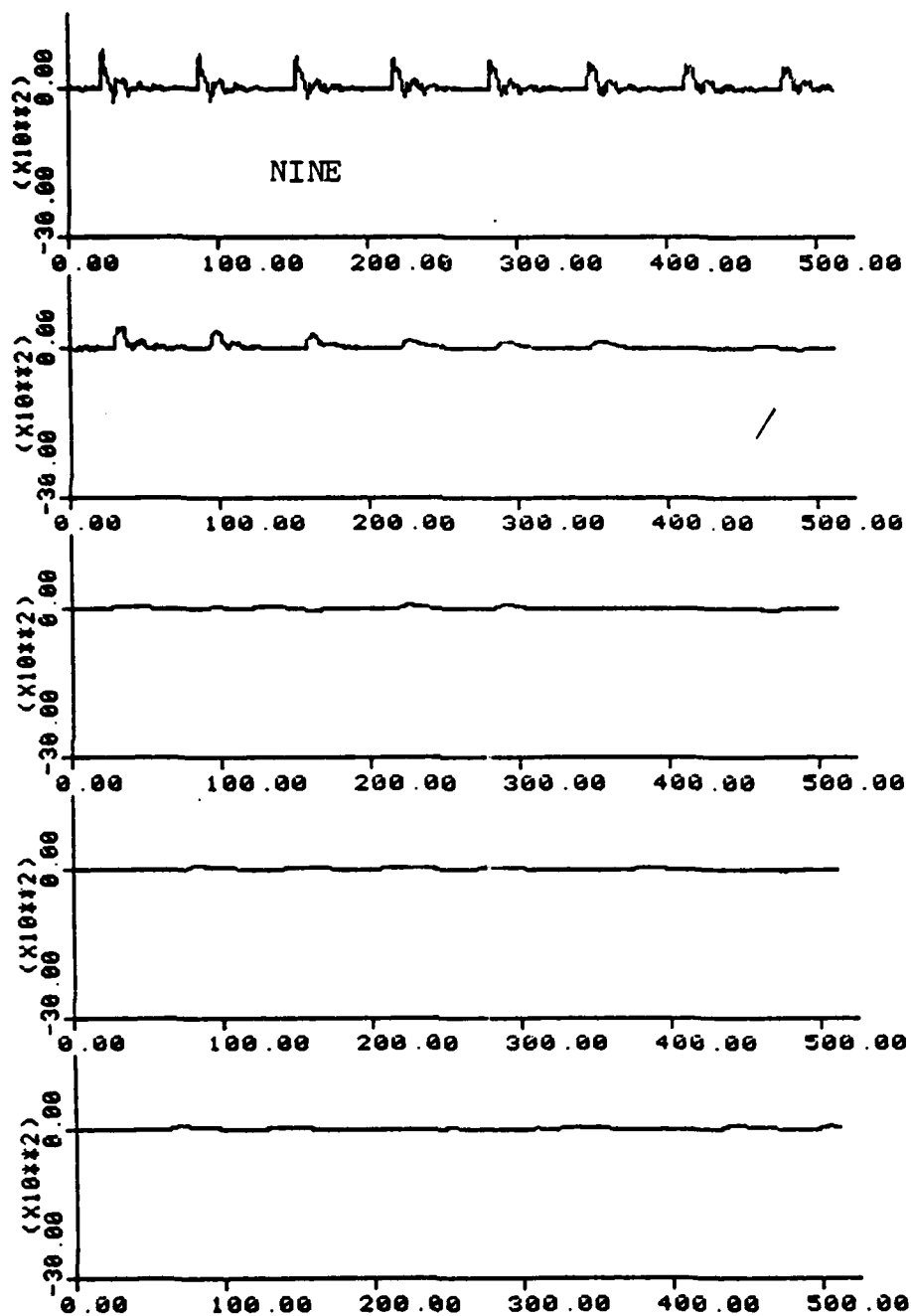


Figure IV-27.5 "ZERC...NINE...TEN"

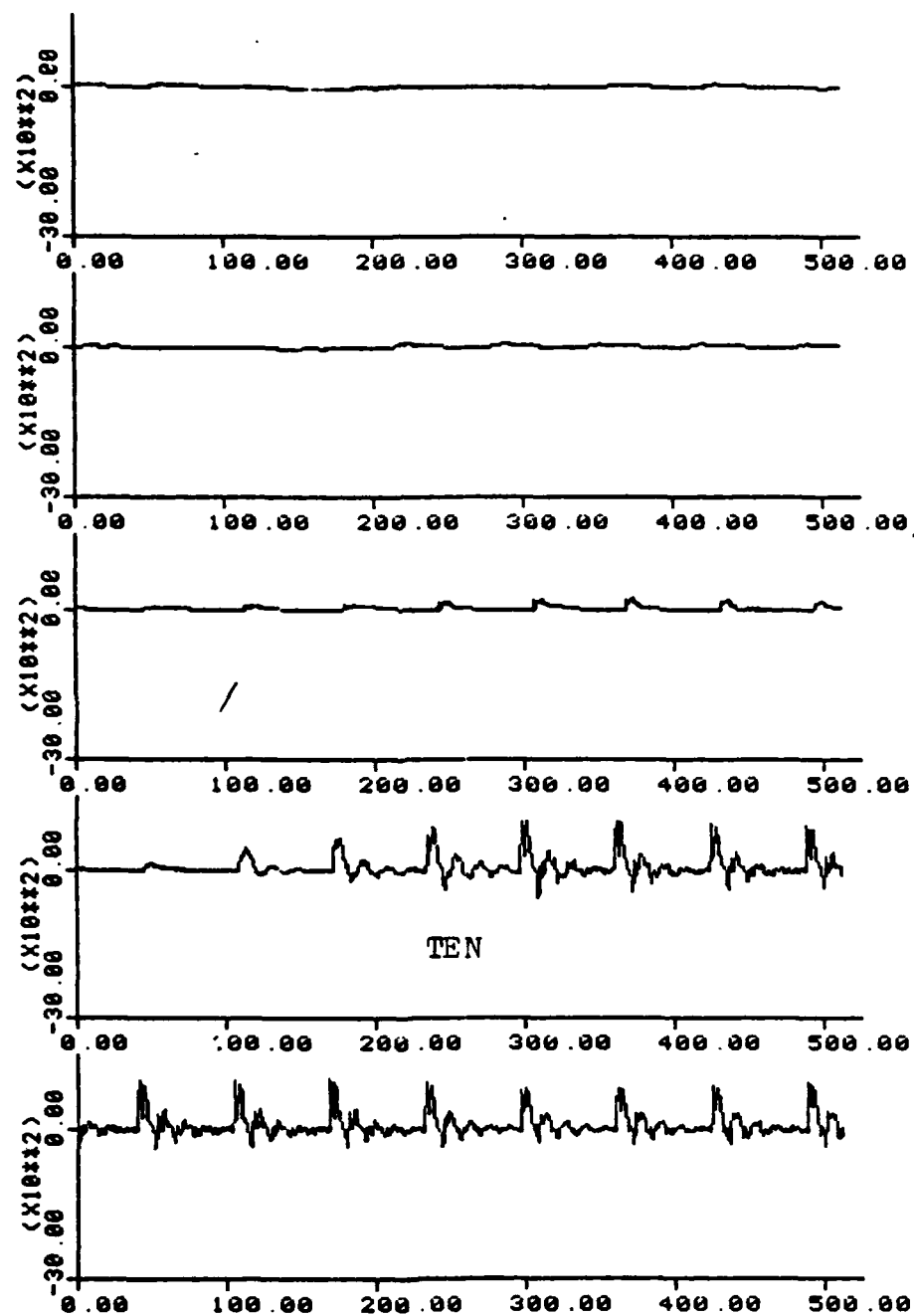


Figure IV-27.6 "ZERC...NINE...TEN"

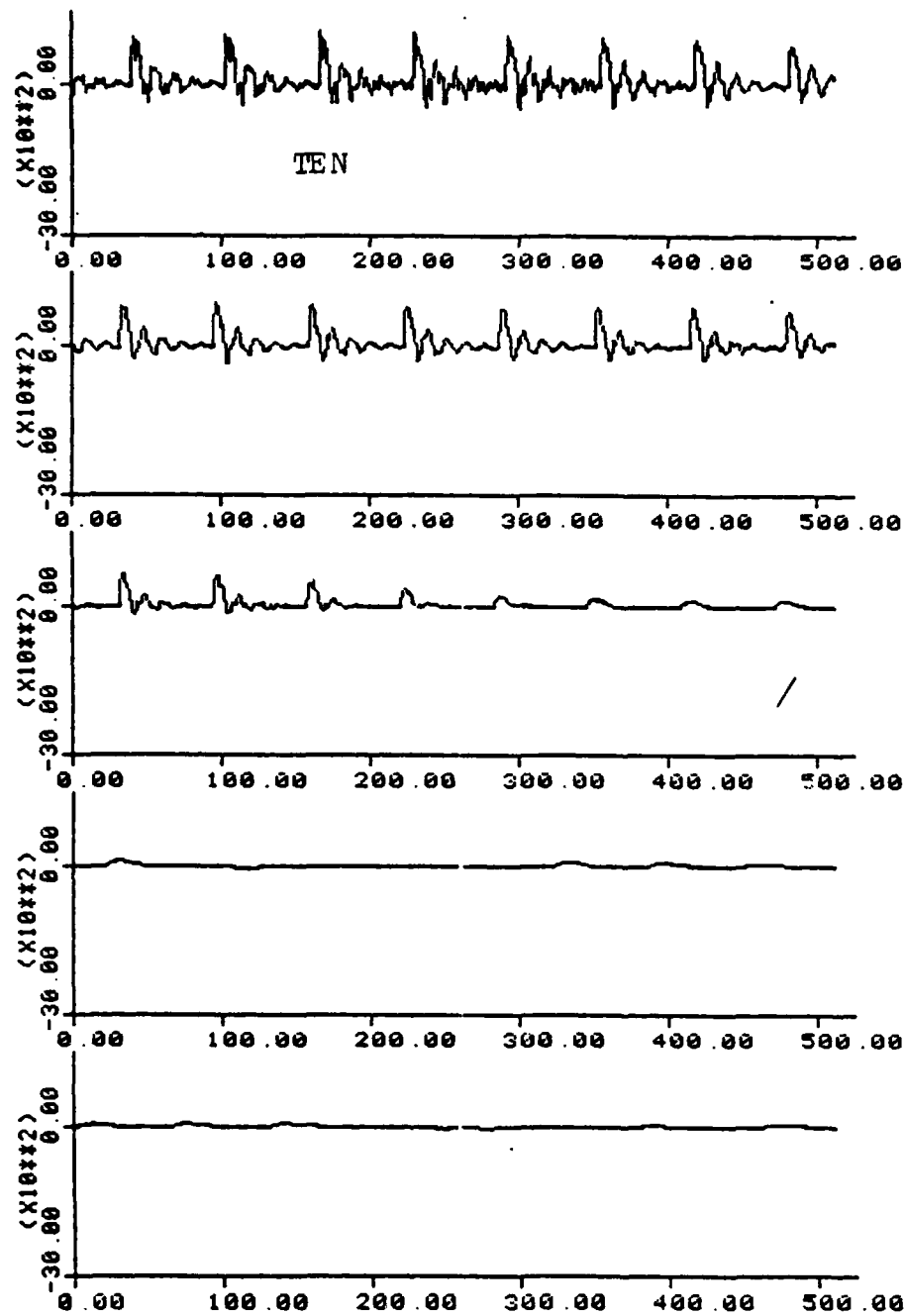


Figure IV-27.7 "ZERC...NINE...TEN"

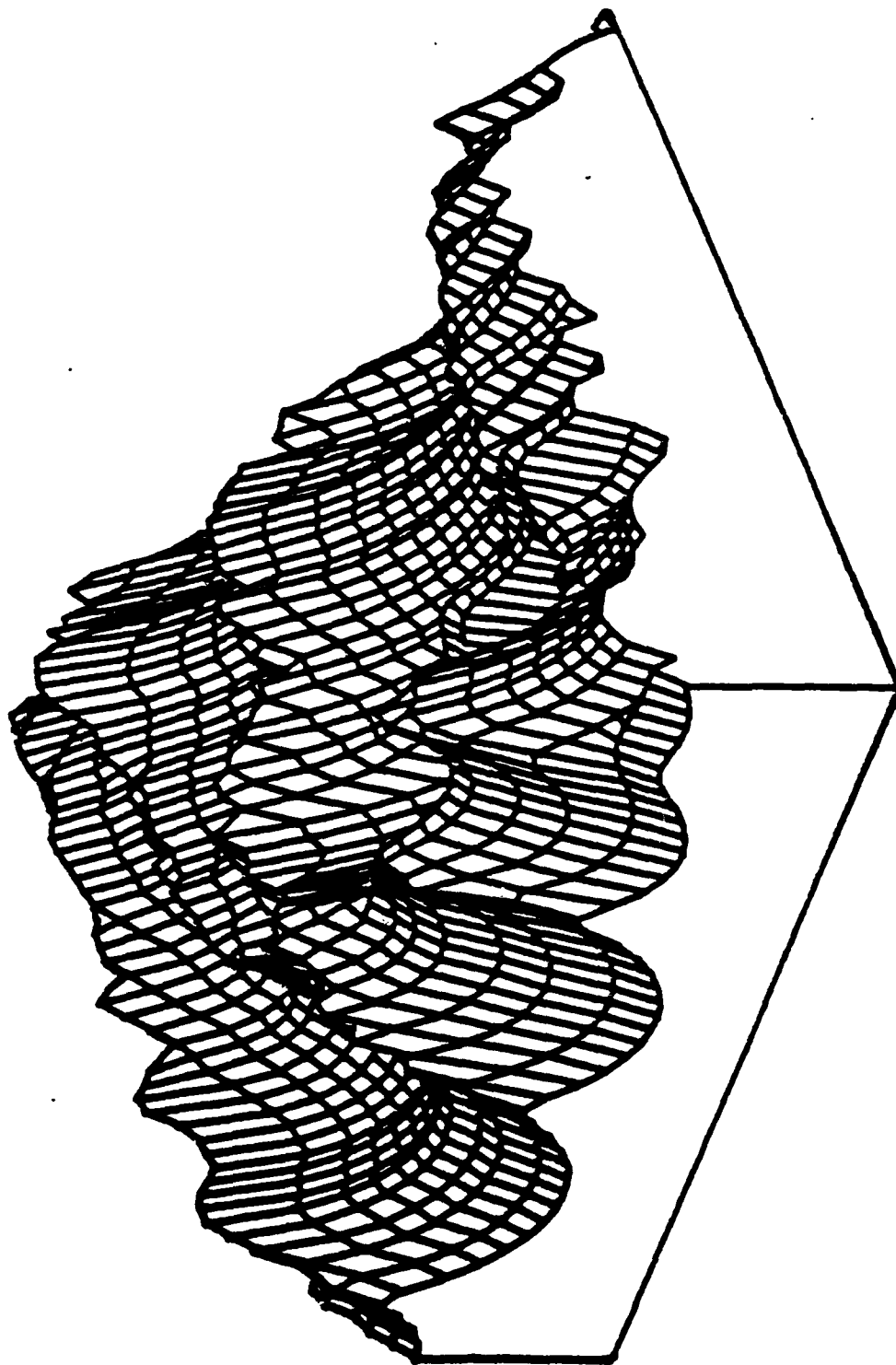


Figure IV-28.b "ZERO"

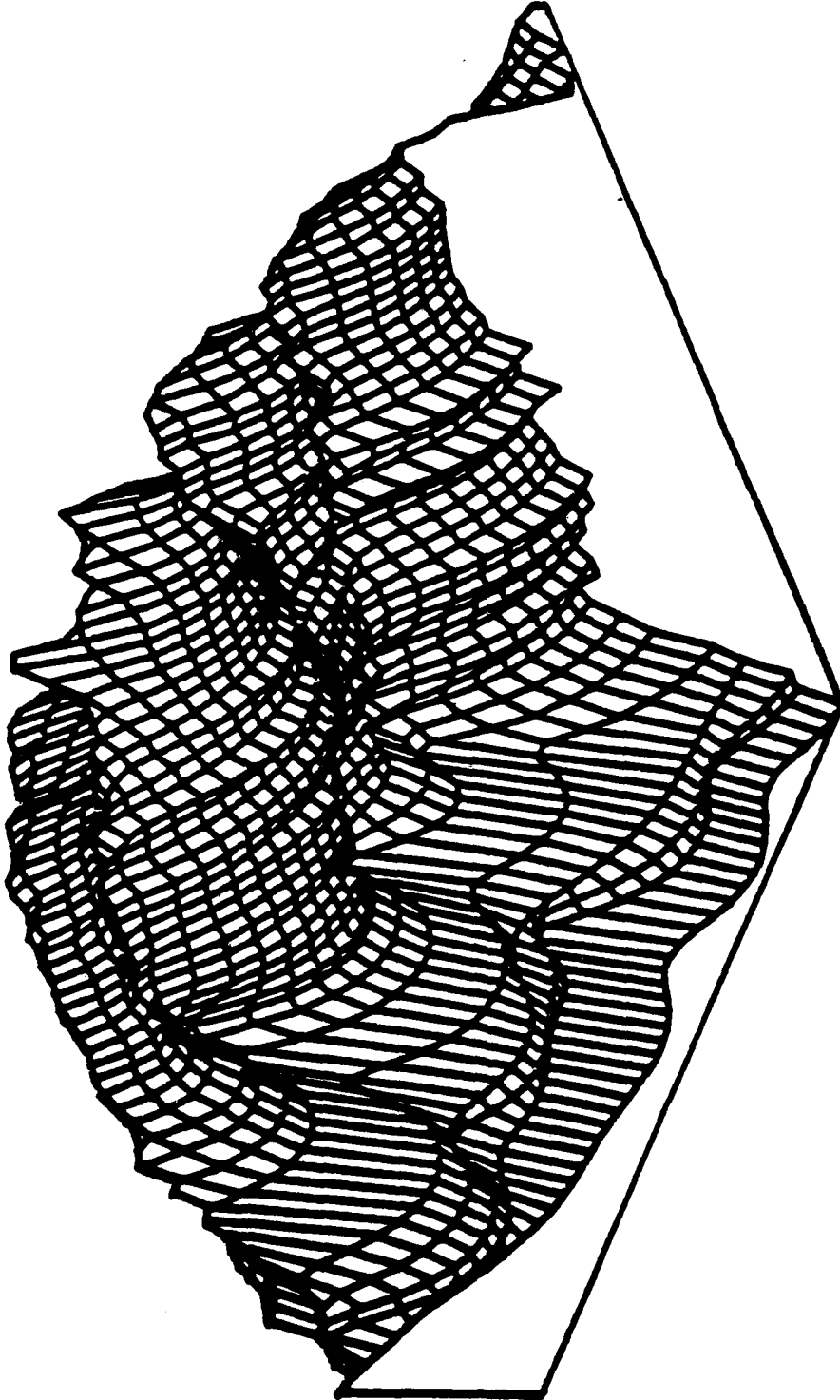


Figure IV-29.b "NINE"

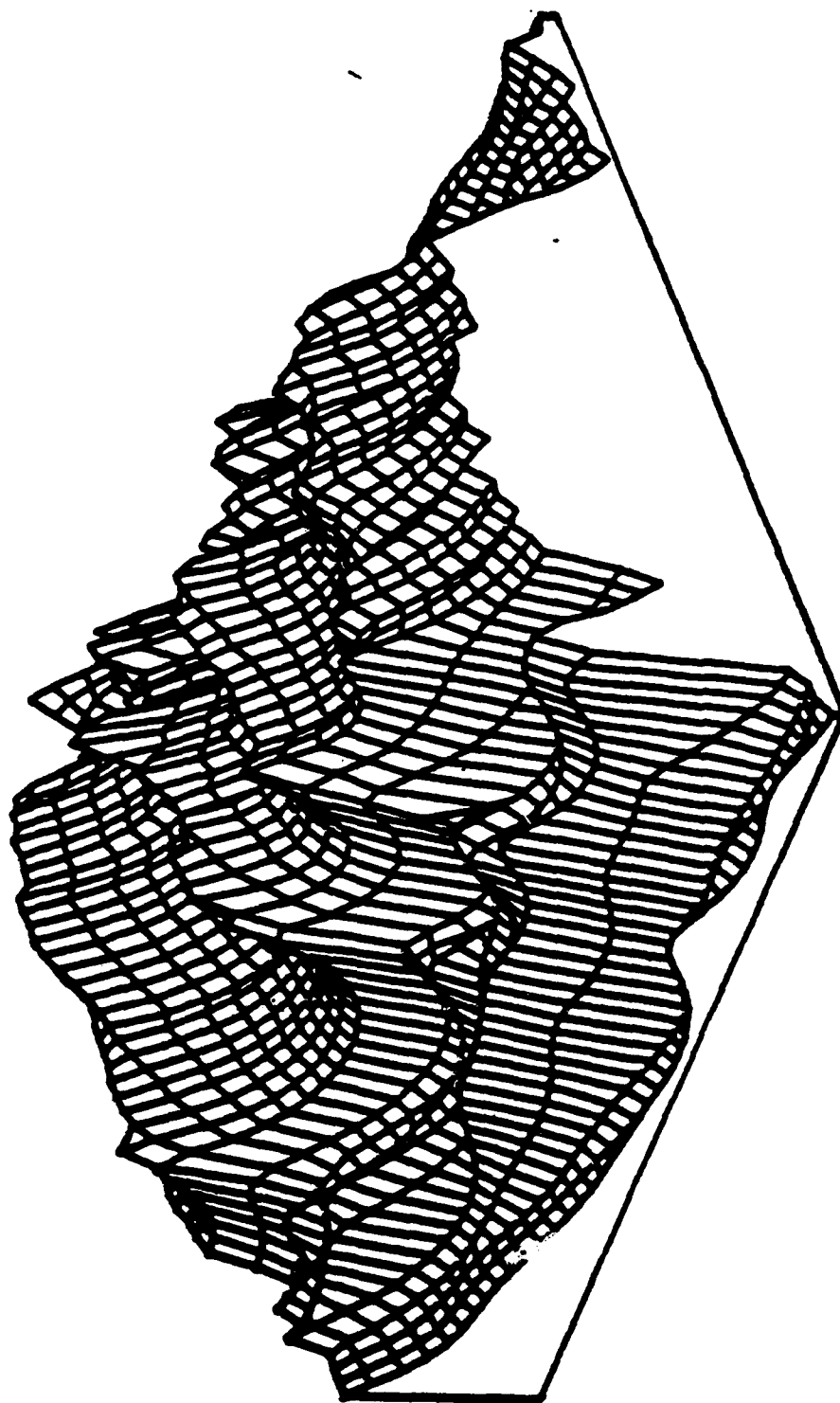
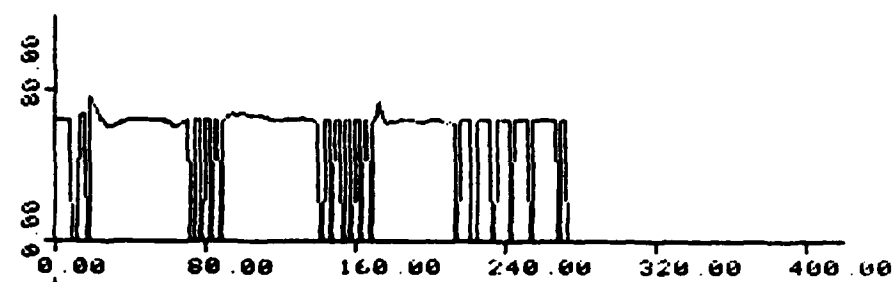
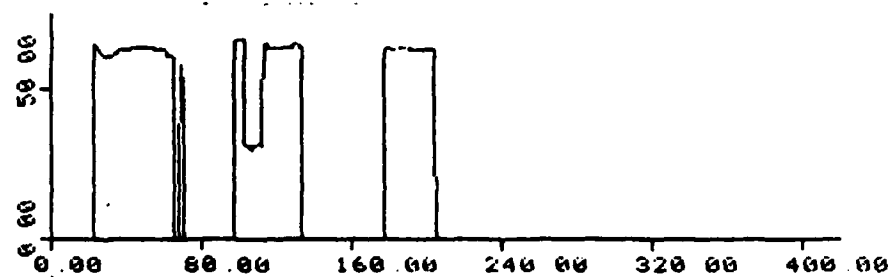


Figure IV-30.b "TEN".



(a) without noise



(b) with noise added

Figure IV-31 Pitch plot of "ZERO...NINE...TEN"
Using SIFT Pitch Detector

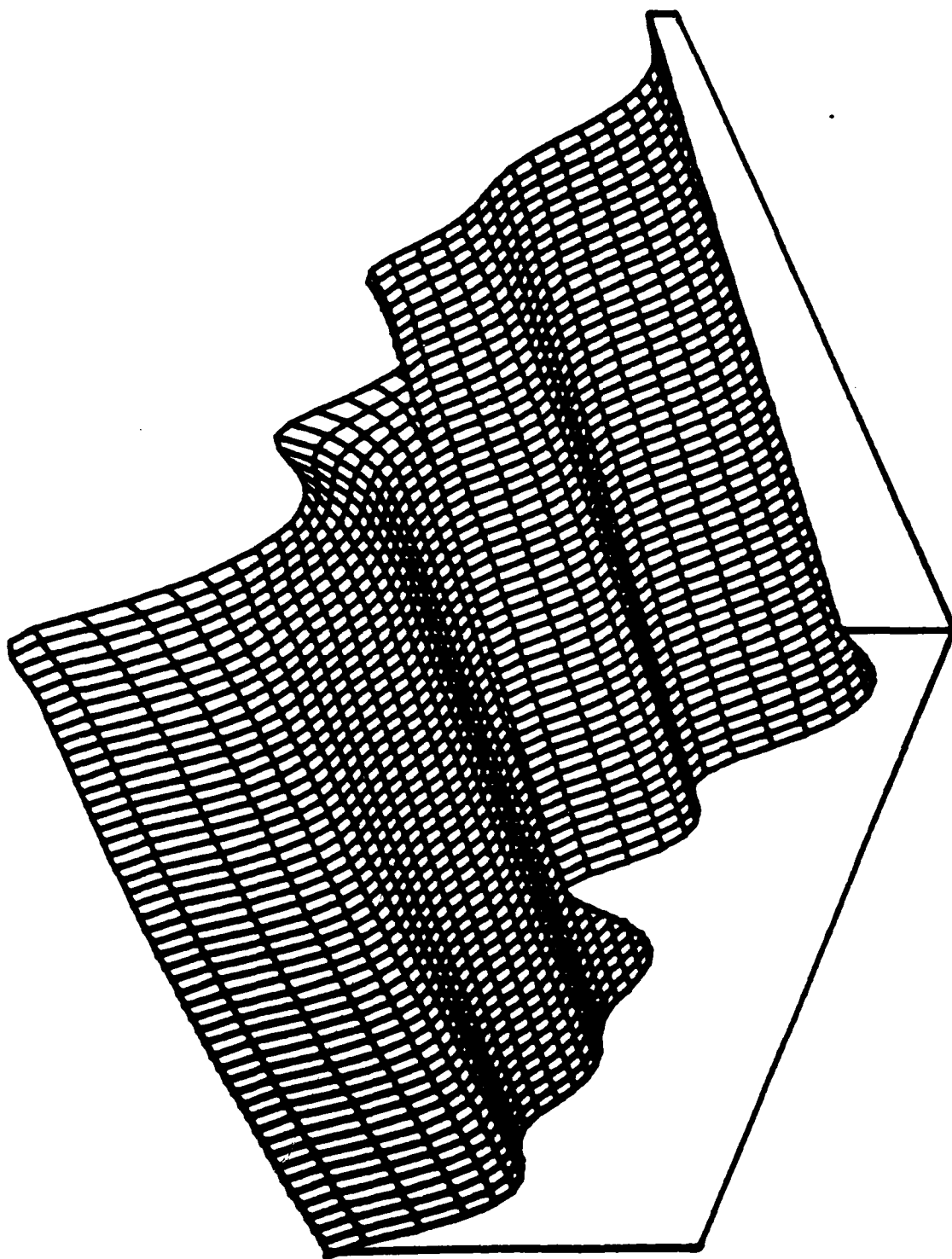


Figure IV-32.1.2-138 The beginning of the word "FIVE"
(.125 msec between formant plots for all of figure IV-32)

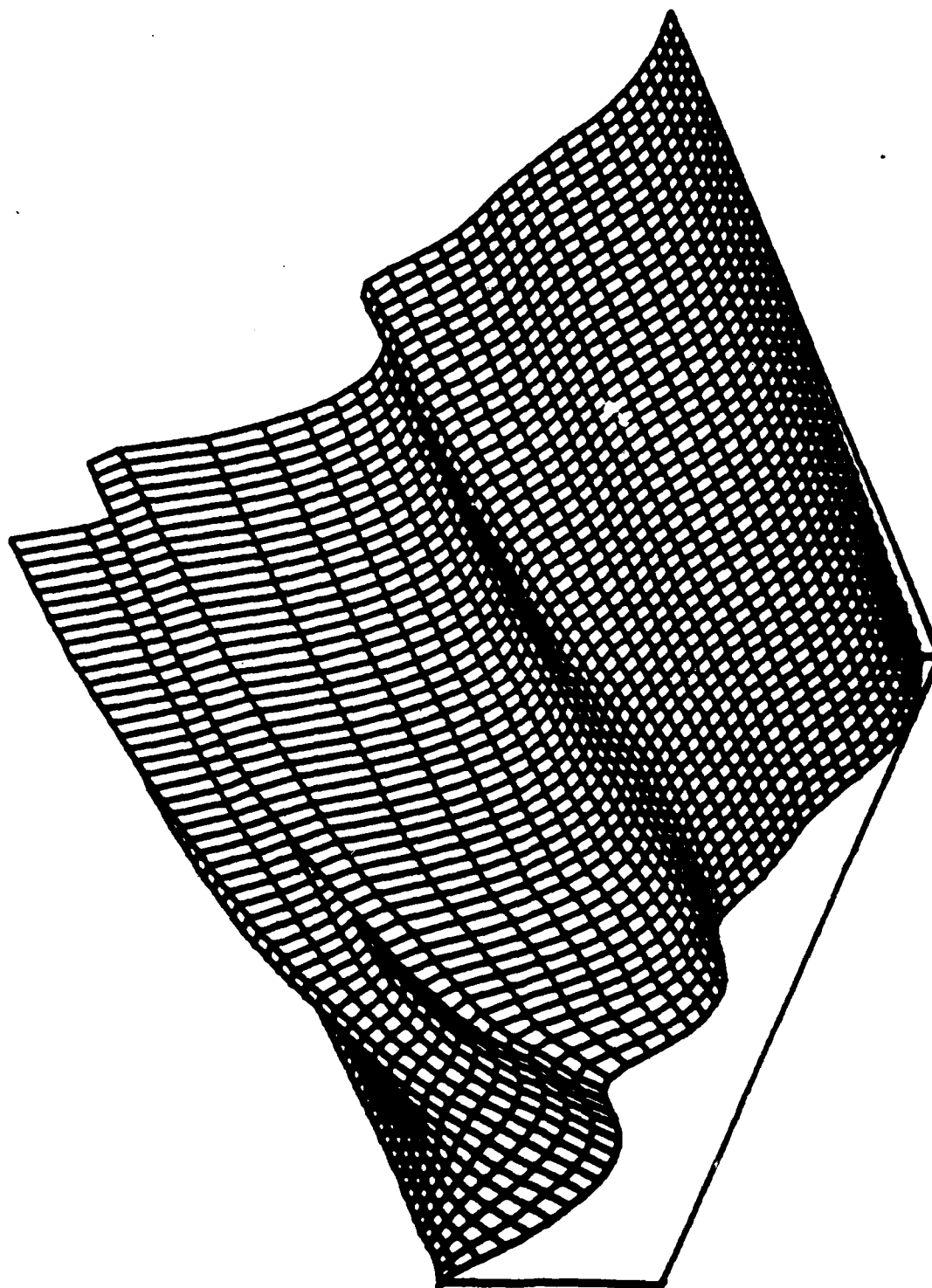


Figure IV-32.1.2-188

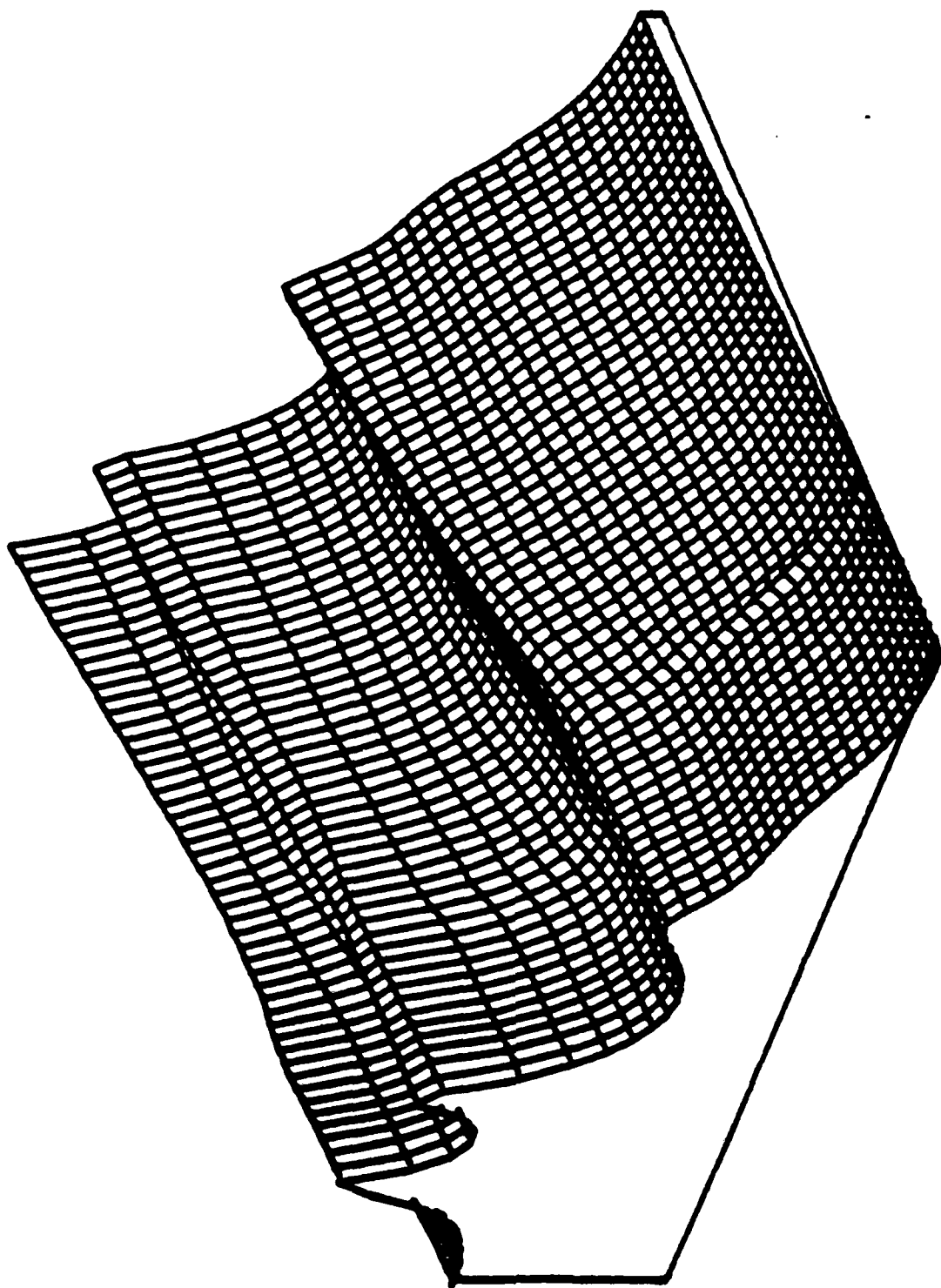


Figure IV-32.1.2-238

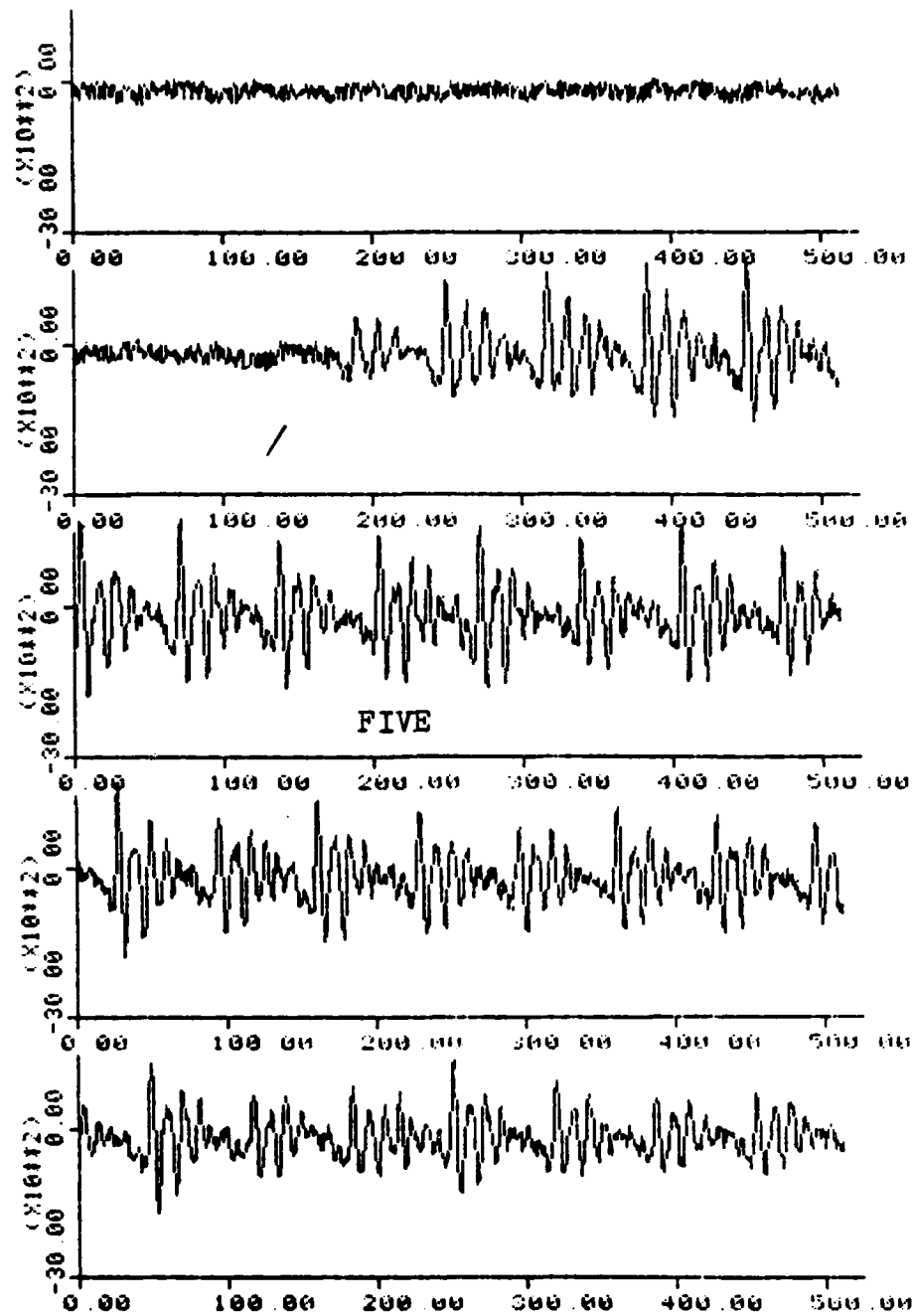


Figure IV-33.1 Original Speech File
"FIVE...SIX...SEVEN...EIGHT" + noise

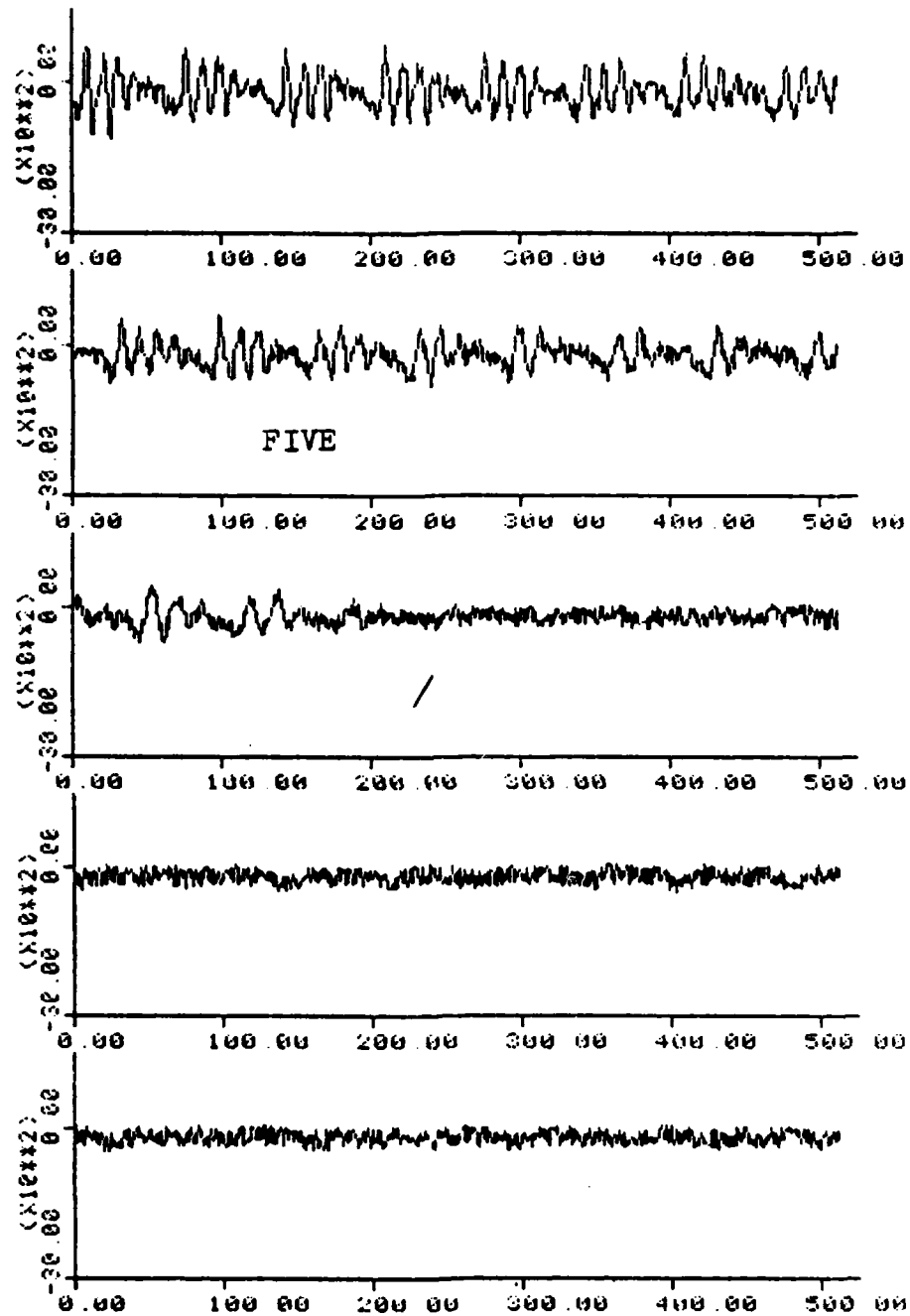


Figure IV-33.2 "FIVE...SIX...SEVEN...EIGHT" + noise

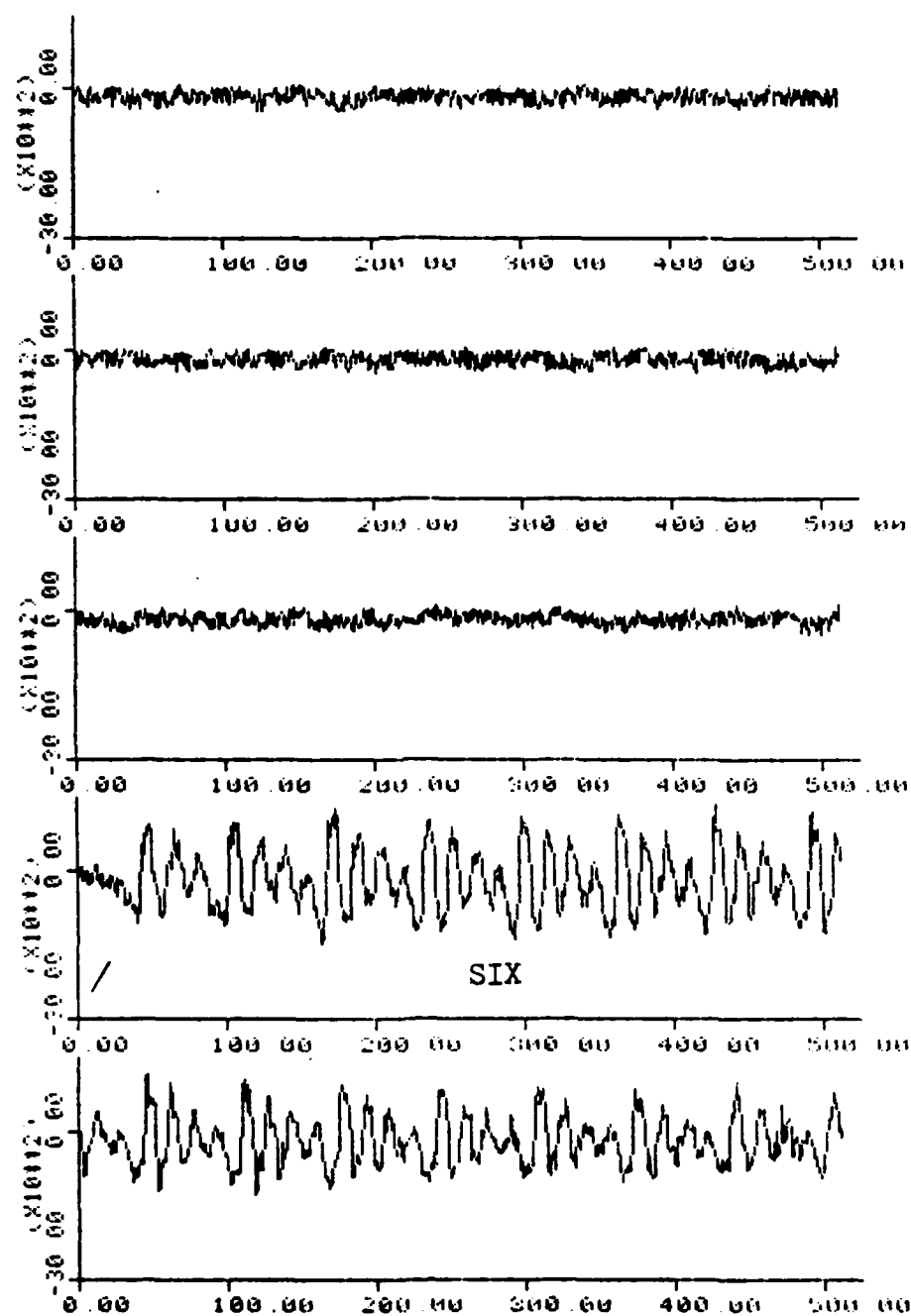


Figure IV-33.3 "FIVE...SIX...SEVEN...EIGHT" + noise

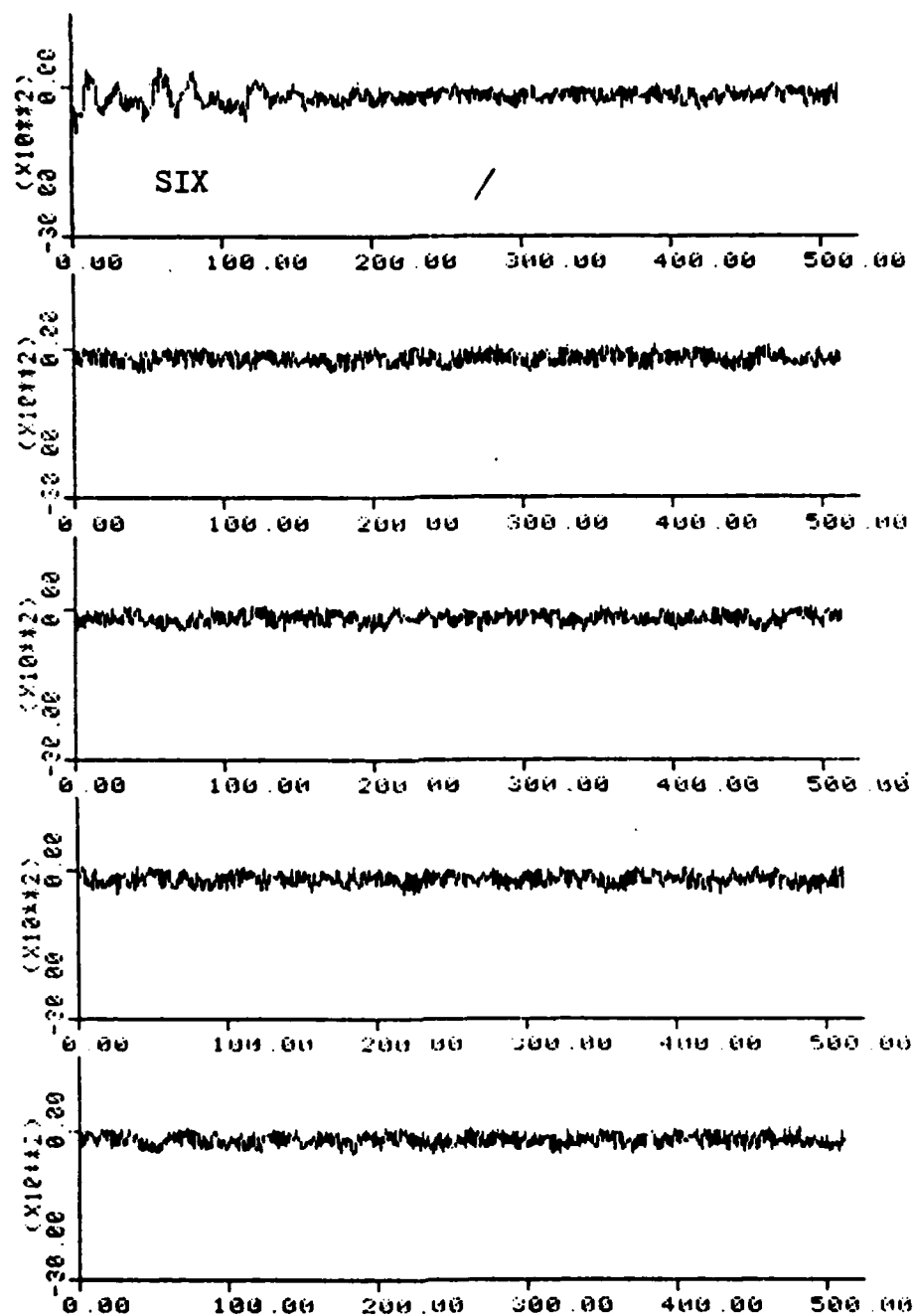


Figure IV-33.4 "FIVE...SIX...SEVEN...EIGHT" + noise

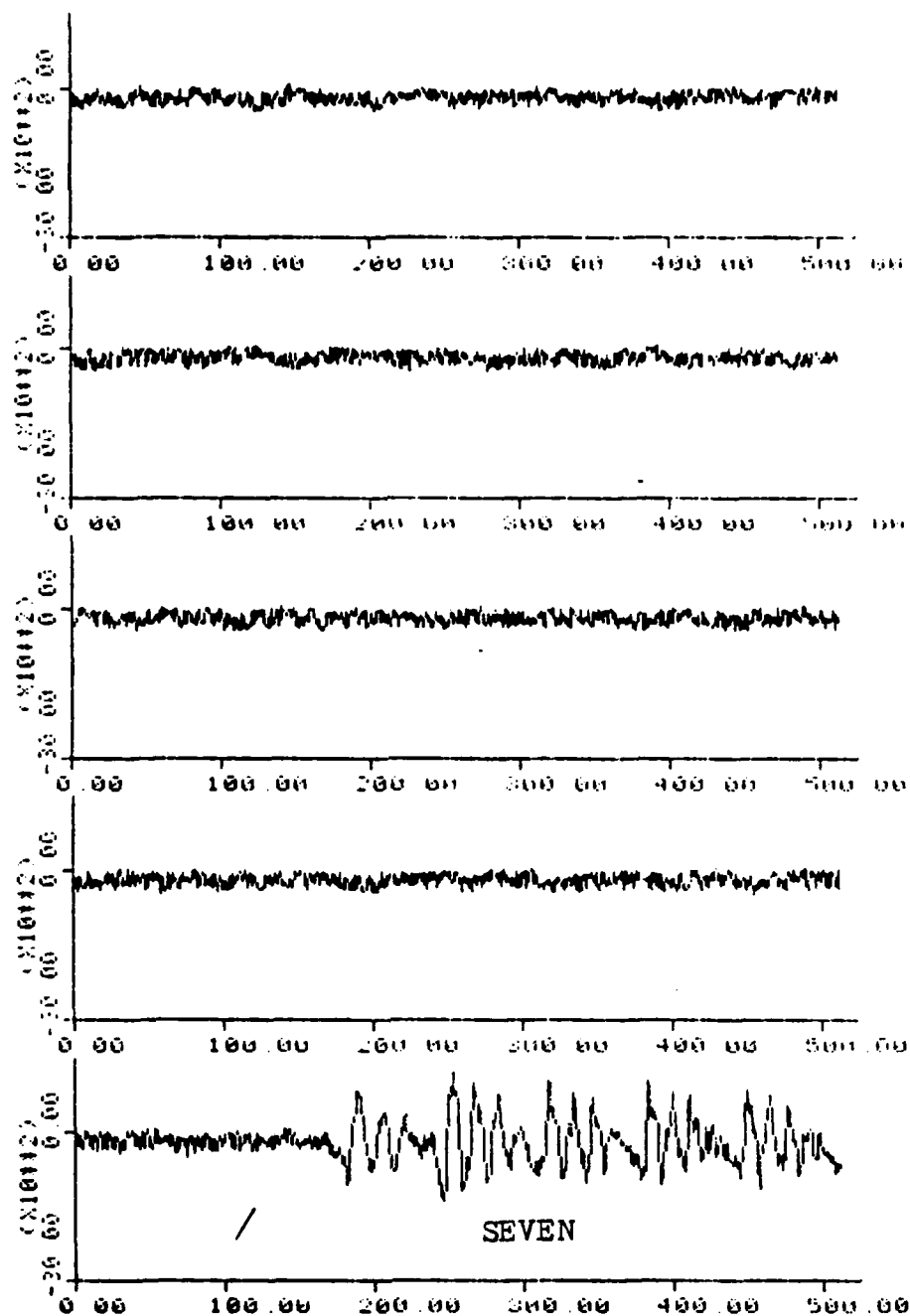


Figure IV-33.5 "FIVE...SIX...SEVEN...EIGHT" + noise

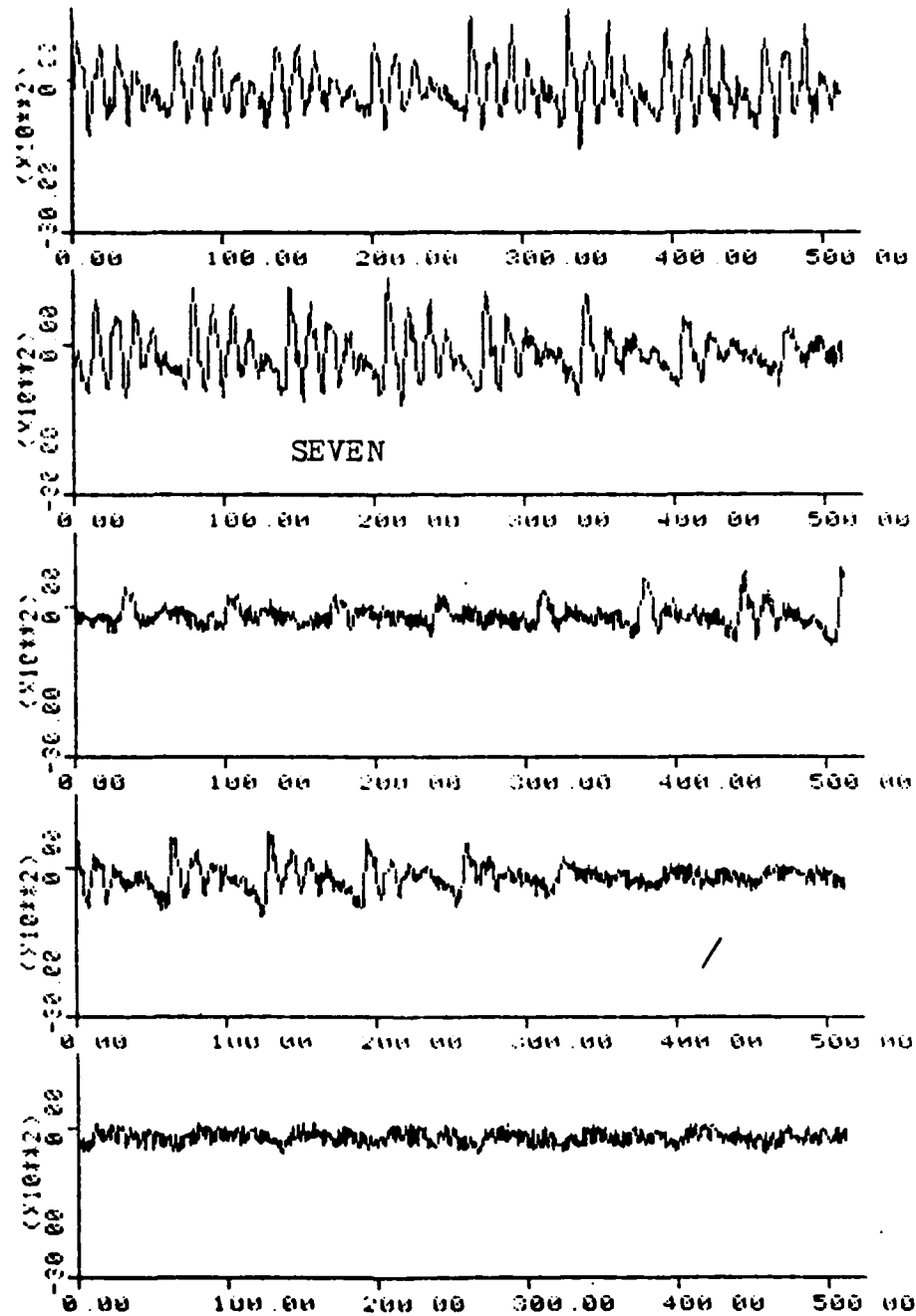


Figure IV-33.6 "FIVE...SIX...SEVEN...EIGHT" + noise

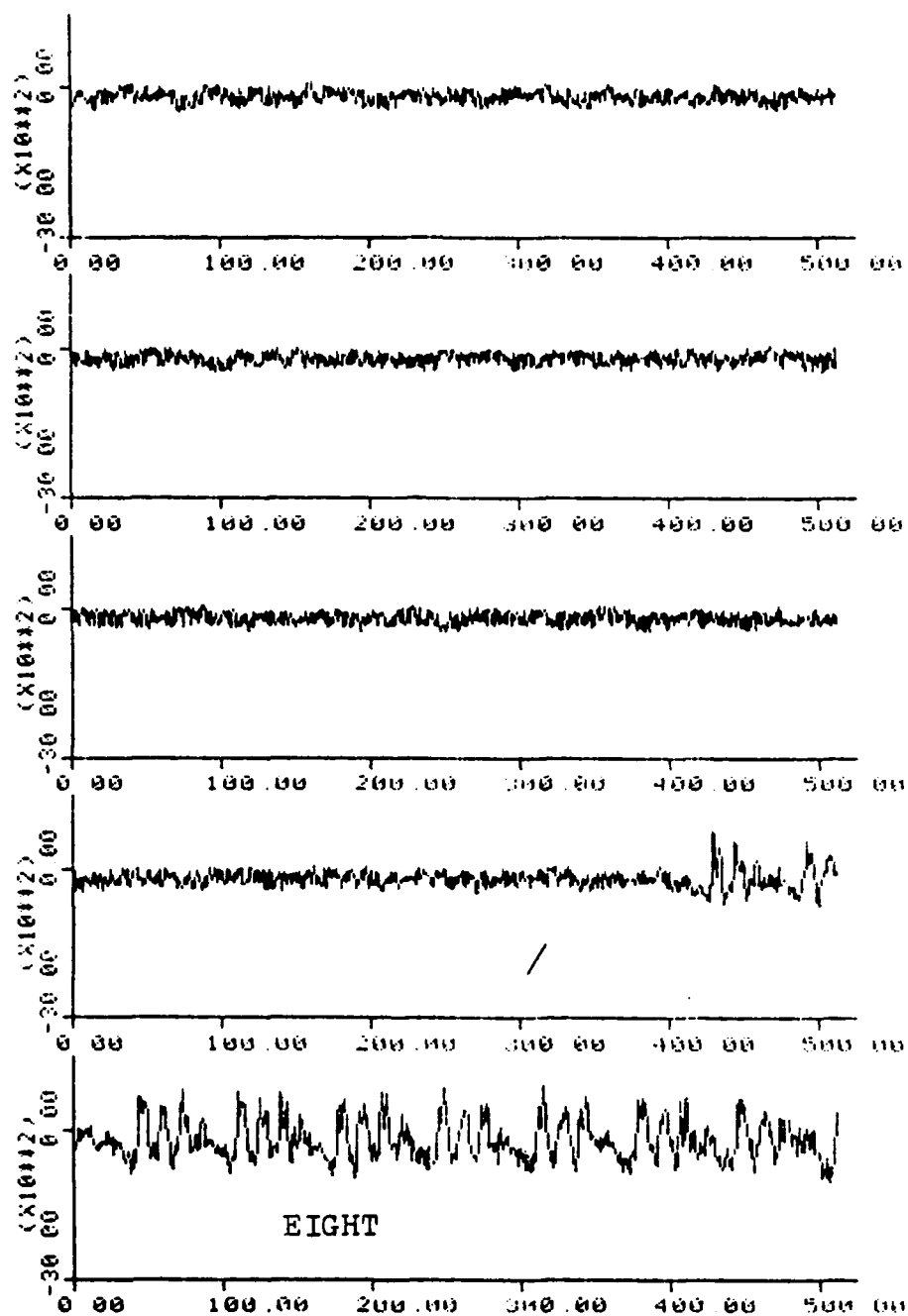


Figure IV-33.7 "FIVE...SIX...SEVEN...EIGHT" + noise

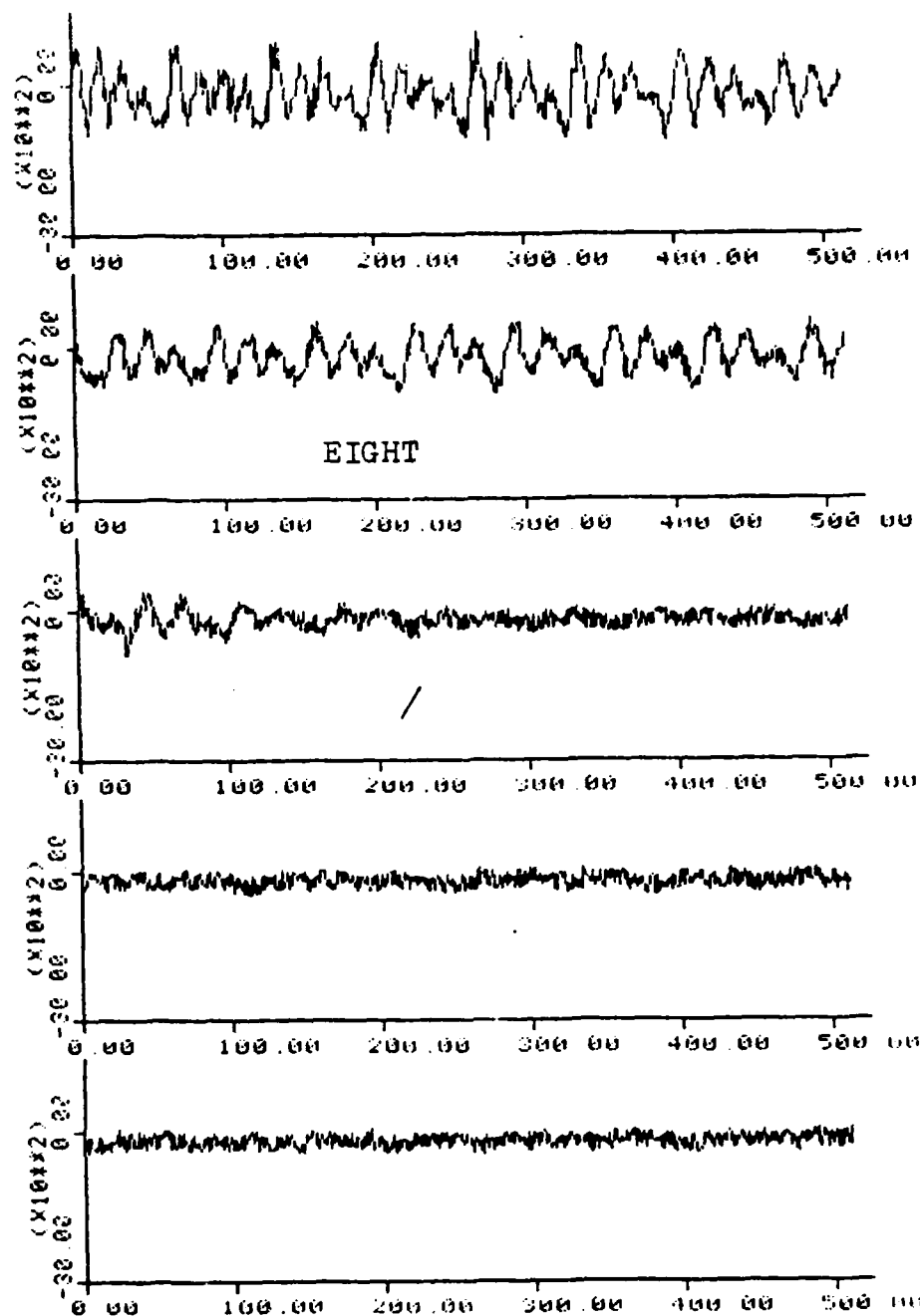


Figure IV-33.8 "FIVE...SIX...SEVEN...EIGHT" + noise

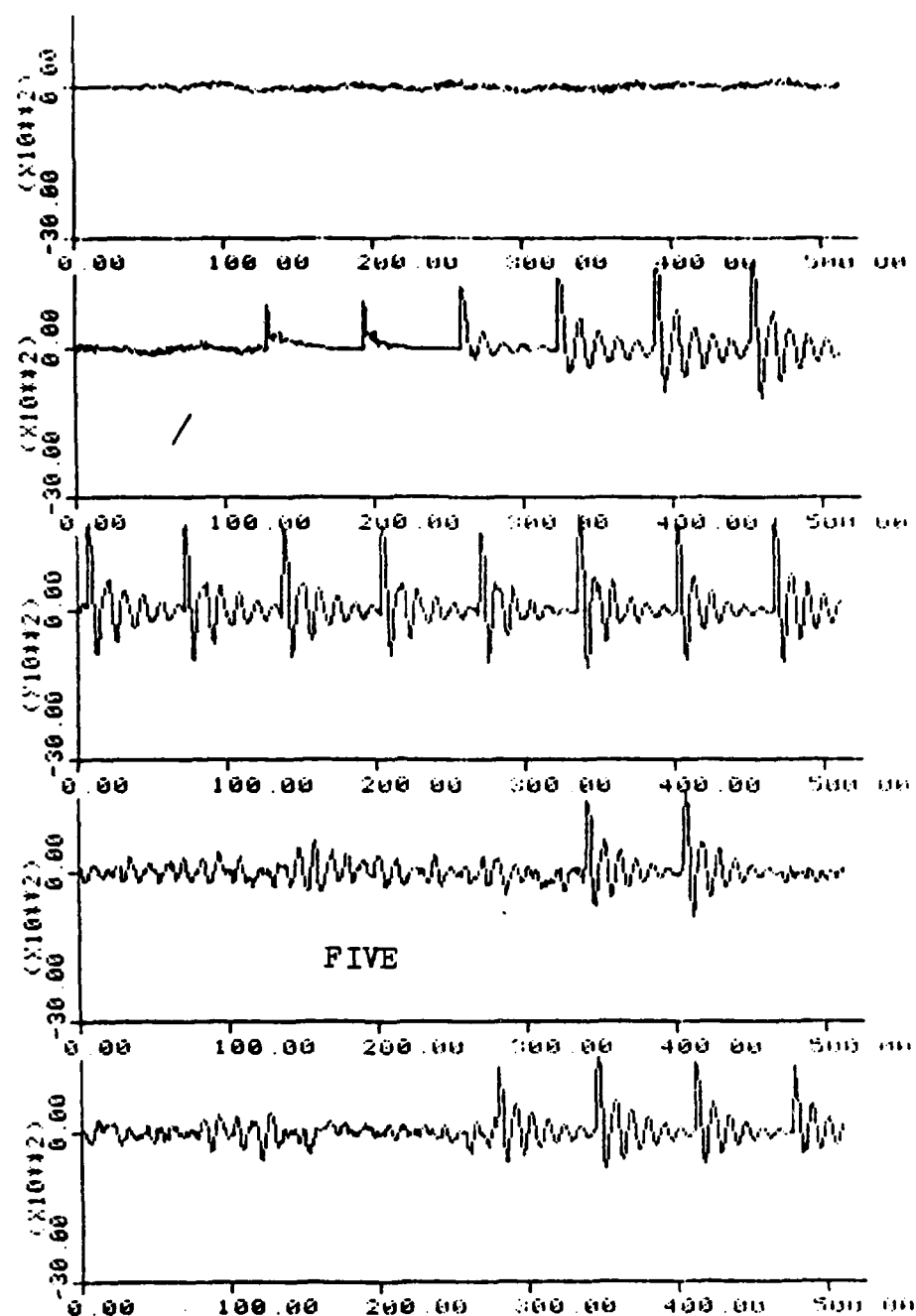


Figure IV-34.1 Vocoder Output with Input of
"FIVE...SIX...SEVEN...EIGHT" + noise

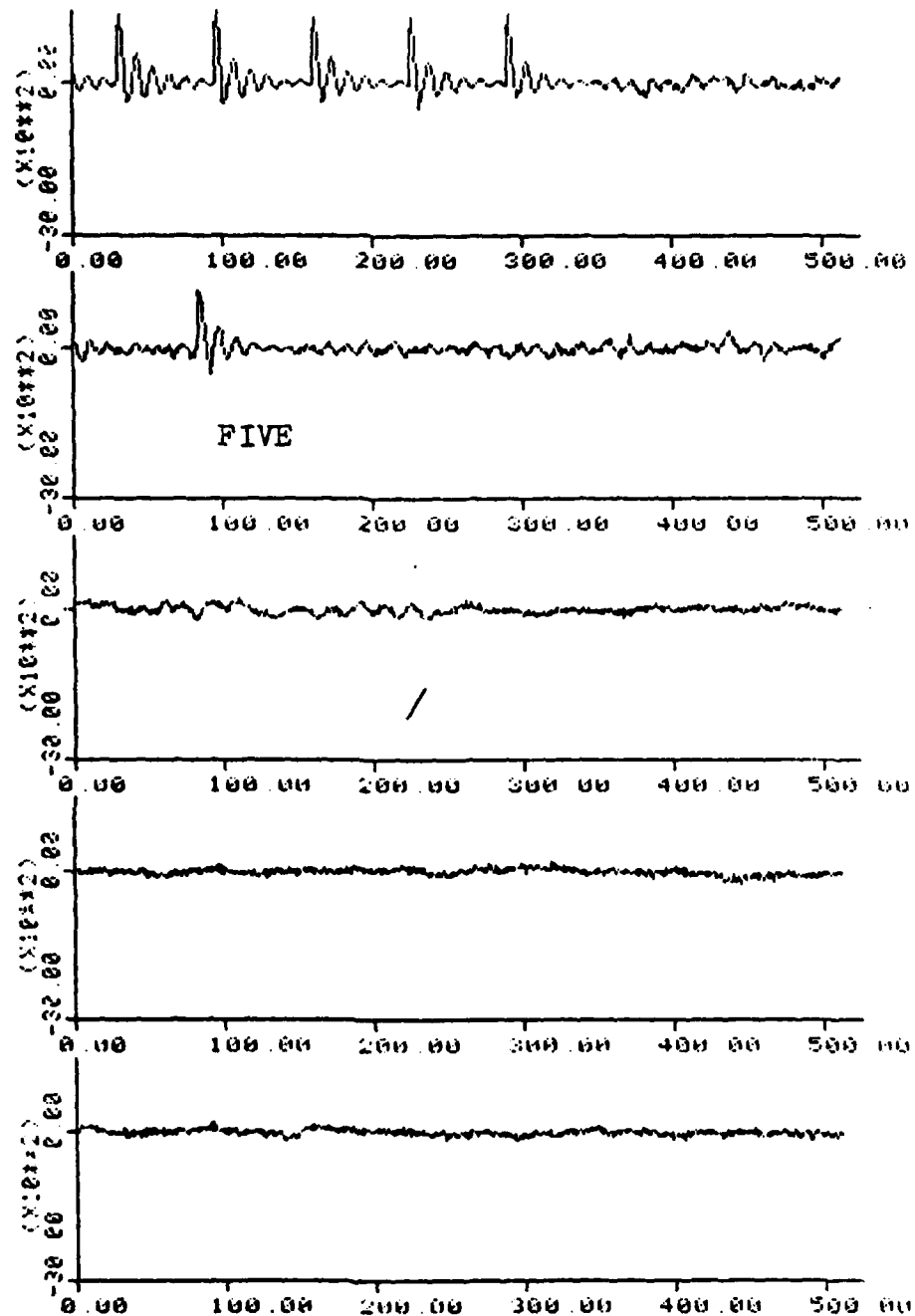


Figure IV-34.2 "FIVE...SIX...SEVEN...EIGHT" + noise

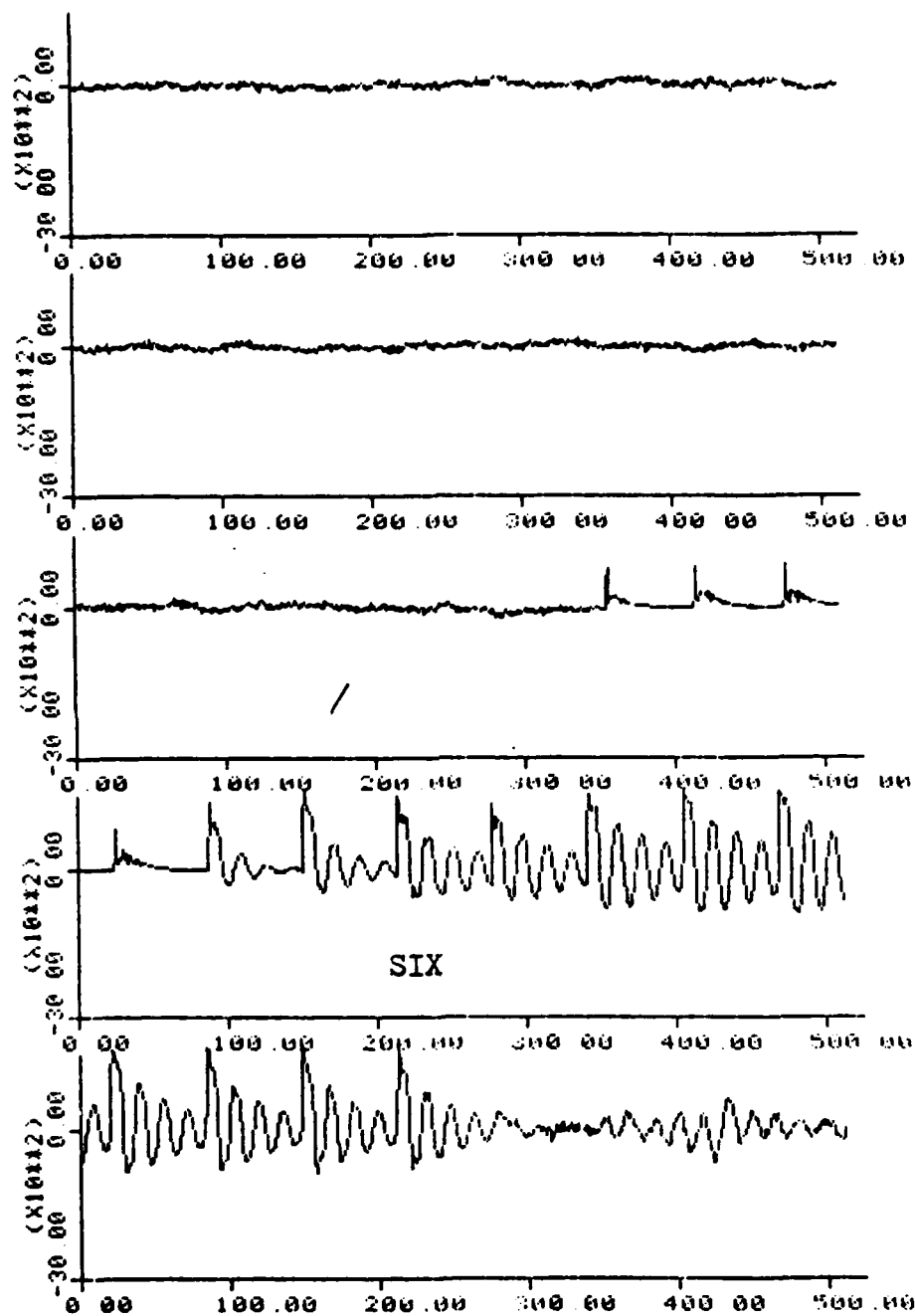


Figure IV-34.3 "FIVE...SIX...SEVEN...EIGHT" + noise

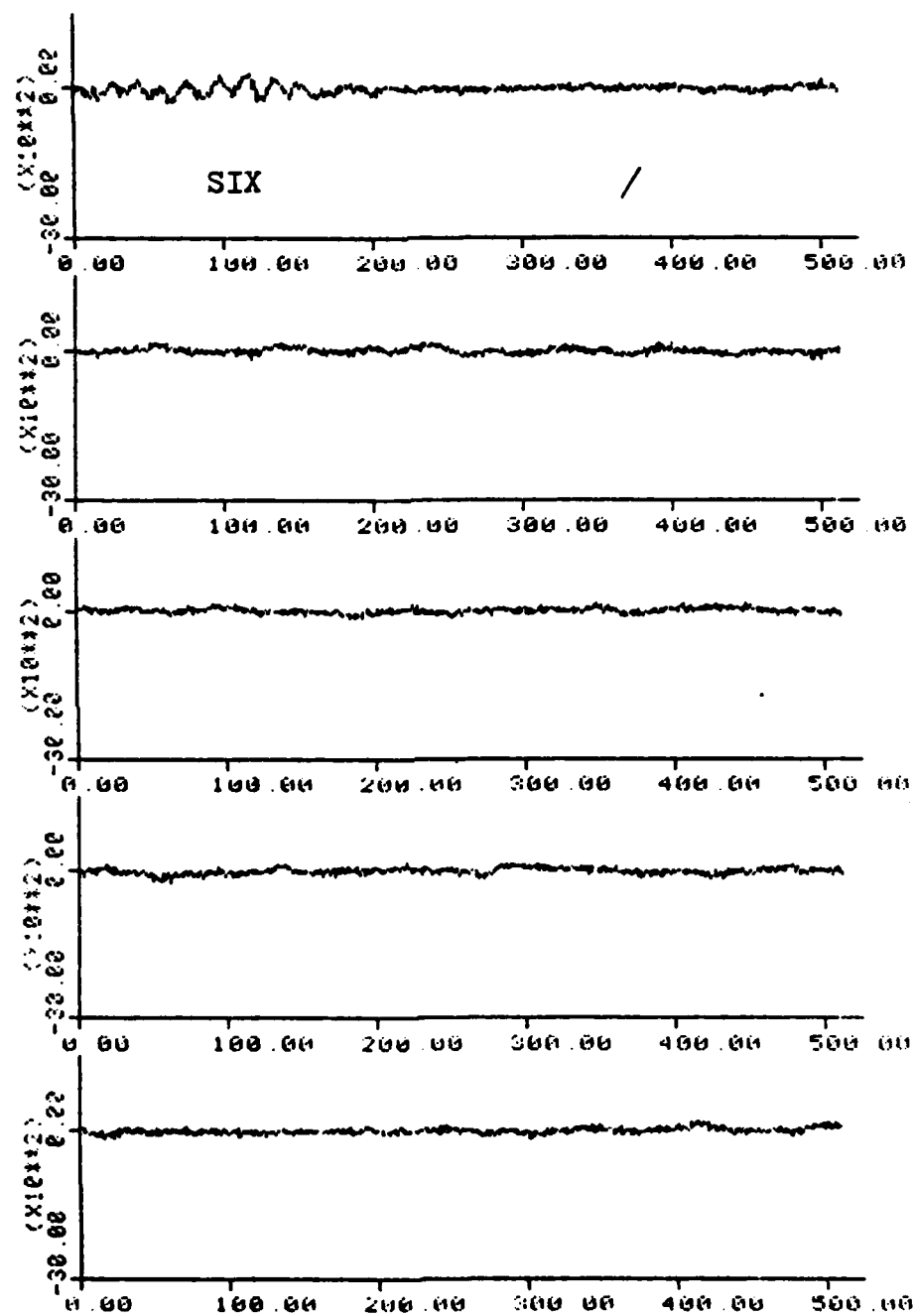


Figure IV-34.4 "FIVE...SIX...SEVEN...EIGHT" + noise

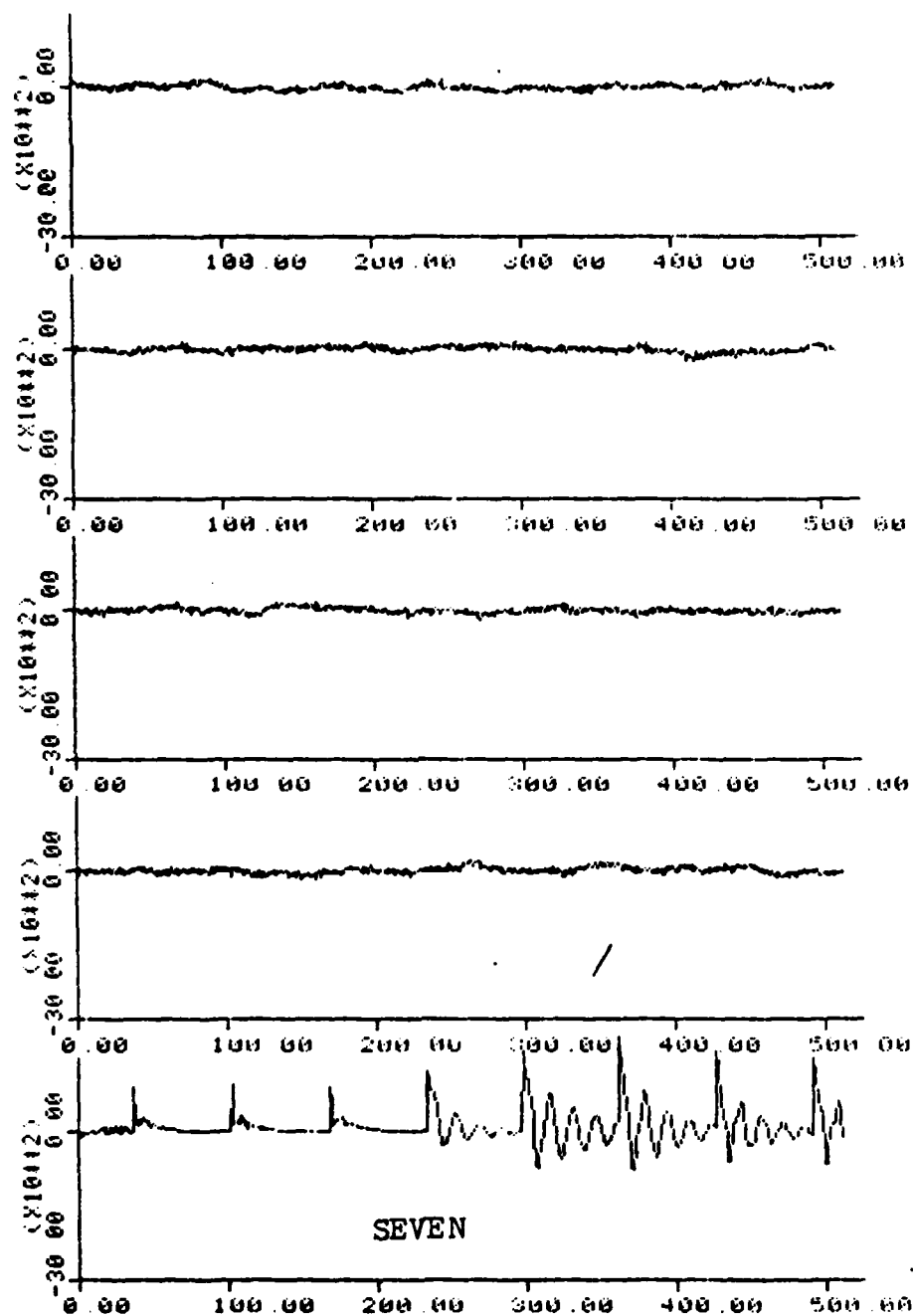


Figure IV-34.5 "FIVE...SIX...SEVEN...EIGHT" + noise

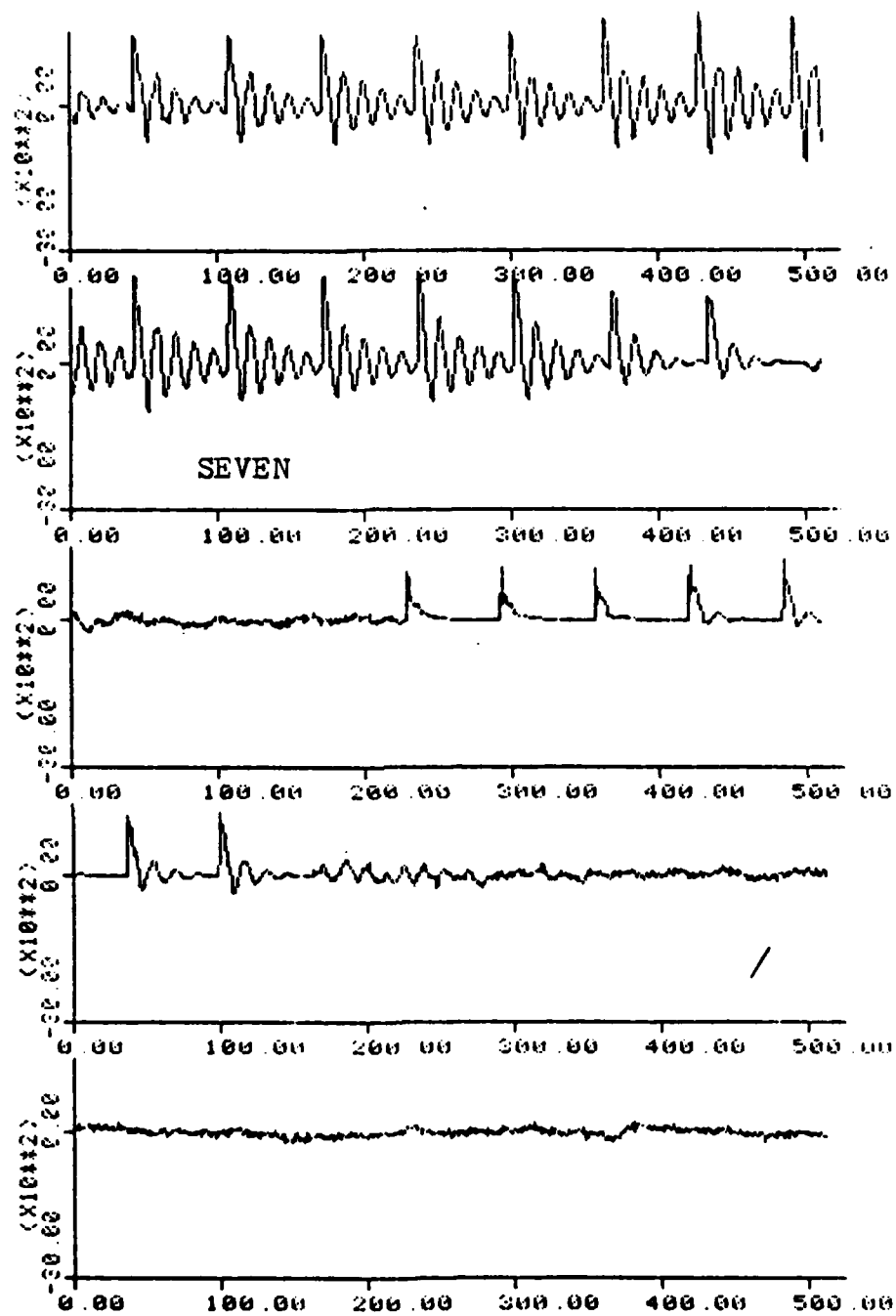


Figure IV-34.6 "FIVE...SIX...SEVEN...EIGHT" + noise

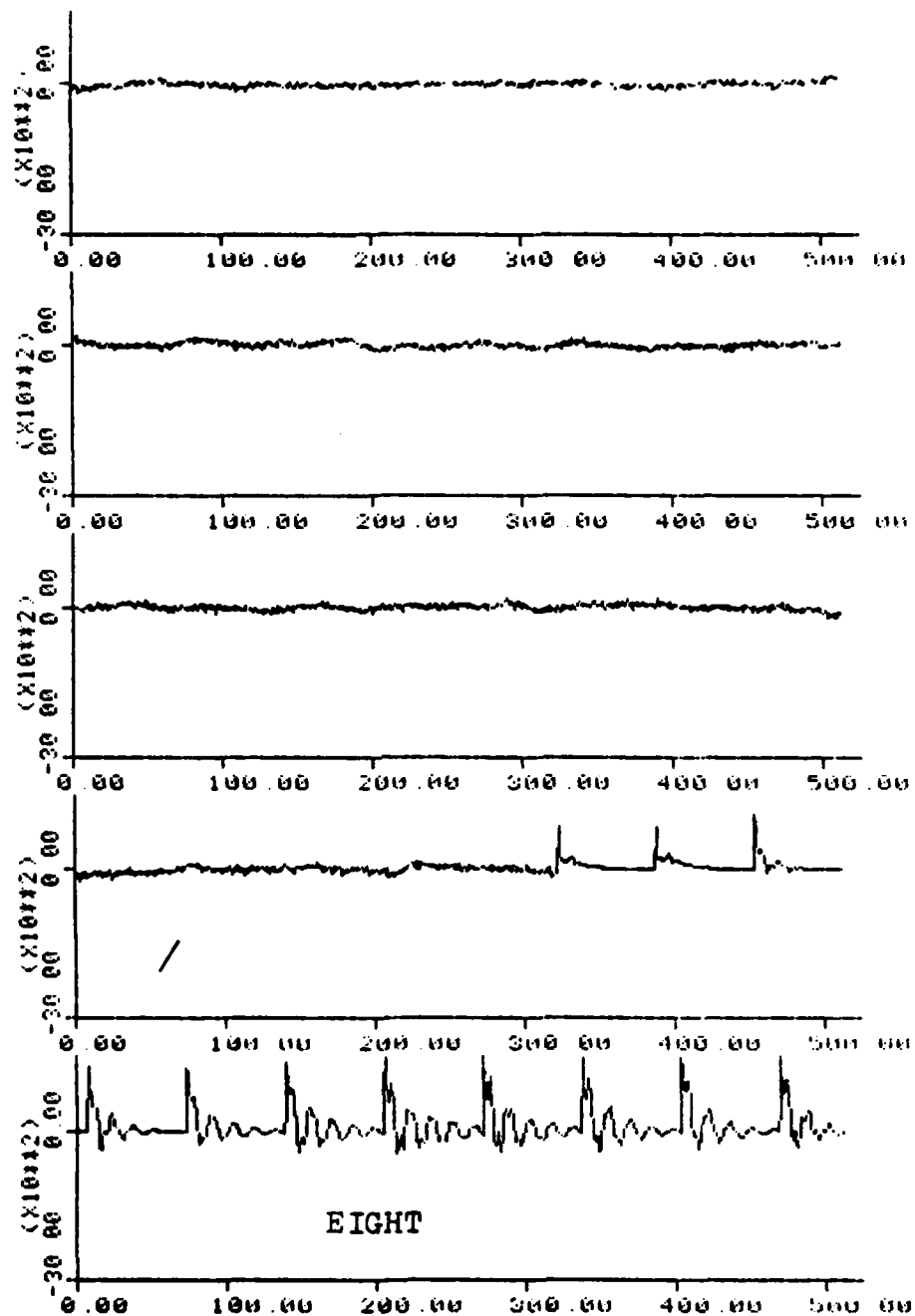


Figure IV-34.7 "FIVE...SIX...SEVEN...EIGHT" + noise

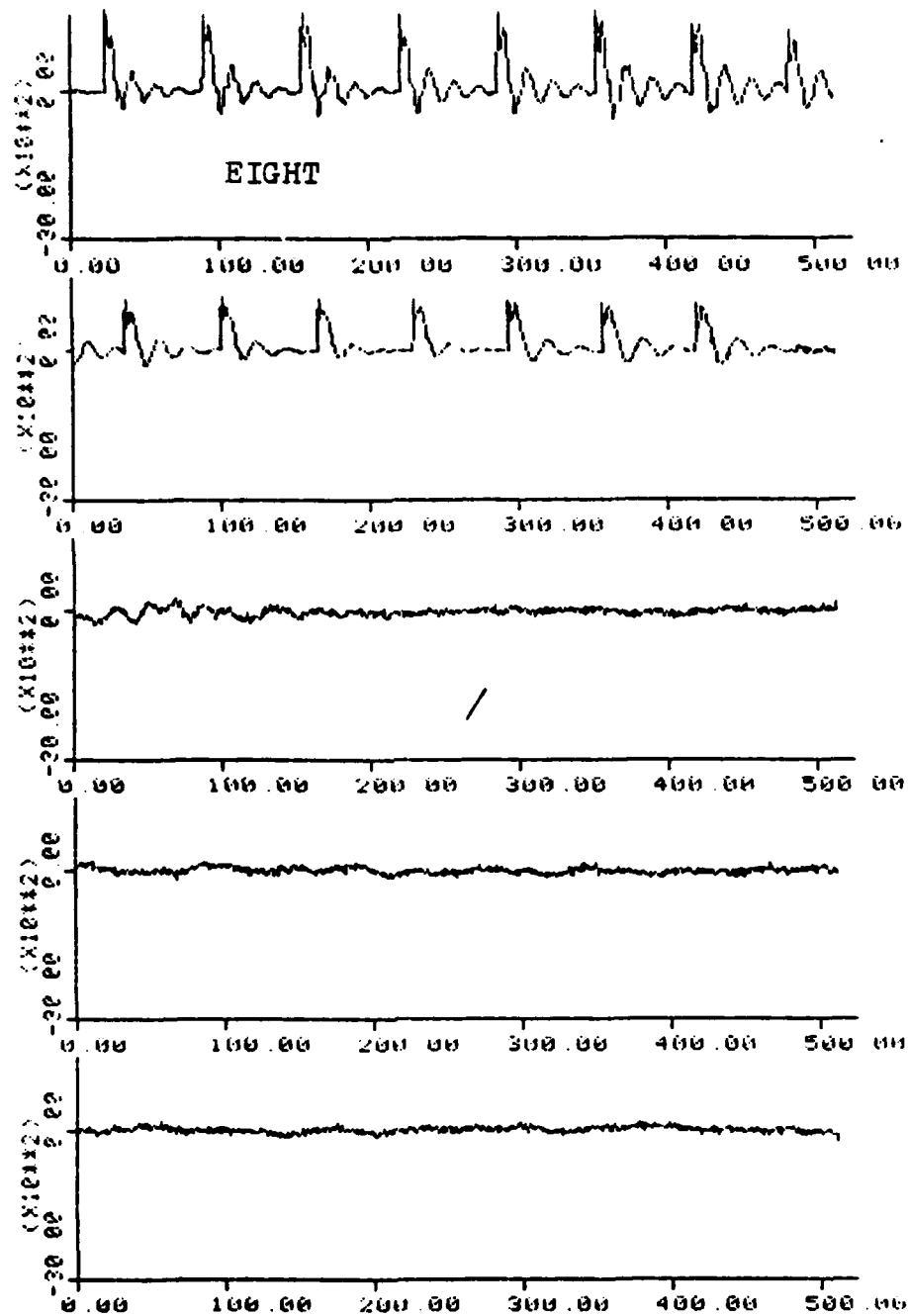


Figure IV-34.8 "FIVE...SIX...SEVEN...EIGHT" + noise

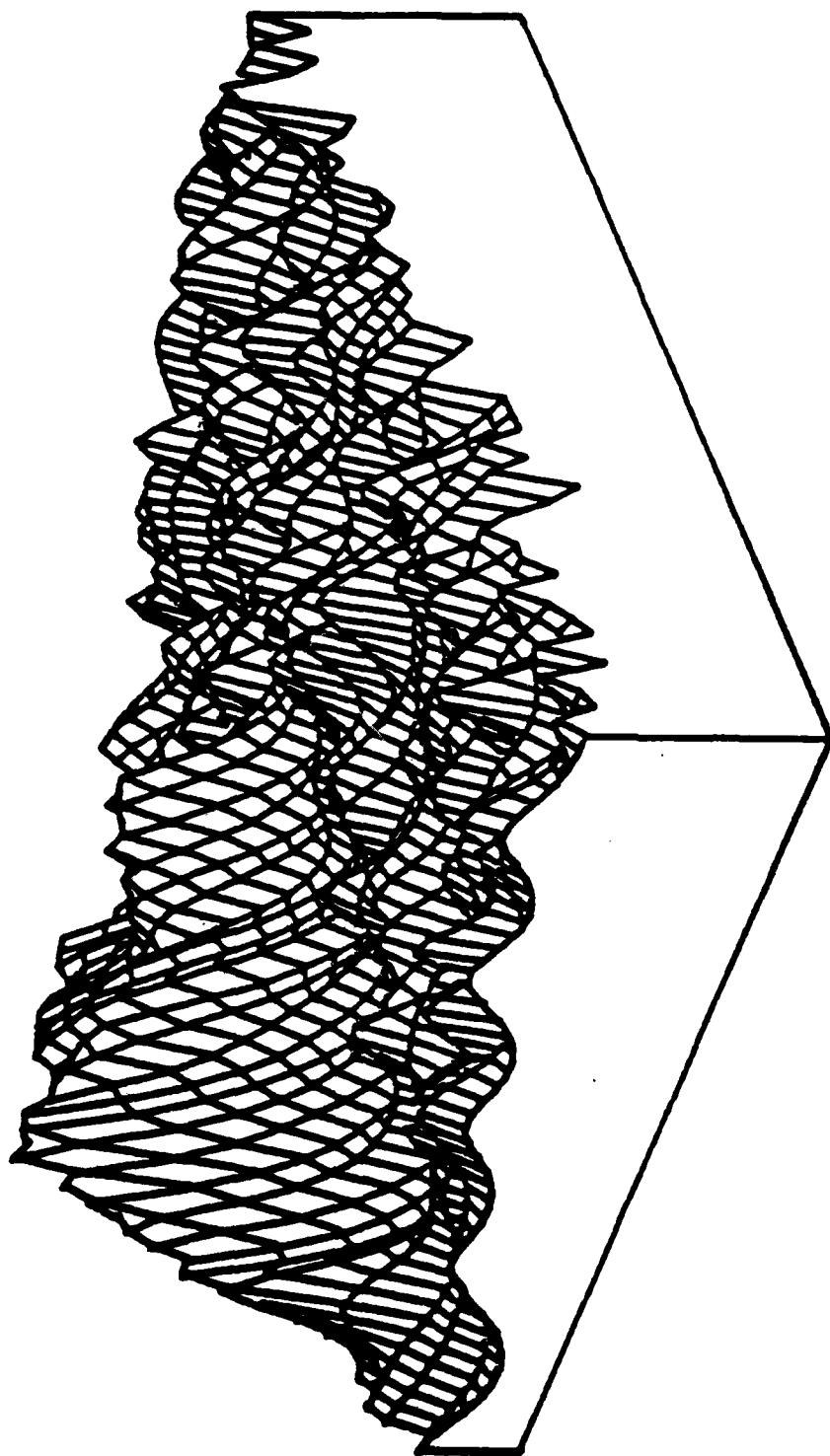


Figure IV-35 "ONE" + noise

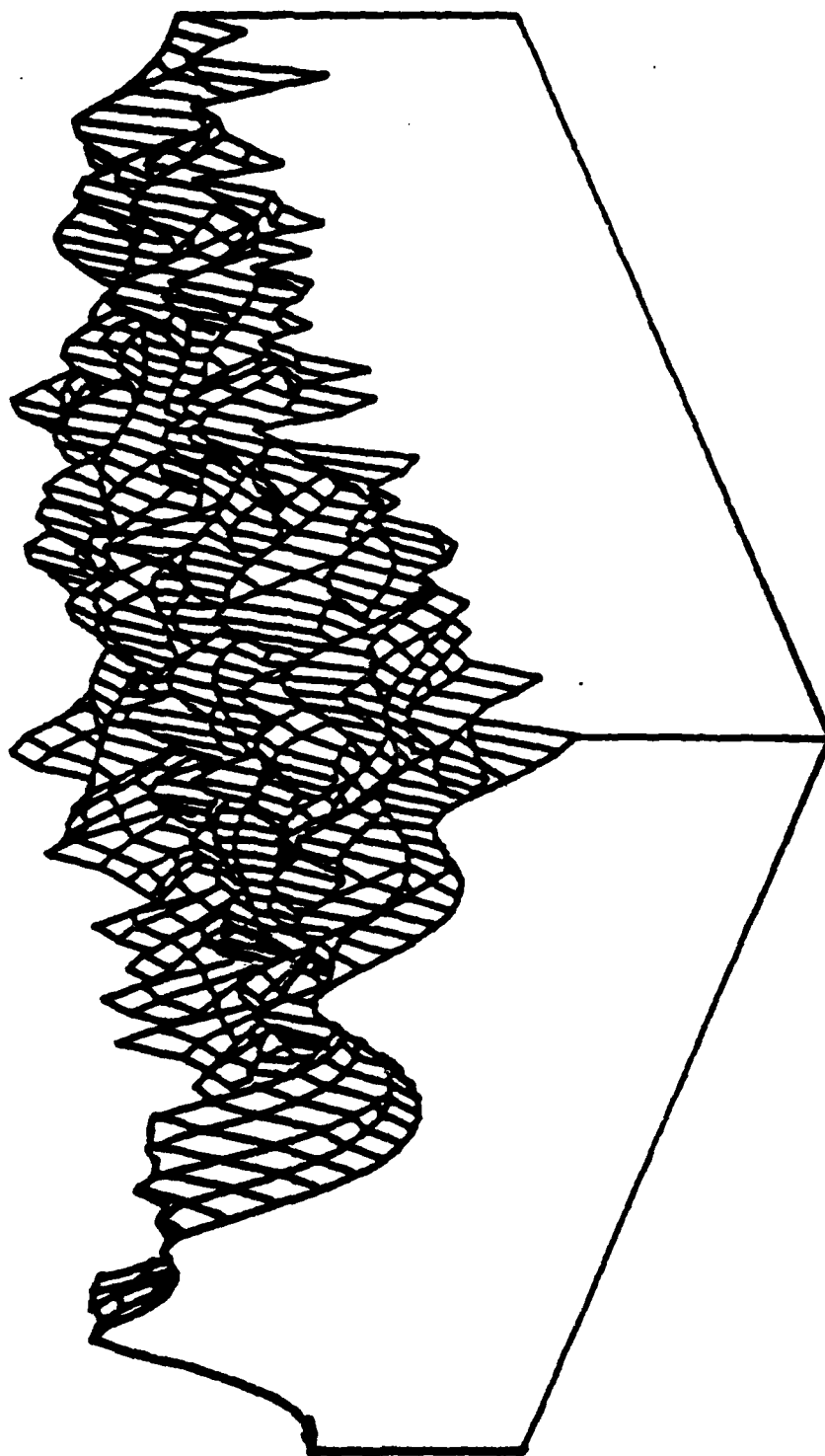


Figure IV-36 "TWC" + noise

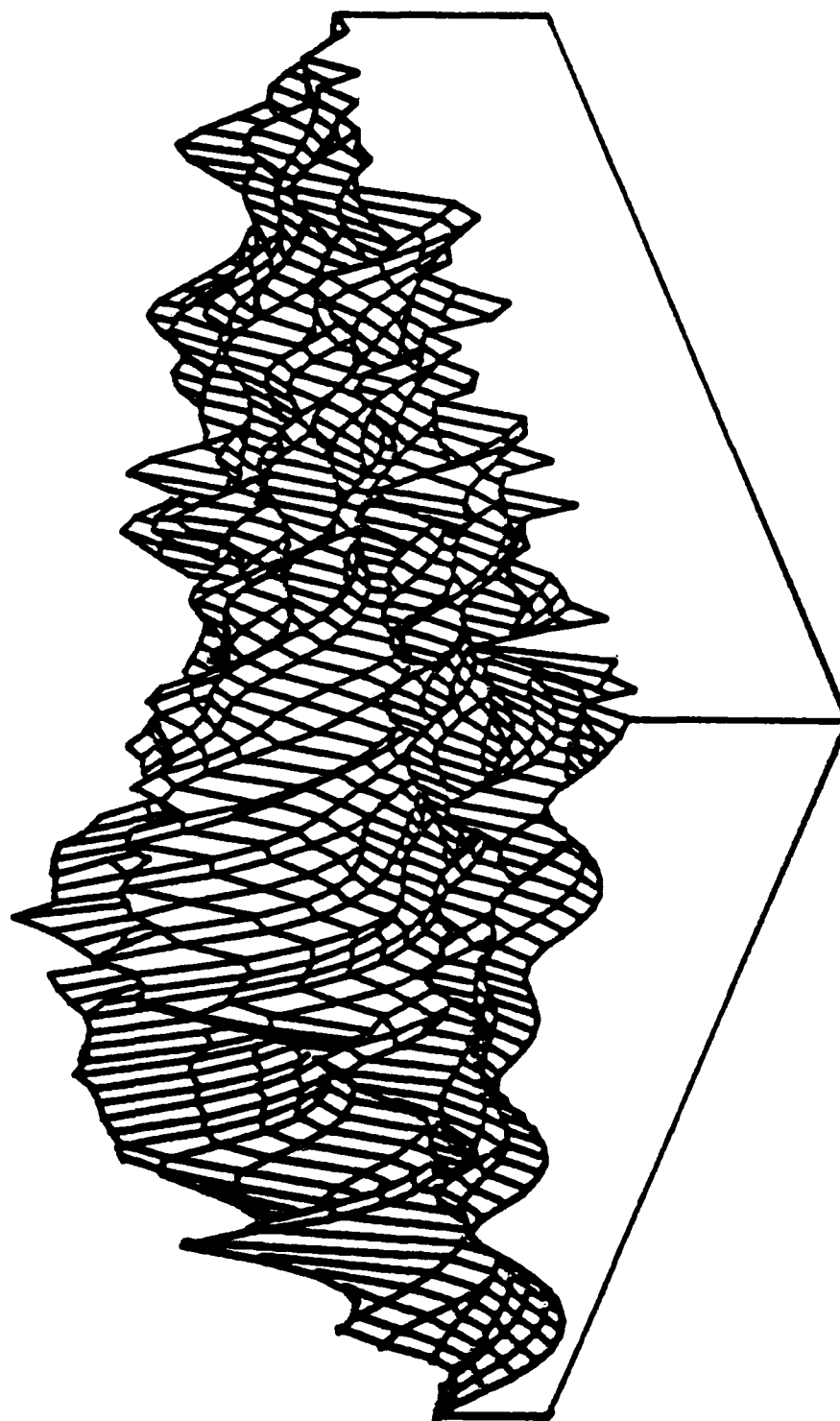


Figure IV-37 "THREE" + noise

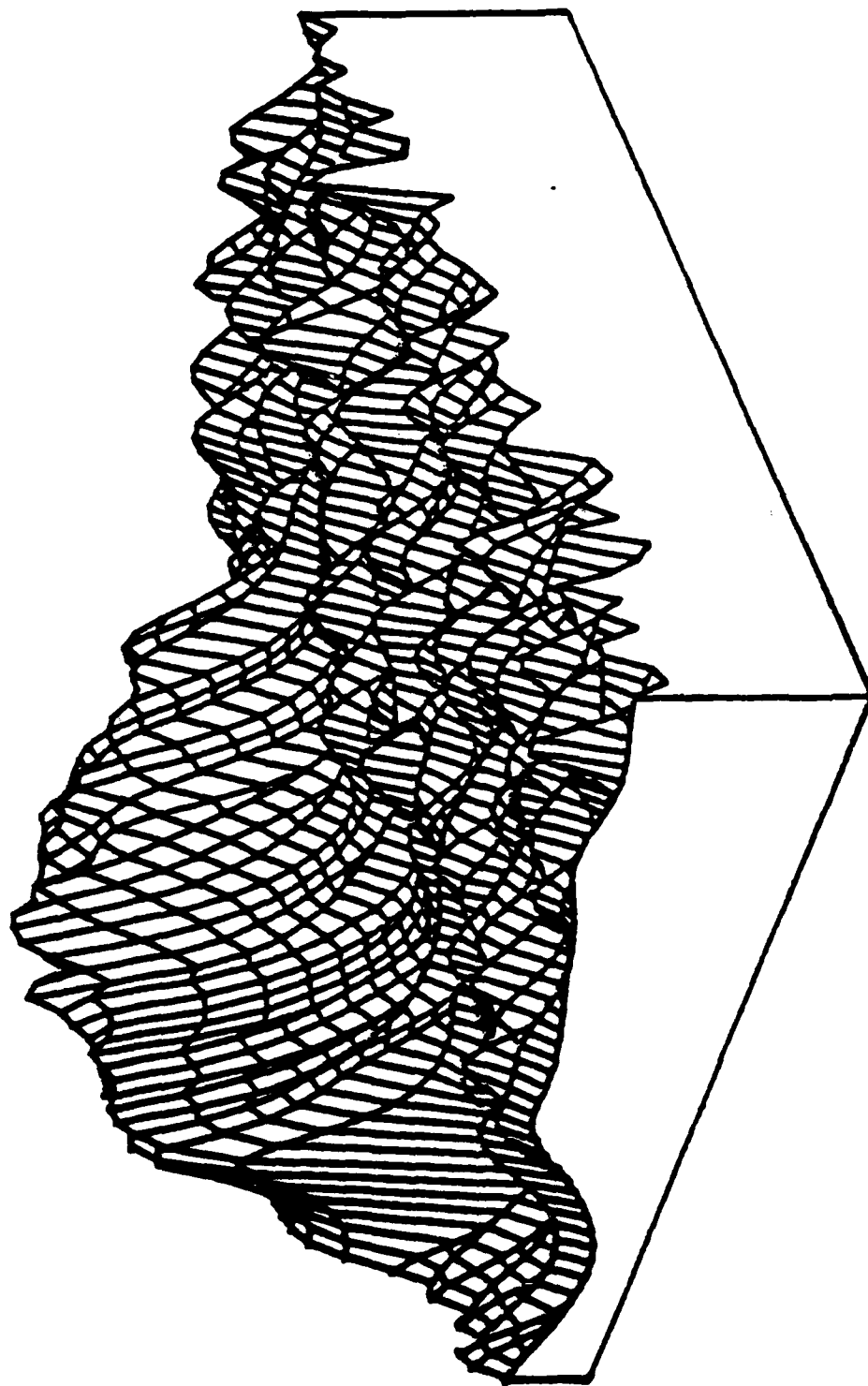


Figure IV-38 "FOUR" + noise

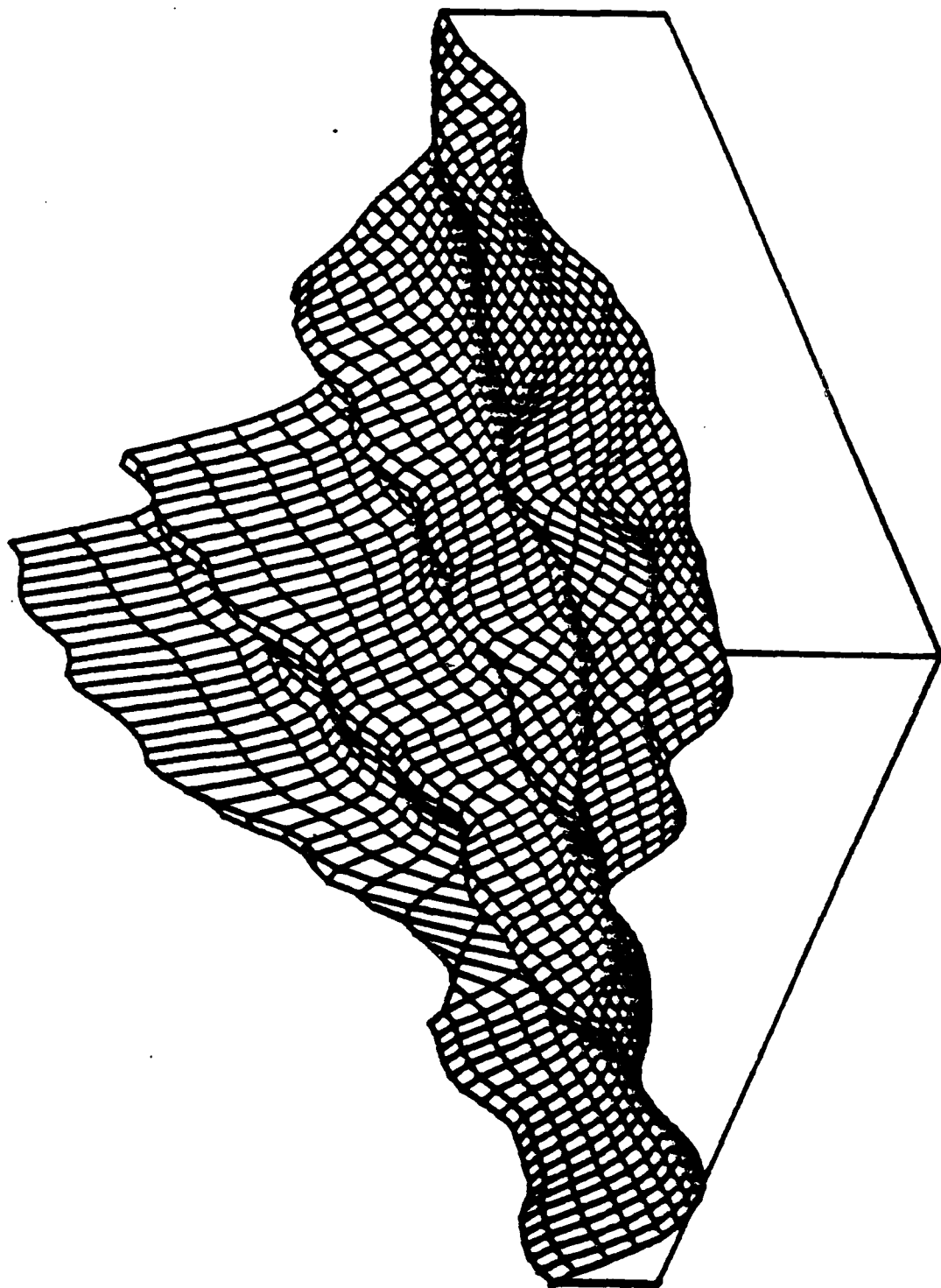


Figure IV-39.2 "FIVE" + noise

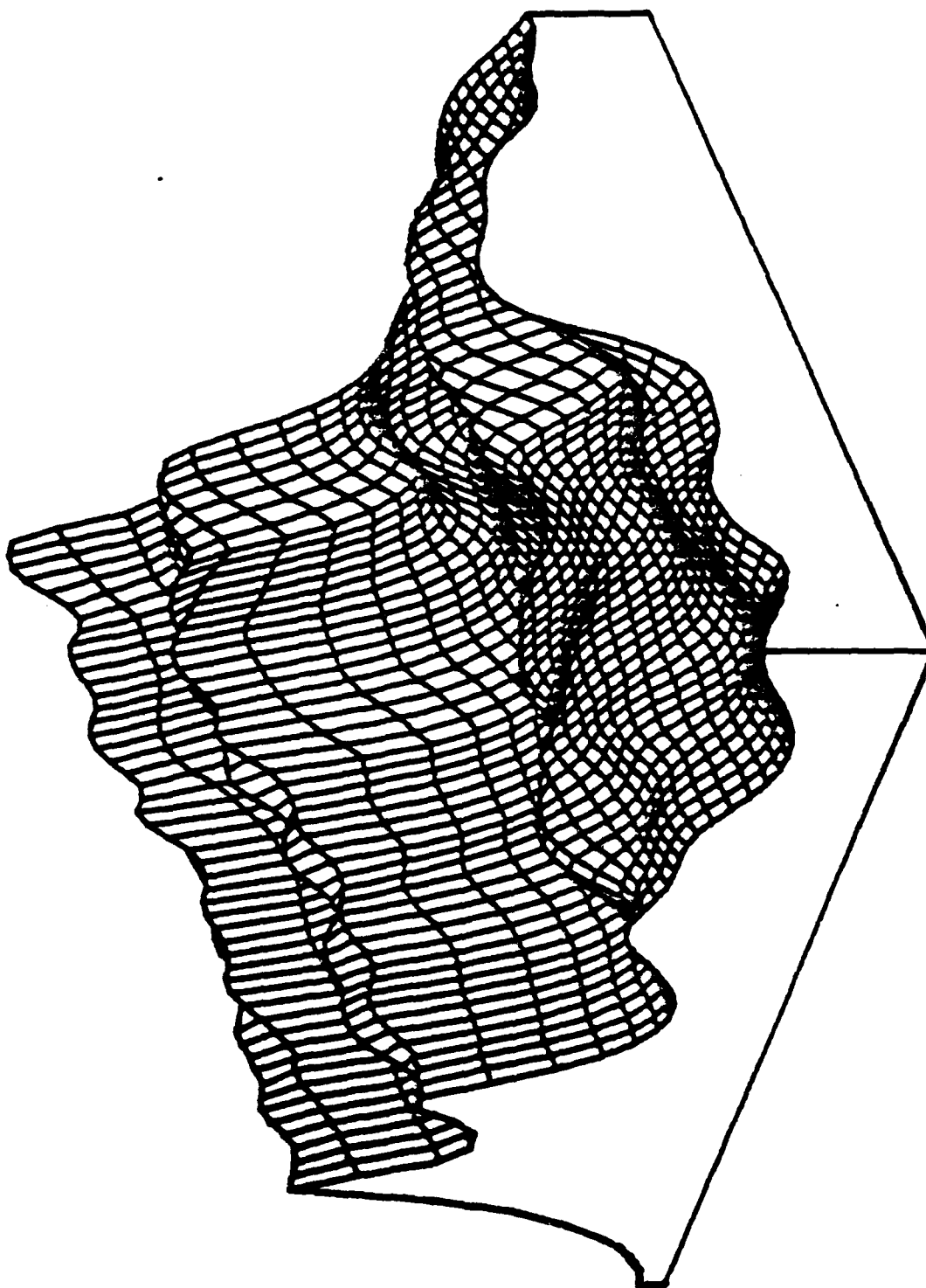


Figure IV-39.3 "FIVE" + noise

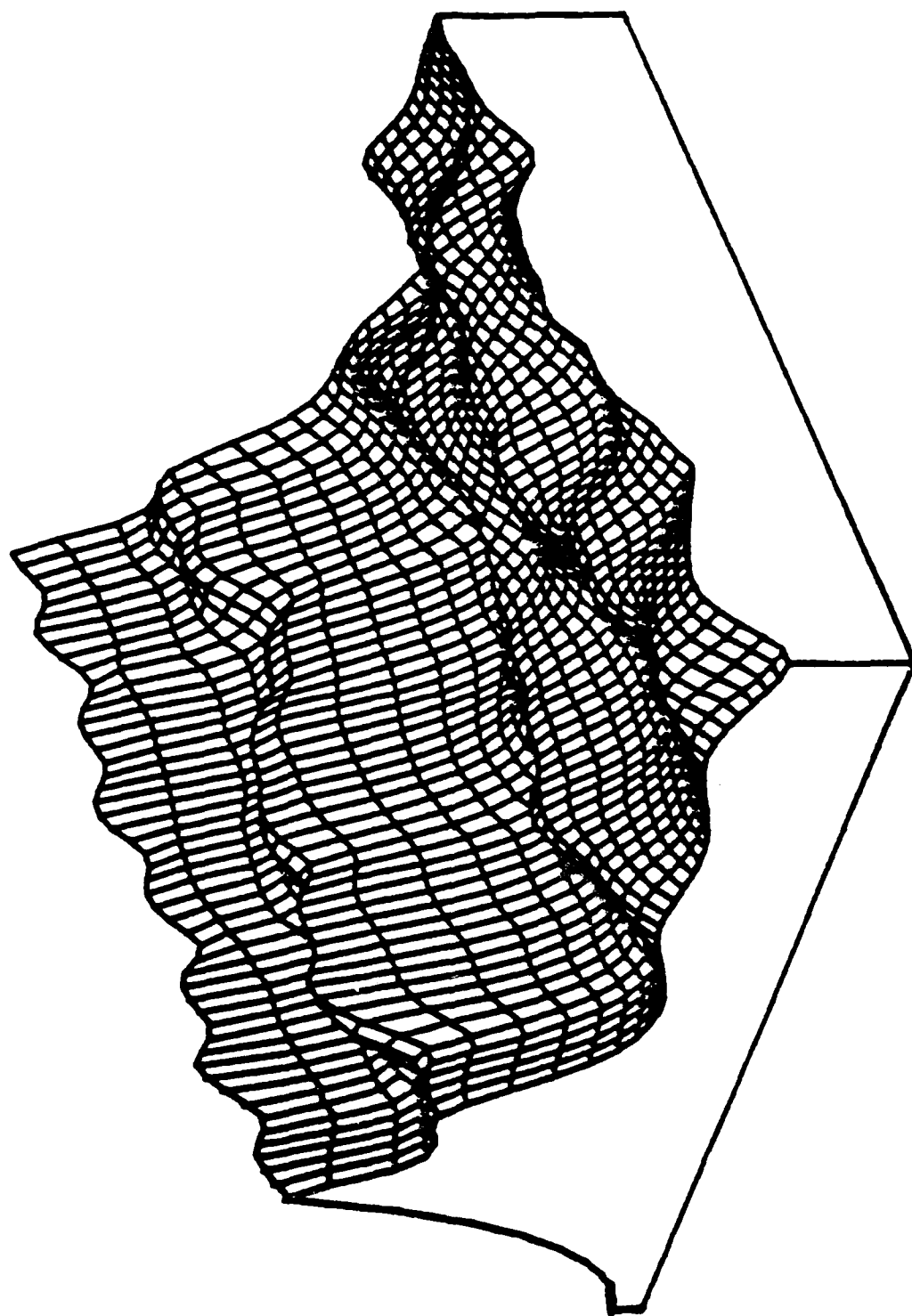


Figure IV-39.4 "FIVE" + noise

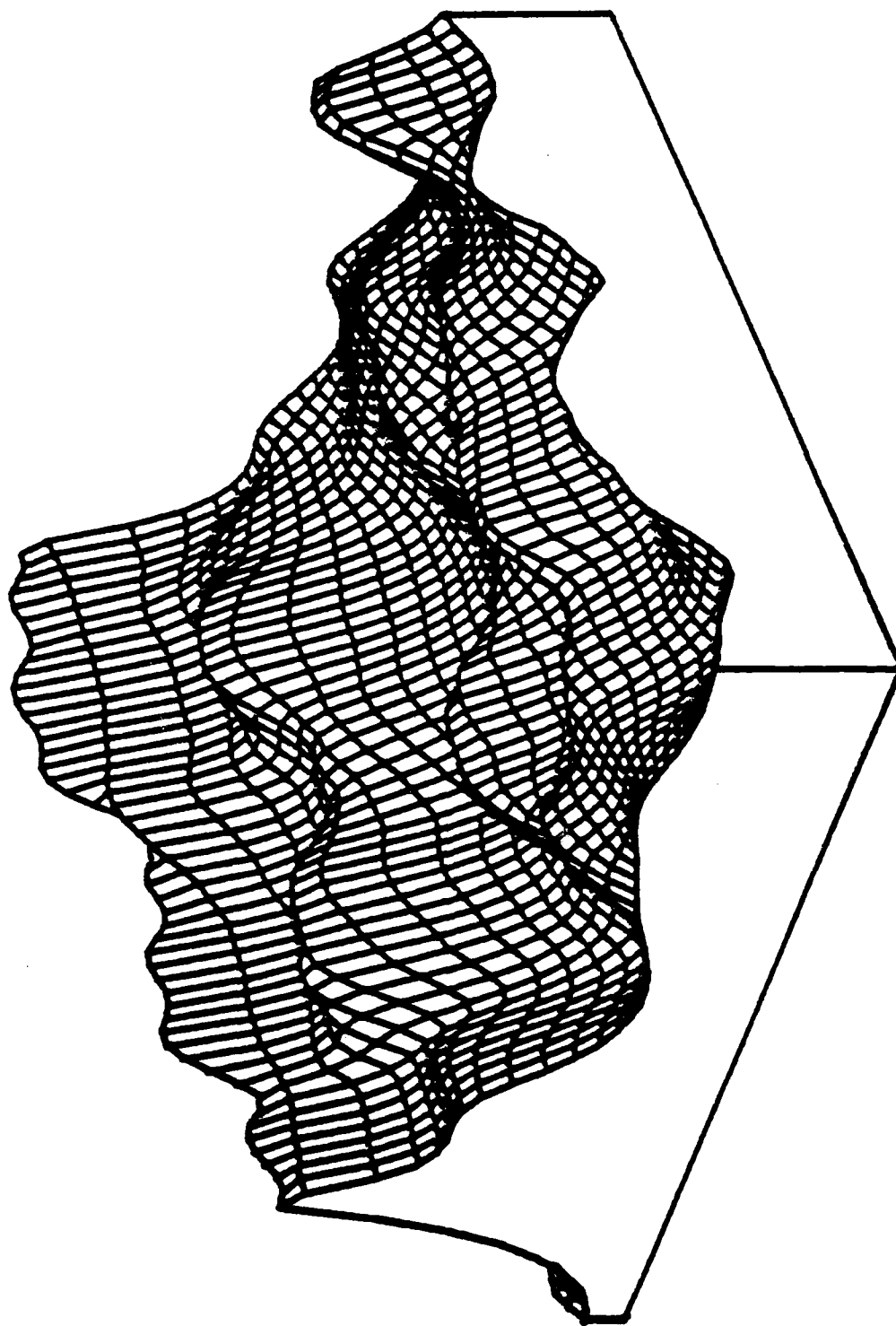


Figure IV-39.5 "FIVE" + noise

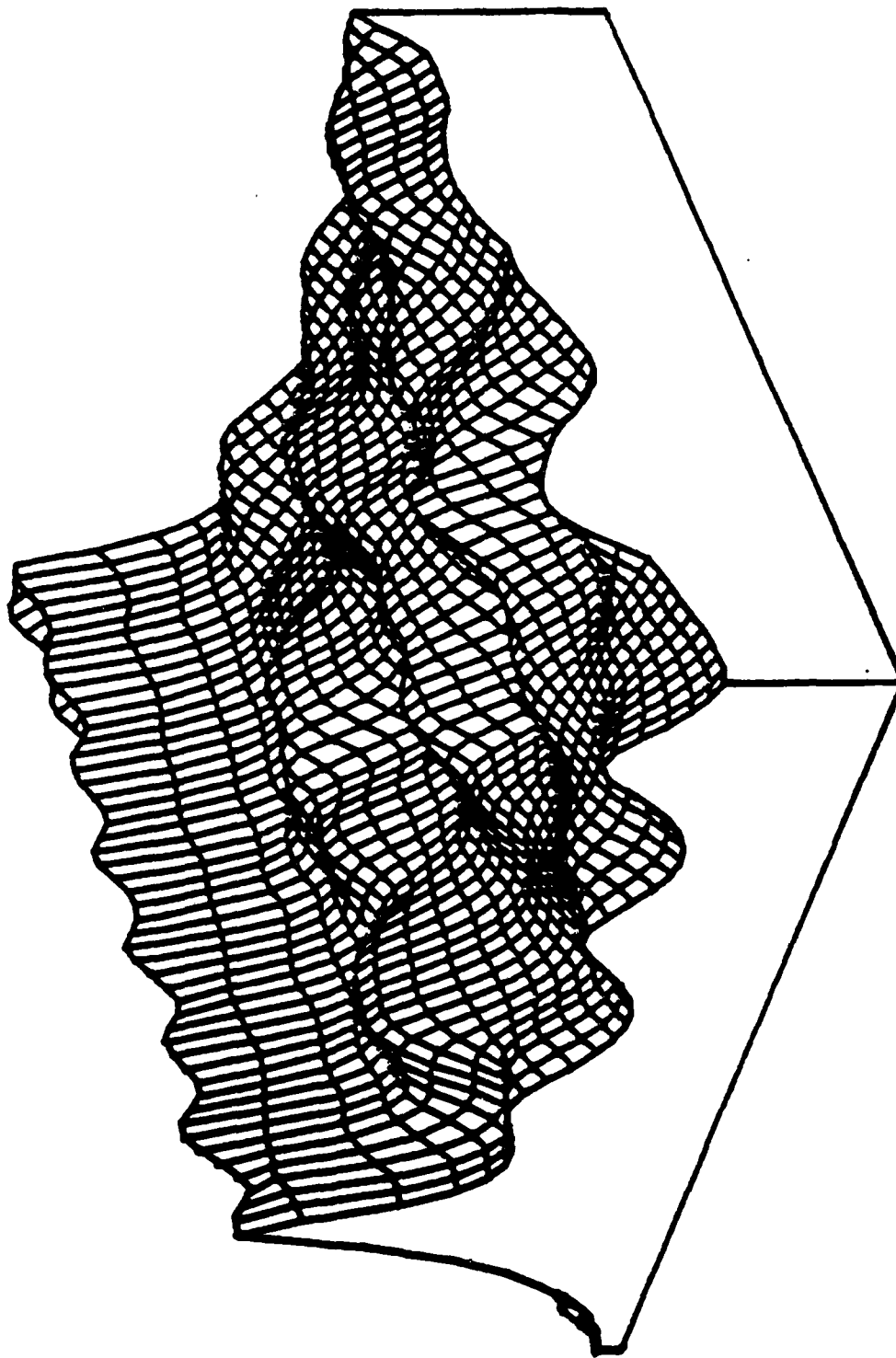


Figure IV-39.6 "FIVE" + noise

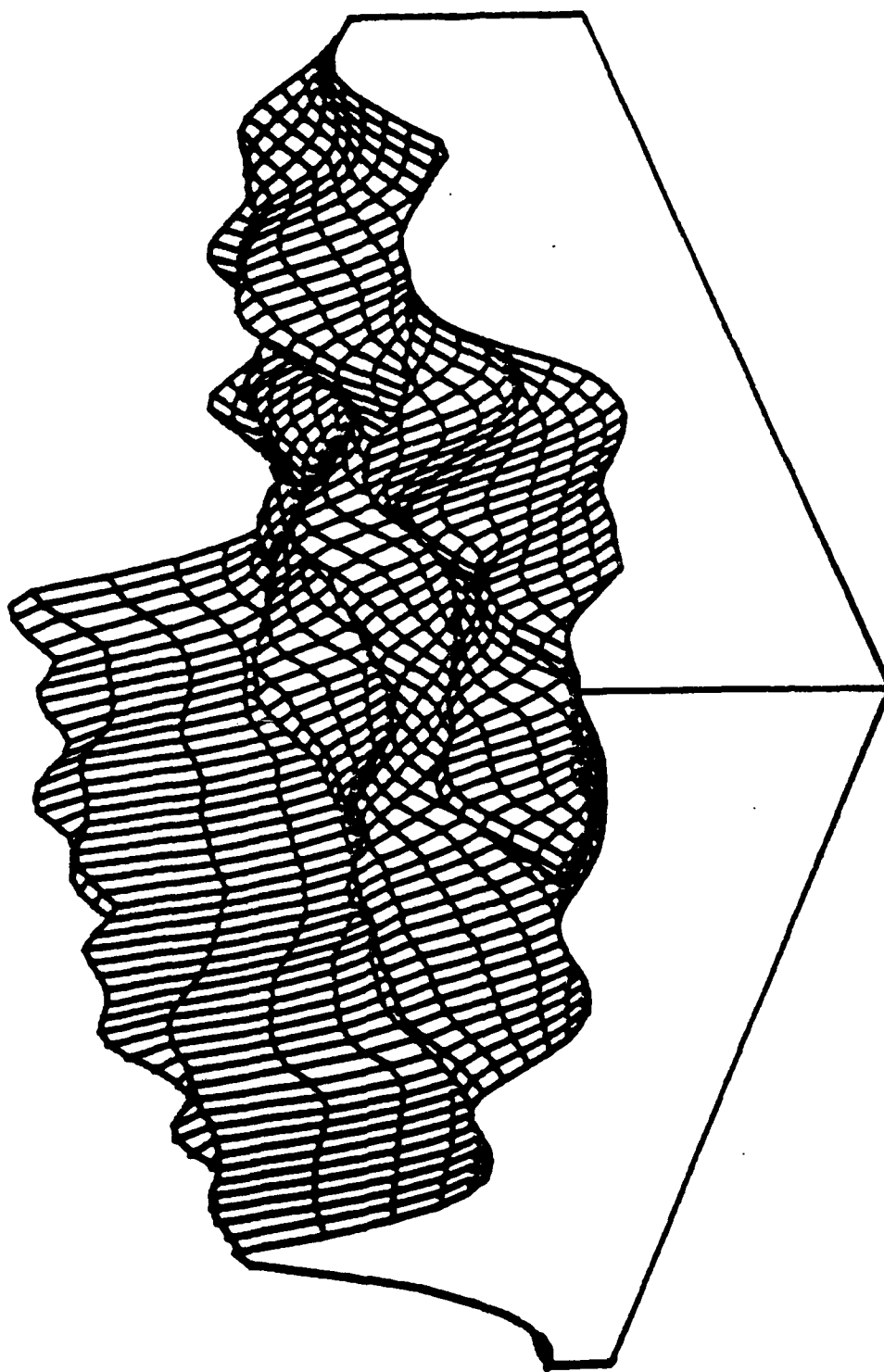


Figure IV-39.7 "FIVE" + noise

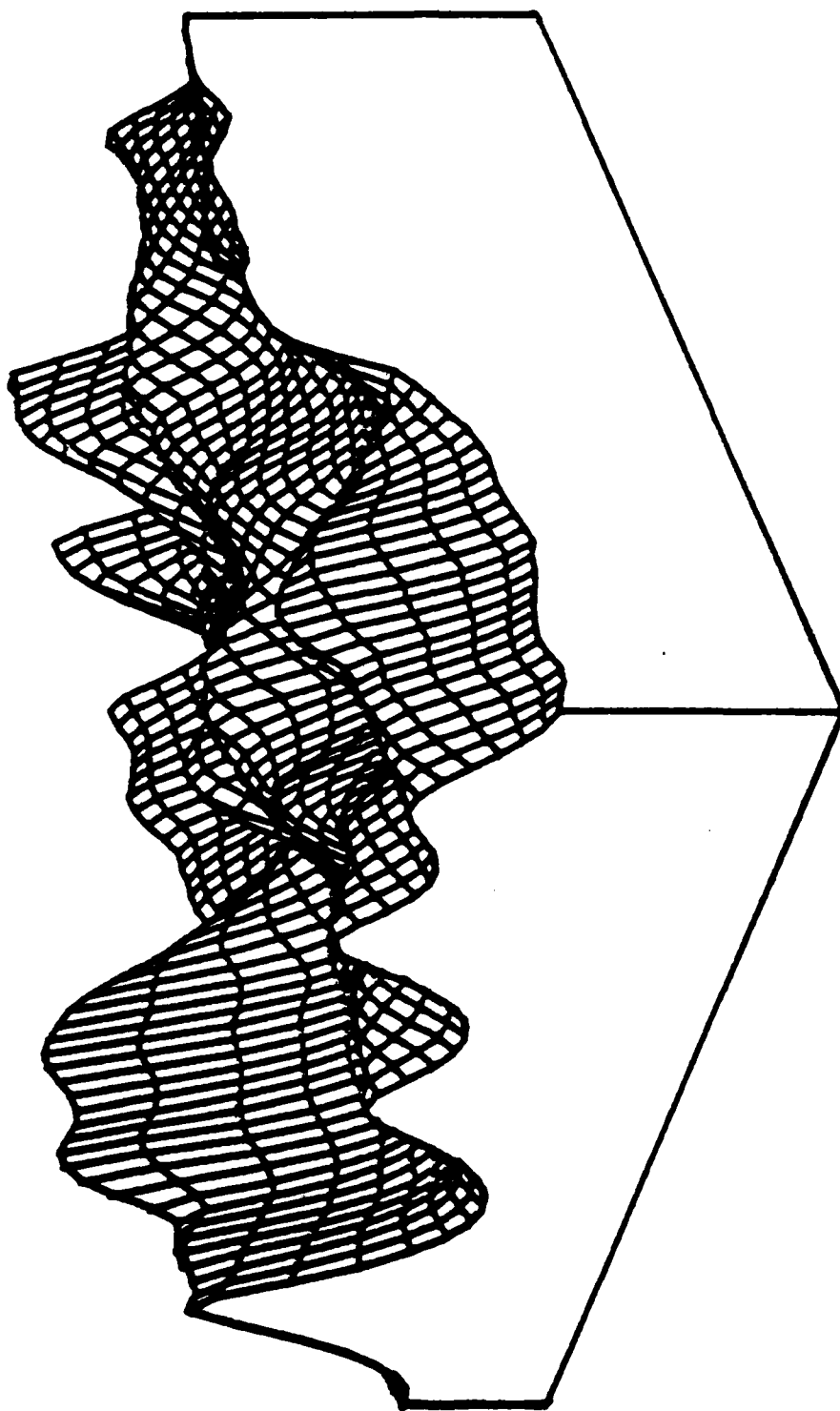


Figure IV-39.8 "FIVE" + noise

V. Recomendations

The areas for future research which either use the RLPC vocoder system or portions of the system include the following:

1. Formal listening tests (including rhyme tests) could be performed to arrive at a quantitative measure of how the rate of updating the synthesis filter affects the quality of speech. The use of a variable rate system could be investigated with the intent of finding a method that would determine when it is best to transmit data.

2. A method to perform spectral distance tests could be created. This would be an additional measure to help judge the differences between the RLPC system and other LPC systems. Differences less than 3dB have been shown to represent a very small variation between systems.

3. Additional methods for determining gain information could be investigated. The method used in this thesis takes the energy in the error signal to find the gain, possibly a better method exists.

4. The recursive method for performing the autocorrelation could be used for LPC analysis or formant estimation (for example in speaker verification system). Typically the large amount of computation for on-line implementations is a limiting factor, the recursive method could be a partial solution to this problem. The recursive architecture would work well with a VLSI circuit realization which might work well in a speaker verification system.

Bibliography

1. Atal, B.S. and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Am. 50:637-655, (August 1971).
2. Barnwell, Thomas P., "Recursive Windowing for Generating Autocorrelation Coefficients for LPC Analysis," IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-29, (October 1981).
3. Beek, Bruno, Edward Newberg, and David Hodge. "An Assessment of the Technology of Automatic Speech Recognition for Military Applications," IEEE Transactions on Acoustics, Speech, and Signal processing, ASSP-25:310-322, (August 1977).
4. Cheng, M.J., C.A. McGonegal, L.R. Rabiner, and A.E. Rosenberg "A comparative Performance Study of Several Pitch Detection Algorithms," IEEE Trans. Acou., Spch., Sig. Pros., (October 1976).
5. Kelton, W. David and Averill M. Law simulation Modeling and analysis, McGraw-Hill, 1982.
6. Kinderman, A.J. and J.G. Ramage, "Computer Generation of Normal Random Variables," Journal of the American Statistical Association, 71, (December 1976).
7. Makhoul, John, "Linear Prediction: A Tutorial Review," Proceedings of the IEEE, 63, (April 1975).
8. Makhoul, John and Lynn K. Cosell, "Adaptive Lattice Analysis of speech," IEEE Transactions on Circuits and Systems, 6, (June 1981).
9. Markel, J.D. and A.H. Gray, Jr Linear Prediction of Speech, Springer-Verlag, New York, 1976.
10. Rabiner L.R. and R.W. Schafer, Digital Processing of Speech Signals, Englewood Cliffs; Prentice-Hall, 1978.
11. Rosenberg A.E., "Effect of Glottal Pulse Shape on the Quality of Natural Vowels," J. Acoust. Soc. Am., 49:583-590, (February 1971).
12. Viswanathan, R. and J. Makhoul, "Quantization Properties of Transmission Parameters in Linear Predictive Systems," IEEE Transactions on Acoustics, Speech, and Signal processing, ASSP-23:309-321, (June 1975).

APPENDIX A

Software

The software system is designed to be versatile thereby allowing the user a great deal of flexibility for experimentation. The overall system design is shown in figure A-1. The system has three major programs including the pitch detector, a speech analyzer and a speech synthesizer. Additionally, there are some utility programs. Each of the main programs and the set of utility programs will be described separately as sections. For details on using the programs or substituting files see the users manual in appendix C.

Pitch detectors

As described in chapter III two pitch detecting algorithms were chosen. The first to be described is the Center Clipping Autocorrelation Pitch Detector (AUTOC). Basically the software structure shown in figure A-2 closely parallels the block diagram of the pitch detector in figure III-1.

From figure A-1 it can be seen that the pitch detector uses two input files, one output file and is run from the console as an autonomous unit. The first input file contains initializing values for the various subroutines.

These values are the clipping level and the voiced/unvoiced threshold. The other input file is a binary speech file with two's complimentary numbers. The output file contains a list of records where one record of pitch information is created for each 80 input speech samples. Each pitch record contains a "0" if the current frame is unvoiced and a "0" for the pitch period. For a voiced frame the record contains a "1" for voiced and the current pitch period in number of sampling rate periods (1/8000 Hz). Additionally a silence indicator is set to "1" for speech (as opposed to silence). For silence the record will consist of three "0"'s.

The logical program flow is shown in figure A-2. After the system is initialized a segment of speech is read in. The subroutine ENERGY computes the energy in each frame and compares it with an unvoiced/voiced threshold. If the energy is greater than the threshold, the frame is considered voiced, otherwise it is unvoiced. Next the subroutine SETMAX finds the largest absolute maximum value in the first third and the last third of a frame. The smaller of the two values is sent to the next subroutine CLPPER. CLPPER multiplies this value by a constant (a typical value is .68) and the result is called the clipping level. Any value between + and - the clipping level is set to 0.0 . Anything larger or smaller than + or - the clipping level is set to 1.0 or -1.0 respectively. This tri-level output sequence, created by CLPPER, is the input

to subroutine AUTCOR which performs a short time autocorrelation computation on the sequence. The autocorrelation is performed using an 80 point sequence shifted over the frame with a maximum of 160 shifts. Finally subroutine AUTCOR finds the first autocorrelation peak greater than 80% of $R(0)$. The appropriate information (voiced/unvoiced etc.) is output to the pitch file and the whole process is started over if more speech is to be processed.

The second pitch detector has been thoroughly described by (Ref 9) with one exception, a silence detector was added. In the same fashion as the other pitch detector the energy in the frame is determined and if it exceeds the silence threshold then it is speech. If not it is silence. To be compatible with the overall system the input and output files are identical with the previous pitch detector.

Speech analyzer

The speech analyzer has one input file, one output file and is run as an autonomous unit from the console. The input file contains initializing values and the output file is a set of predictor coefficients. The flow chart of the speech analyzer is shown in figure A-3.

Four values are used from the initialization file. The first real value is the filter shape coefficient. This is the value described in chapter II and it determines the window shape of the recursive autocorrelator (with a typical

value of .98). The second real value allows the program user to scale the input speech (a typical value is 1.0). The first integer variable determines the separation (in sample points) of filter coefficients that are output. For example, if the variable is 80 then a set of filter coefficients is written to the output file every 80 points. The second integer variable is a "1" if the user wishes to have the speech pre-emphasized, otherwise a "0".

Initially one block (256 integer values) of speech is read in. Subroutine RLPC recursively computes the autocorrelation values for each speech sample in the block. Subroutine INVFIL uses the autocorrelation values and calculates a set of LPC predictor coefficients. At the interval determined by integer filter separation variable the filter coefficients are written to the output file. Then another block of speech is read in. This process continues until there is no more speech.

Speech synthesizer

The speech synthesizer has three input files and one output file and is run as an autonomous unit from the console. The first input file contains initializing values, the second is a file of LPC predictor coefficients, and the third is a file of pitch data. The synthesized speech is written to the output file. The flow chart of the speech synthesizer is shown in figure A-4.

First, subroutine IOF reads in the input and output file

names. Three values are used from the initialization file. The first value is a real number that allows the user to change the emphasis on the voiced and unvoiced input to the digital filter synthesizer. The next two initialization values are integers. The first is the separation between frames and the second is a "1" if de-emphasis is or "0" if not.

If the current segment of speech to be produced is voiced the subroutine VOICED is called. VOICED produces a sequence $u(n)$ (the glottal pulse) that is as long as the current pitch period. The current pitch period (at the current frame's beginning) was found using linear interpolation. If current segment is unvoiced then subroutine UNVOCD is called. UNVOCD produces a normally distributed random sequence $u(n)$ 80 samples long.

Whether voiced or unvoiced the sequence $u(n)$ is passed to subroutine THROAT. This subroutine uses $u(n)$ as an input to a direct form digital filter. The coefficients of the filter come from the LPC predictor coefficients found in the program CODER. The user has the option of how often to update the coefficients and based on this the filter coefficients are asynchronous with the pitch. The output sequence of the filter is the synthesized speech. If selected the speech is de-emphasized and then with or without de-emphasis the speech is written to the output filter. If more speech is to be produced the process starts all over by checking for voiced or unvoiced, otherwise

processing stops.

Utility software

Five utility programs were created to allow the user to look at a variety of outputs and put them in a useful form. The primary use of the programs are stated below:

The first program SCALE1 takes the vocoder output and scales the speech to the proper magnitude for the D/A converter.

The second program PLTTER either plots the pitch detector output or a speech file.

The third program SETUP allows the user to update a file of parameters that are the initialization values for the vocoder.

The fourth program KEVAL lets the user create a file which is the input to a 3-D plotter.

The final program FORMANT allows the user to create an artificial vowel using poles or a set of filter coefficients.

SCALE 1 requires one input file and one output file. The input file is any binary (blocked) speech and the output file becomes a scaled speech file with a maximum of 2000.0. One additional option allows uniform noise to be added to the speech file. The speech file, plus noise, is still scaled to a maximum of 2000.0. The maximum value for the noise is input at the console and actually scales the uniform (-0.5 to 0.5) density function. For example, if 500.0 is entered at the console the noise falls between

-250.0 and 250.0 inclusively. The subroutine DRAND is called to find the random number. This program or an equivalent one must be used if the user is going to output speech on the D/A converter. The D/A converter only allows numbers to vary between -2048 and 2047 so SCALE1 meets this requirement.

The program PLTTER requires one input file. The file name is entered at the console and this file can be either the file output from the pitch detector for a pitch plot or a speech file for a plot of speech. If a pitch plot is required two plots are output on the PRINTRONIX printer. The top plot is a plot of the pitch period and the bottom is a plot of silence (a "1") or speech (a "0"). The second possible output is a plot of the speech file. Each page has ten plots of speech except the last page which has the required number of plots to plot of the remaining speech. The only subroutine called PLOT10 which is available in the Data General system. If the user requires constant scales on plot output two statements must be added to the systems PLOT10. The two statements follow the line of code "CALL ASCALE (Y,..." and are as follows:

FY=-3000.0

DY= 3000.0

These set the scales to a maximum value of 3000.0. PLTTER automatically scales the speech to a maximum value of 3000.0.

SETUP requires one input file and either uses the input

file as an output file or uses an additional file for output. The purpose of this program is to allow the user to change the initial conditions of the vocoder by changing the initial conditions file. The input file is a file already containing initial conditions. If a new file is to be created the program sets all values to zero and then the user inputs initial conditions from the console. If the input file contains initial conditions the user is allowed to change any value. A list of the initial conditions can be output to the PRINTRONIX printer and/or the console. Finally the initial conditions can overwrite the input file or be put in a new file. No subroutines are called by this program.

KEVAL has one input file and if the option is selected one output file. The input file is a file of predictor coefficients output from the vocoder. The first value in the file is the order (p) of the filter and the remaining values are sets of predictor coefficients of length p . One of three possible outputs are available each time the program is run. Basically different methods of viewing how predictor coefficients vary was created. The first method allows the user to select a sequence of predictor coefficients which KEVAL uses to find a log magnitude plot for each coefficient set. These log magnitude plots are output to a file. This file is formatted to be input to the program PLOT3D which is on the Data General system. PLOT3D then is used to make a 3-D formant plot. The second type of

output allows the user to plot a series of formant plots on the PRINTRONIX printer. The third type of output allows the user to use the Tektronix 4010-1 terminal to create linear magnitude plots, log magnitude plots, phase plots and/or impulse response plots for subsequent predictor coefficient sets. The required system subroutines include PLOT10 for Tektronix plots and PLOT5 is needed for Printronix plots.

FORMANT is a program that was developed originally to create a file which is the output of an artificial vowel that is impulsed at a constant frequency. An input file can be used to either enter the vowels' filter coefficients in direct form or as poles and zeros. If the input is a file of poles and zeros then these are converted to coefficients in the direct form using the subroutines EXPAND and POLYMLT. If input files are not available the user can enter either pole and zeros or direct form filter coefficients through the console. If the user wants to use the same predictor coefficients later they can be saved in an output in the direct form. Formant impulses the digital filter (which uses the predictor coefficients) at the impulsed frequency. The output from this process is then written to the output file. The user can then have the impulse response plotted using the system routine PLOT10.

In conclusion, each of the utility programs provide various ways to make information from the vocoder more useful.

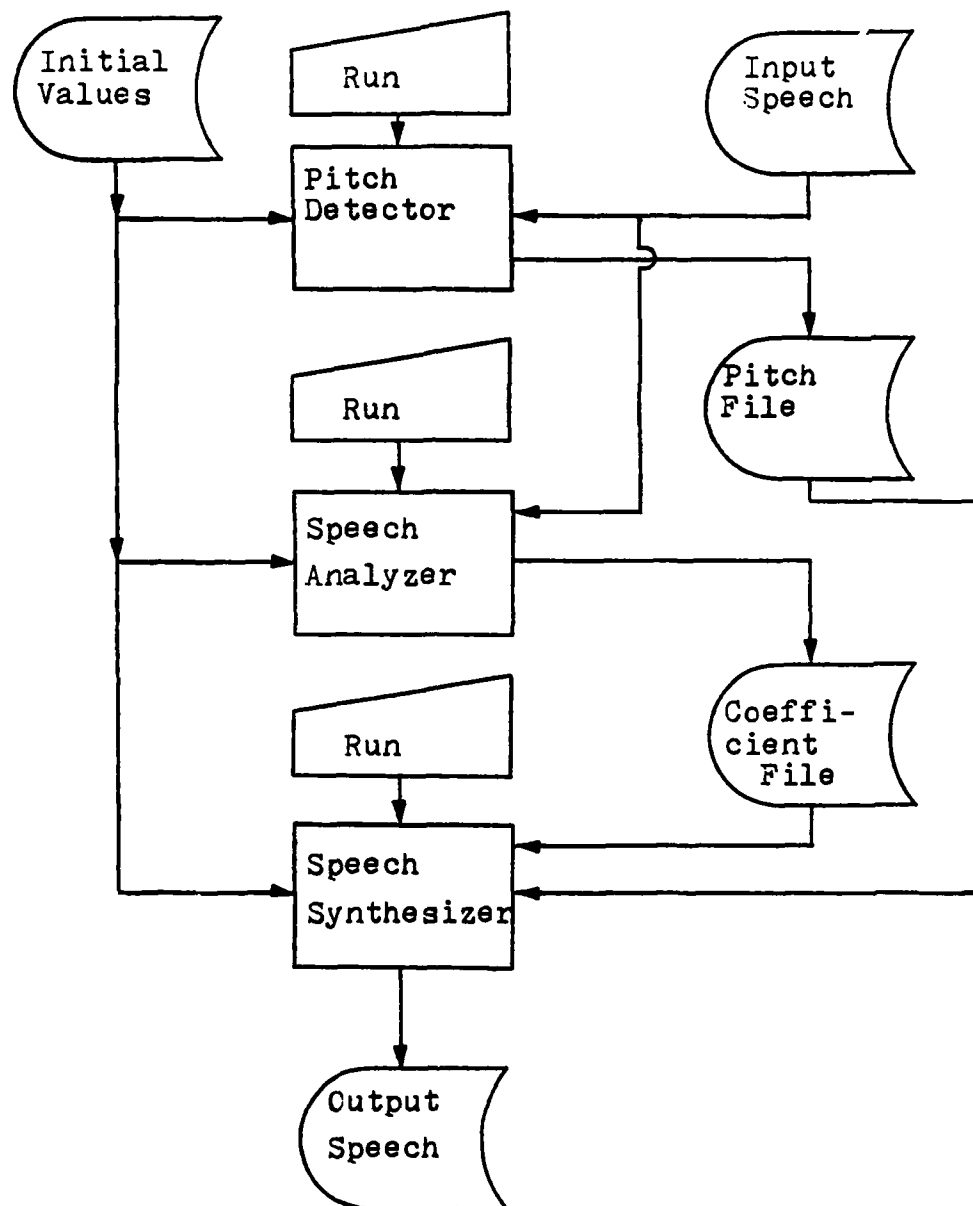


Figure A-1 I10. Block Diagram of Vocoder

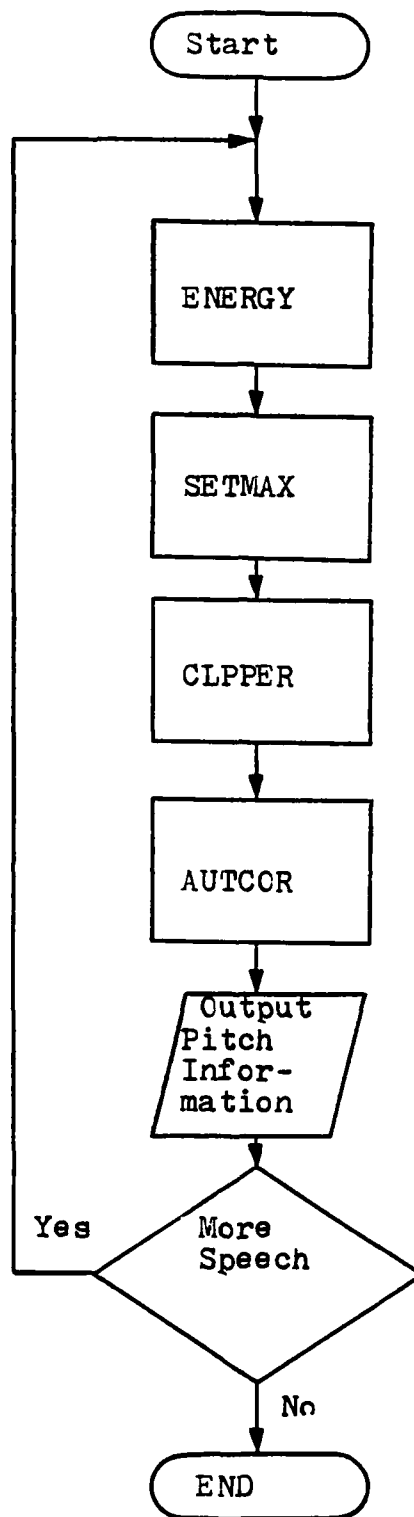


Figure A-2 Flowchart of PITCH

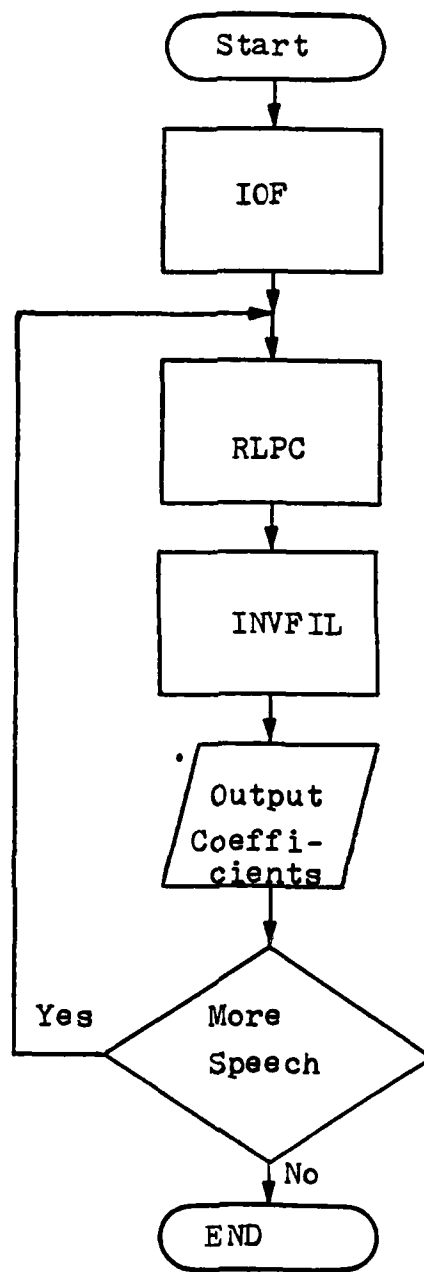


Figure A-3 Flowchart of Coder

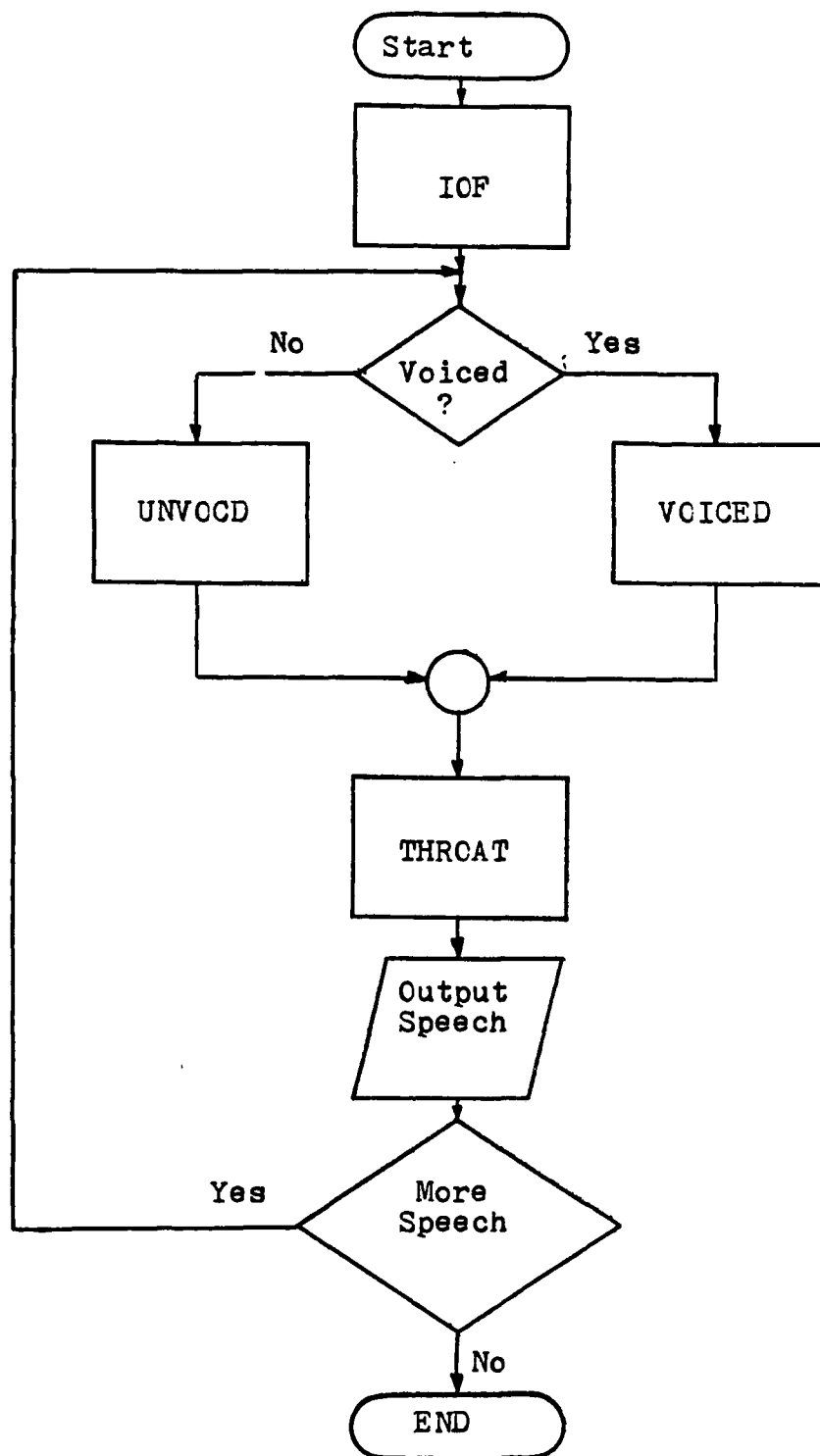


Figure A-4 Flowchart of SYNTH

APPENDIX B

This appendix contains listings of all the software used in the vocoder including utility software. What follows is a table of contents for the software.

Contents

	<u>Page</u>
Speech Analyzer (CODER)	B- 3
CODER	- 3
IOF	- 6
RLPC.	- 7
INVFIL.	- 9
Speech Synthesizer (SYNTH)	-10
SYNTH	-10
VOICED.	-18
THROAT.	-14
UNVOCD.	-16
IOF	- 6
Pitch Detector (SIFT)	-20
SIFT.	-20
SIFTA	-23
DIRECT.	-28
AUTO.	-27
ENER.	-26
IOF	- 6
Pitch Detector (PITCH2)	-29
PITCH2.	-29
IOF	- 6
LPF	-34
ENERGY.	-36
SETMAX.	-38
CLPPER.	-39
AUTCOR.	-40

Utility Programs	-42
SCALE1.	-42
PLTTER.	-44
SETUP	-46
KEVAL	-51
FORMANT	-56
EXPAND.	-62
POLYMLT	-63

```

C*****
C
C PROGRAM:      CODER
C AUTHOR:       WILL JANSSEN
C DATE:         6 JULY 83
C LANGUAGE:     FORTRAN5
C FUNCTION:     THIS PROGRAM INPUTS A SPEECH FILE (NO PARTICULAR
C               LENGTH REQUIRED) AND PRODUCES A FILE OF LPC
C               COEFFICIENTS.
C
C LOAD LINE:    RLDR CODER RLPC INVFIL IOF @FLIB@
C RUN LINE:     CODER S1/I PRAM/I COEF/I
C INPUTS:       FILSPI-IS THE FIRST INPUT FILE WHICH IS AN INTEGER
C               SPEECH FILE (BINARY)
C               PARAM-2ND INPUT FILE (OUTPUT OF SETUP) CONTAINING 25
C               RECORDS OF REAL DATA THEN 25 RECORDS OF INTEGER DATA.
C               THIS DATA IS USED TO INITIALIZE VARIOUS PARTS OF THIS
C               PROGRAM (RECOMEND USING UTILITY PROGRAM SETUP TO INITIALIZE
C               THIS FILE). DATA FROM THIS FILE USED IN THIS PROGRAM:
C               RECORD#4-RPARAM(4)=>RLPC WINDOW SHAPE (TYPICALLY .98)
C               RECORD#5-RPARAM(5)=>SPEECH SCALING FACTOR (TYP. 1.0)
C               RECORD#27-IPARAM(2)=># OF POINTS BETWEEN OUTPUT LPC
C               FILTER COEFFICIENTS.
C               RECORD#28-IPARAM(3)=>(1-USE PRE-EMPHASIS, 0-DON'T)
C OUTPUT:       COFA-THIS FILE IS A BINARY FILE CONTAINING LPC PREDICTOR
C               COEFFICIENTS WHERE THE FIRST VALUE IS THE FILTER ORDER
C               (VARIABLE NORDER). THE REST OF THE FILE IS ORGANIZED AS
C               SETS OF LPC FILTER COEFFICIENTS (NORDER OF THEM PER SET)
C               WHERE THE 0TH POSITION IS ASSUMED TO EQUAL 1.0 .
C*****

```

```

C*****
      DIMENSION SP1(-80:256), COEF(256,0:10), SP1OLD(2),
X      Q(256), RPARAM(25), IPARAM(25)
      INTEGER FILUFD(18), SPEECH(256), FILSPI(7), PARAM(7), COFA(7),
X      AUTGAN(7)
      DOUBLE PRECISION W(0:3,0:10)

```

```

C
C***READ INPUT AND OUTPUT FILES FOR DATA
C

```

```

      IENDS = 0
      NUMB = 0
      SP1OLD(1) = 0.0
      SP1OLD(2) = 0.0
      NFILES = 3
      CALL IOF(NFILES, MAIN, FILSPI, PARAM, COFA, AUTGAN,
X      MS, S1, S2, S3, S4)
      CALL OPEN(1, FILSPI, 1, IER)
      IF(IER .NE. 1) TYPE"OPEN ERROR ", IER
      MBLOCKS = 1 ; SET FOR READING ONE BLOCK AT A TIME
      CALL OPEN(2, PARAM, 1, IER)
      IF(IER .NE. 1) TYPE"OPEN ERROR ON", PARAM(1), IER
      DO 10 I=1,25
        READ FREE(2) RPARAM(I)
10      CONTINUE
      DO 11 I=1,25
        READ FREE(2) IPARAM(I)
11      CONTINUE
      CALL CLOSE(2, IER)
      IF(IER .NE. 1) TYPE"CLOSE FILE ", PARAM(1), IER
      CALL DFILW(COFA, IER)

```

```

        IF(IER .EQ. 13)GO TO 60
        IF(IER .NE. 1)TYPE"DELETE ERROR", IER
60      CALL CFILW(COFA, 2, IER)
        IF(IER .NE. 1)TYPE"CREATE FILE ERROR ", IER
        CALL OPEN(3, COFA, 3, IER)
        IF(IER .NE. 1)TYPE"OPEN ERROR ON", COFA(1), IER
C
C***ZERO SPEECH ARRAY FOR FIRST RUN
C
        ISAV = 80
        ITOT = ISAV + 257
        IS = ISAV + 1
        DO 100 KN=1, ITOT
            I1 = KN - (ISAV + 1)
            SP1(I1) = 0.0
100      CONTINUE
        A = RPARAM(4) ; FILTER SHAPE COEFFICIENT
        SSCALE = RPARAM(5) ; SET SCALE FACTOR TO REDUCE SPE. MAG.
        NPOINS = IPARAM(2) ; FILTER POINTS SEPERATION
        IPRE = IPARAM(3)
        NCUR = 1
        NORDER = 10 ; ORDER OF THE FILTERS
C
C***PASS FILTER ORDER TO SYNTH
C
        WRITE BINARY(3) NORDER
C
C*** ZERO FILTER FOR FIRST RUN
C
        DO 140 I=0, 3
            DO 140 I1=0, 10
                W(I, I1) = 0.000
140      CONTINUE
C
C***PROCESS THE DATA
C
        NV = 0
145      CONTINUE
        IF(NV .EQ. 0)GO TO 175 ; SKIP FIRST TIME
        DO 150 NT=1, ISAV ; SAVE LAST 80 SPEECH VALUES
            NT1 = NT - ISAV
            NT2 = NT + 256 - ISAV
            SP1(NT1) = FLOAT(SPEECH(NT2))/SSCALE ; SCALE SPEECH TO MAX 2
150      CONTINUE
C
C***READ IN SPEECH
C
175      CALL RDBLK(1, NV, SPEECH, MBLOCKS, IENDS)
        IF((IENDS .NE. 1) .AND. (IENDS .NE. 9))TYPE"READ BLOCK ERROR ", IENDS
        DO 200 I=1, 256
            SP1(I) = FLOAT(SPEECH(I))/SSCALE
200      CONTINUE
C
C      PRE-EMPHASIZE THE SPEECH
C
        IF(IPRE .EQ. 0)GO TO 310
        DO 300 I=-75, 256
            SP1OLD(1) = SP1(I)
            SP1(I) = SP1(I) - .9 * SP1OLD(2)
            SP1OLD(2) = SP1OLD(1)

```

```

300    CONTINUE
310    CONTINUE
      CALL RLPC(SP1,NORDER,W,A,COEF,G)
C
C***WRITE PREDICTOR COEFFICIENTS THEN GAIN TO COEFFICIENT FILE FOR EACH FRAME
C
500    IF(NCUR .GT. 256)GO TO 1000
      DO 900 KB=1,NORDER
        WRITE BINARY(3) COEF(NCUR,KB)
900    CONTINUE
      WRITE BINARY(3) G(NCUR)
      NCUR = NCUR + NPOINS
      GO TO 500
1000   CONTINUE
      NCUR = NCUR - 256
      NUMB = NUMB +256
C      TYPE"JUST COMPLETED  ",NUMB,"  POINTS"
      NV = NV + 1
      IF(IENDS .NE. 9)GO TO 145
9000   CONTINUE
9500   CONTINUE
      CALL CLOSE(1,IER)
      IF(IER .NE. 1)TYPE"CLOSE ERROR ON FILSPI ",FILSPI(1),IER
      CALL CLOSE(3,IER)
      IF(IER .NE. 1)TYPE"CLOSE ERROR ON COFA ",COFA(1),IER
      CALL CLOSE(4,IER)
      STOP
      END

```



```

SUBROUTINE IOF(N,MAIN,F1,F2,F3,F4,MS,S1,S2,S3,S4)
C*****
C   ADAPTED FROM SUBROUTINE WRITTEN BY LT SIMMONS 10 SEPT 81
C
C   THIS FORTRAN 5 SUBROUTINE WILL READ FROM THE FILE
C   COM.CM (FCOM.CM IN THE FORE GROUND) THE PROGRAM NAME,
C   ANY CLOBAL SWITCHES, AND UP TO THREE LOCAL FILE
C   NAMES AND CORRESPONDING LOCAL SWITCHES.
C
C   ARGUMENTS:
C
C   N IS THE NUMBER OF LOCAL FILES AND SWITCHES TO BE
C   READ FROM (F)COM.CM. N MUST BE 1, 2, OR 3.
C
C   MAIN IS AN ASCII ARRAY FOR THE MAIN PROGRAM FILE NAME.
C
C   F1, F2, F3, AND F4 ARE THE THREE ASCII ARRAYS TO RETURN
C   THE LOCAL FILE NAMES.
C
C   MS IS A TWO-WORD INTEGER ARRAY THAT HOLDS ANY GLOBAL
C   SWITCHES.
C
C   S1, S2, S3, AND S4 ARE TWO-WORD INTEGER ARRAYS THAT
C   HOLD THE LOCAL SWITCHES CORRESPONDING TO F1 THROUGH
C   F4 RESPECTIVELY.
C*****
C
C   DIMENSION
C
C   DIMENSION MAIN(7),MS(2)
C   INTEGER F1(7),F2(7), F3(7), F4(7),S1(2),S2(2),
X S3(2),S4(7)
C
C   CHECK BOUNDS ON N
C
C   IF(N.LT.1.OR.N.GT.4)STOP ;N OUT OF BOUNDS
C
C   PROCESS THE DATA IN (F)COM.CM
C
C   CALL GROUND(I) .FIND OUT WHICH GROUND PROGRAM IS IN
C   IF(I.EQ.0)OPEN 0,"COM.CM" ;OPEN CH. 0 TO COM.CM
C   IF(I.EQ.1)OPEN 0,"FCOM.CM" ;OPEN CH. 0 TO FCOM.CM
C   CALL COMARG(0,MAIN,MS,IER) ;READ FROM (F)COM.CM
C   IF(IER.NE.1)TYPE" COMARG ERROR: ",IER
C   WRITE(10,1)MAIN(1) ;TYPE PROGRAM NAME
1  FORMAT(' PROGRAM ',S13,'RUNNING.')
C   CALL COMARG(0,F1,S1,JER) ;READ FROM (F)COM.CM
C   IF(JER.NE.1)TYPE" COMARG ERROR (F1): ",JER
C   IF(N.EQ.1)GO TO 2 ;TEST N
C   CALL COMARG(0,F2,S2,KER) ;READ FROM (F)COM.CM
C   IF(KER.NE.1)TYPE" COMARG ERROR (F2): ",KER
C   IF(N.EQ.2)GO TO 2 ;TEST N
C   CALL COMARG(0,F3,S3,LER) ;READ FROM (F)COM.CM
C   IF(LER.NE.1)TYPE" COMARG ERROR (F3): ",LER
C   IF(N.EQ.3)GO TO 2 ;TEST N
C   CALL COMARG(0,F4,S4,NER) ;READ FROM (F)COM.CM
C   IF(NER.NE.1)TYPE" COMARG ERROR (F4): ",NER
2  CLOSE 0
RETURN

```

```

SUBROUTINE RLPC(SPI, NORDER, W, A, COEF, G)
C*****
C
C THIS SUBROUTINE CALCULATES THE RECURSIVE AUTOCORRELATION FOR
C EACH FRAME WHERE THE FRAME CAN VARY FROM ONE SAMPLE TO 80
C SAMPLES / FRAME
C
C INPUTS: SPI -THIS IS THE INPUT SPEECH
C          NORDER-THIS IS THE ORDER OF THE RECURSIVE AUTOCORRELATION
C          SYSTEM
C          W -THIS IS THE INTERMEDIATE NODAL VALUES OF THE
C          RLPC FILTER
C          A -THIS IS ALPHA THE VALUE OF THE WINDOW SHAPE
C
C OUTPUT: COEF -THIS IS A SET OF PREDICTOR COEFFICIENTS THAT
C          CUME FROM A SUBROUTINE CALL TO INVFIL
C          G -THIS IS THE GAIN ALSO FROM INVFIL
C*****
C
C DIMENSION SPI(-80:256), G(256), COEF(256, 0:10),
C X T(11), COE(0:10)
C DOUBLE PRECISION R(0:10), W(0:3, 0:10), TEMP1, TEMP2, S(0:10)
C TYPE"RUNNING RLPC"
C A0 = 1.0
C A1 = 3.0 * (A**2)
C A2 = -3.0 * (A**4)
C A3 = A**6
C
C C***PROCESS 256 SPEECH SAMPLES THROUGH THE FILTER
C
C DO 1500 N=1,256
C
C C***SET UP DELAY PRODUCTS INTO FILTER
C
C DO 500 I=0,NORDER
C S(I) = DBLE(SPI(N-I) * SPI(N)) ; INPUT TO FILTERS
C 500 CONTINUE
C
C C***CALCULATE R(K) FOR EACH FILTER
C
C DO 600 K=0,NORDER
C W(0,K) = A1*W(1,K)+A2*W(2,K)+A3*W(3,K)+S(K)
C TEMP1 = (K+1) * (A**K) * W(1,K)
C TEMP2 = (K-1) * (A**(K+2)) * W(1,K)
C R(K) = TEMP1 - TEMP2
C TYPE"R=",R(K), " N= ",N, " K= ",K
C 600 CONTINUE
C DO 650 KS=1,NORDER
C R(KS) = R(KS)/R(0)
C 650 CONTINUE
C ROS = SNGL(R(0))
C R(0) = 1.0
C ACCEPT"NEXT?", NYZKJH
C
C C*** PREPARE W FOR NEXT AUTOCORRELATION
C
C DO 700 K=0,NORDER
C W(3,K) = W(2,K)
C W(2,K) = W(1,K)
C W(1,K) = W(0,K)

```

HD-A138 008

A RECURSIVE LINEAR PREDICTIVE VOCODER(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
W A JANSSEN DEC 83 AFIT/GE/EE/83D-33

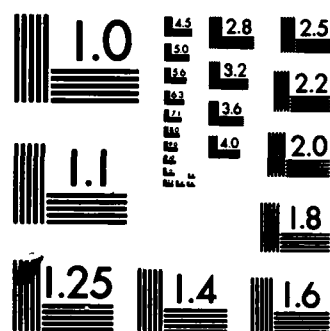
4/4

UNCLASSIFIED

F/G 17/2

NL

END



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

      W(O,K) = 0 0
700  CONTINUE
      CALL INVFIL(R,NORDER,COE,GF)
      G(N) = SQRT(KOS * GF)/100000.0
      DO 725 IWHY=0,NORDER
        COEF(N,IWHY) = COE(IWHY)
725  CONTINUE
      DO 5000 IB=0,NORDER
        X      IB1 = IB + 1
        X      T(IB1) = SNGLR(IB)
X5000 CONTINUE
      X      MODE = 1
      X      NG = 1
      X      NBUT = NORDER + 1
      X      IFSC = 0
      X      ACCEPT"DO NEXT ONE?",NY
      X      CALL GRPH2("AUTCOR",NG,T,U,NBUT,MODE,YM,YA,IFSC)
      X      TYPE"N = ",N
1500 CONTINUE
      RETURN
      END

```

```

C*****
C
C   THIS SUBROUTINE IS THE INVERTER PORTION OF THE SUBROUTINE
C   AUTO AS PRESENTED IN MAKEL & GRAY
C
C   INPUTS :R5      -THIS IS A SET OF AUTOCORRELATION VALUES PASSED
C                   FROM THE SUBROUTINE RLPC
C                   NORDER-THIS IS THE ORDER OF THE RLPC SYSTEM
C
C   OUTPUTS:COE     -THIS IS A SET OF PREDICTOR COEFFICIENTS THAT ARE
C                   FOUND BY INVERTING THE AUTOCORRELATION MATRIX
C                   GF      -THE CURRENT GAIN OF THE SYSTEM
C*****

```

```

      SUBROUTINE INVFIL(R5,NORDER,COE,GF)

      DIMENSION COE(0:10)
      DOUBLE PRECISION R5(0:10),R(11),RC(21),A(20),S,ALPHA,AT

C
      M = NORDER
      MP=M+1
      DO 15 K=1,MP
        R(K) = R5(K-1)
15    CONTINUE
      X    WRITE(12,999)(R(I),I=1,4)
      X    WRITE(12,998)(R(I),I=5,MP)
      X998  FORMAT(5X,6(G16.8,1X))
      X999  FORMAT(1X,6(G16.8,1X))
      RC(1)=-R(2)/R(1)
      A(1)=1.
      A(2)=RC(1)
      ALPHA=R(1)+R(2)*RC(1)
      DO 40 MINC=2,M
        S=0.000
        DO 20 IP=1,MINC
          S=S+R(MINC-IP+2)*A(IP)
20    CONTINUE
        RC(MINC)=-S/ALPHA
        MH=MINC/2+1
        DO 30 IP=2,MH
          IB=MINC-IP+2
          AT=A(IP)+RC(MINC)*A(IB)
          A(IB)=A(IB)+RC(MINC)*A(IP)
          A(IP)=AT
30    CONTINUE
        A(MINC+1)=RC(MINC)
        ALPHA=ALPHA+RC(MINC)*S
        IF(ALPHA)50,50,40
40    CONTINUE
50    DO 500 NT=0,NORDER
        COE(NT) = SNGL(A(NT+1))
500  CONTINUE
      X    WRITE(12,1001)
      X    WRITE(12,1002)(COE(I),I=0,4)
      X    WRITE(12,1002)(COE(I),I=5,M)
      X1002 FORMAT(1X,6(D16.8,1X))
      X1001 FORMAT(1X,"ALPHAS ")
      GF = SNGL(ALPHA)
      RETURN

```

```

C*****
C
C PROGRAM: SYNTH
C AUTHOR: WILL JANSSEN
C DATE: 12 JULY 83
C LANGUAGE: FORTRAN
C FUNCTION: THIS PROGRAM INPUTS A PREDICTOR COEFFICIENT
C FILE AND PITCH INFORMATION TO SYNTHESIZE
C SPEECH.
C
C LOAD LINE: RLD R SYNTH VOICED THROAT UNVOICED @FLIB@
C RUN LINE: SYNTH PRAM/I COEF/I SPE/O SNOUT/I
C INPUTS: PARAM-1ST INPUT FILE (OUTPUT OF SETUP) CONTAINING
C 25 RECORDS OF REAL DATA THEN 25 RECORDS OF INTEGER
C DATA. THIS DATA IS USED TO INITIALIZE VARIOUS PARTS
C OF THIS PROGRAM (RECOMEND USING UTILITY PROGRAM SETUP
C TO INITIALIZE THIS FILE). DATA FROM THIS FILE USED IN
C THIS PROGRAM:
C RECORD#7-RPARAM(7)=>RATIO OF UNVOICED TO VOICED IMPULSE
C (OPTIMUM AROUND .1)
C RECORD#8-RPARAM(8)=>POS. SLOPE PORTION OF GLOTTAL PULSE
C (EX. .1 WOULD BY 1/10 OF CURRENT
C PITCH PERIOD)
C RECORD#9-RPARAM(9)=>NEG. SLOPE PORTION OF GLOTTAL PULSE
C (UNITS SAME AS ABOVE)
C RECORD#27-IPARAM(2)=>NUMBER OF POINTS BETWEEN FRAMES
C RECORD#28-IPARAM(3)=>(1-USE DE-EMPHASIS, 0-DON'T)
C RECORD#29-IPARAM(4)=>(1-USE GLOTTAL PULSE, 0-IMPULSE)
C
C COFA-2ND INPUT FILE IS A BINARY FILE (OUTPUT OF CODER) WHERE
C THE FIRST VALUE IS THE FILTER ORDER (VARIABLE NORDER).
C THE REST OF THE FILE IS ORGANIZED AS SETS OF LPC FILTER
C COEFFICIENTS (NORDER OF THEM PER SET) WHERE THE 0TH POSITION
C IS ASSUMED TO EQUAL 1.0 .
C
C PTCHIF-3RD INPUT FILE (OUTPUT OF PITCH) CONTAINING 3 VALUES
C PER RECORD (SEE LINE 470 FOR THE FORMAT)
C VOICD2-THE FIRST VALUE (1-FOR VOICED SPEECH, 0-FOR UNVOICED
C IF NOT SILENCE)
C SIL2-THE SECOND VALUE (1-FOR SPEECH, 0-FOR SILENCE)
C PP2-THE THIRD VALUE IS THE PITCH PERIOD (IN SAMPLES)
C
C OUTPUTS: SPCH-IS THE OUTPUT FILE WHICH IS AN INTEGER FILE OF THE
C GENERATED SPEECH.
C
C*****
C DIMENSION RPARAM(25), IPARAM(25), SPEECH(256),
C X U(180), FILTER(20), S(180), W(0:20)
C INTEGER PARAM(7), COFA(7), PTCHIF(7), SPCH(7), VOICD2, VOICD1,
C X SIL1, SIL2, PP1, PP2, FRMPOS, PPPOS1, PPPOS2, PPF, FRMSIZ,
C X X(256), INTS(256)
C DOUBLE PRECISION IX
C DATA IS, IP, KS, KEND/1.0, 0.0, 0/
C
C***READ INPUT/OUTPUT FILE NAMES
C
C IX = DBLE(203)
C
C*** ZERO SYNTHESIS FILTER
C

```

```

DO 3 I=0.20
  W(I) = 0.0
3  CONTINUE
  ALPHA = 1000.0 ; OUTPUT GAIN VALUE
  NFILES = 4
  CALL IOF(NFILES,MAIN,PARAM,COFA,SPCH,PTCHIF,MS,S1,
X  S2,S3,S4)
  TYPE"PAST IOF"
  CALL OPEN(2,PARAM,1,IER)
  IF(IER.NE.1)TYPE"OPEN ERROR ON ",PARAM(1),IER
C
C*** INPUT INITIALIZING DATA
C
DO 10 I=1.25
  READ FREE(2) RPARAM(I)
10  CONTINUE
DO 11 I=1.25
  READ FREE(2) IPARAM(I)
11  CONTINUE
X  TYPE"READ IN PRAM"
  CALL CLOSE(2,IER)
  IF(IER.NE.1)TYPE"CLOSE ERROR ON ",PARAM(1),IER
  CALL OPEN(3,COFA,3,IER)
  IF(IER.NE.1)TYPE" OPEN ERROR ON ",COFA(1),IER
  CALL OPEN(4,PTCHIF,1,IER)
  IF(IER.NE.1)TYPE" OPEN ERROR ON",PTCHIF(1),IER
  CALL OPEN(1,SPCH,3,IER)
  IF(IER.NE.1)TYPE"OPEN ERROR ON ",SPCH(1),IER
  FRMSIZ = IPARAM(2) ; DETERMINE FRAME SIZE
  IDE = IPARAM(3) ; DE-EMPH IF EQ TO 1
  RATIO1 = RPARAM(7) ; RATIO OF UNVOCD TO VOICED IMPULSE
  IPUL = IPARAM(4)
  PER1 = RPARAM(8)
  PER2 = RPARAM(9)
C
DO 400 I=1.180 ; INITIALIZE THE SPEECH ARRAY
  S(I) = 0.0
  U(I) = 0.0
400 CONTINUE
C
  READ BINARY(3) NORDER ; DETERMINE FILTER ORDER
  READ(4,470) VOICD2,SIL2,PP2
470  FORMAT(3X,3(I10,3X))
  NVERIFY = 1
  LASTD = 0
  PPPOS2 = 1
  PPPOS1 = 1
  NPOS = 1
  PP1 = 0
  NBEGP = 1
  NFILP = 1
  NVOLD = 0
  S40 = 0.0
  S60 = 0.0
  S30 = 0.0
C
C***INITIALIZE FILTER
C
DO 490 K=1,NORDER
  READ BINARY(3)FILTER(K)

```



```

490     CONTINUE
      READ BINARY(3)G
C
C***START SPEECH LOOP
C
675     CONTINUE
C
C*** CHECK FOR PROPER PITCH BOUNDARY PLACEMENT
C
680     IF(PPPOS2 .GE. NBEOP) GO TO 700
      PPPOS1 = PPPOS2 ; UPDATE PITCH PERIOD BOUNDARY
      PP1 = PP2
      NVOLD = VOICD2      ; STORE OLD VOIC/UN SWITCH
      READ(4,470) VOICD2,SIL2,PP2
      PPPOS2 = PPPOS2 + 80      ; NOTE: THIS VALUE MAY BE GREATER THAN 256
      GO TO 680
700     CONTINUE
      IF((VOICD2 .EQ. 0).OR. (NVOLD.EQ. 0))      GO TO 800 ; VOICED/UNVOICED SWJ
      NDIF = PP2 - PP1      ; CALCULATE PITCH PERIOD FOR FRAME DEG
      NPOSDF =NBEOP - PPPOS1
      RATIO = FLOAT(NPOSDF)/80.0
      IF(NVERYF .EQ. 1)PP1 = PP2
      NVERYF = 0
      PPF = INT(RATIO * FLOAT(NDIF)) + PP1      ; INTERPOLATE PITCH
      CALL VOICED(U,PPF,ALPHA,IPUL,PER1,PER2)
      ISIZE = PPF      ; KEEPS TRACK OF PITCH PERIOD SENT TO THROAT
750     CONTINUE
      GO TO 900
800     CONTINUE
      ISIZE = 80
      CALL UNVOCD(U, ISIZE, ALPHA, IX, SIL2, RATIO1)
900     CONTINUE
      CALL THROAT(U, ISIZE, FILTER, NORDER, G, S, W, FRMSIZ, NBEOP,
X LASTD, NFILP, SIL2, KS)
C
C***OUTPUT SPEECH
C
      DO 970 MV=1, ISIZE
C
C*** DE-EMPHASIZE SPEECH
C
940     IF(IDE .EQ. 0)GO TO 950
      S40 = S(MV)
      S30 = S40 + .9 * S40
      IF((S30.LT..001).AND.(S30.GT.-.001))S30=0.0
      S40 = S30
      INTS(MV) = IFIX(S30)
      GO TO 960
950     INTS(MV) = IFIX(S(MV))
960     CONTINUE
970     CONTINUE
C
C*** WRITE OUT SPEECH
C
1000    IP = IP + ISIZE
      L = 1
      IF(IP .GE. 256)GO TO 1210 ; SPLIT S(I) & WRBLK(....,X,...)
      DO 1200 I=IS, IP
        X(I) = INTS(L)
        L = L + 1

```

```

1200 CONTINUE
GO TO 1240
1210 CONTINUE
IP = IP - 256 , RESET IP
DO 1220 I=IS, 256
X(I) = INTS(L) , LOAD UP X(I)
L = L + 1
1220 CONTINUE
CALL WRBLK(1, KS, X, 1, IER)
IF(IER .EQ. 9) GO TO 9000 , END OF FILE
IF(IER .NE. 1) TYPE "WRBLK ERROR ON FILE #2", IER
IF(IP .EQ. 0) GO TO 1230
DO 1230 I=1, IP
X(I) = INTS(L) , RESTART LOAD UP OF X(I)
L = L + 1
1230 CONTINUE
KS = KS + 1
1240 IS = IP + 1
IF(LASTD .NE. 1) GO TO 675 , ZERO OUT TO END OF FILE
IF(KS .GE. 98) GO TO 9000
DO 1300 I=IP, 256
INTS(I) = 0
1300 CONTINUE
IP = 1
GO TO 1220
C GO TO 675 , CONTINUE CREATING SPEECH
9000 CONTINUE
CALL DVDCHK(ICODE1)
IF(ICODE1 .EQ. 1) TYPE "DIVIDE BY ZERO OCCURRED"
CALL OVERFL(ICODE2)
IF(ICODE2 .EQ. 1) TYPE "OVERFLOW OCCURRED"
IF(ICODE2 .EQ. 3) TYPE "UNDERFLOW OCCURRED"
STOP
END

```

SUBROUTINE THROAT(U, ISIZE, FILTER, NORDER, G, S, W, FRMSIZ, NDEGP,
X LASTD, NFILP, NSIL1, IN)

```

C*****
C
C THIS SUBROUTINE INPUTS U (A SEQUENCE OF VOICED/UNVOICED
C INPUTS) AND PASSES THEM THROUGH A TIME VARYING DIGITAL FILTER
C TO PRODUCE AN OUTPUT SPEECH SEQUENCE.
C
C INPUTS: U      -IS THE EXCITATION TO THE "THROAT" DIGITAL FILTER
C                WHICH CAN BE EITHER VOICED OR UNVOICED.
C      ISIZE -IS THE LENGTH OF THE INPUT EXCITATION U.
C      FILTER-IS THE VALUES OF THE COEFFICIENTS (AND KEEPS
C                OLD VALUES BETWEEN CALLS)
C      NORDER-IS THE ORDER OF THE FILTER
C      G      -IS THE GAIN
C      S      -IS THE OUTPUT SPEECH SEQUENCE
C      W      -IS THE VALUE OF THE FILTER AT EACH NODE
C      FRMSIZ-IS THE NUMBER OF POINTS BETWEEN COEFFICIENT UPDATES
C      NDEGP -KEEPS TRACK OF THE CURRENT POSITION (WHEN THIS EXCEEDS
C                A FRAME BOUNDARY NEW COEFFICIENTS ARE READ IN)
C      LASTD -WHEN THIS EQUALS ONE THE PROGRAM HAS READ THE LAST
C                COEFFICIENT SET
C      NFILP  -KEEPS TRACK OF ABSOLUTE FRAME POSITION
C      NSIL1  -A ZERO INDICATES SILENCE, A ONE IS SPEECH
C      IN     -CURRENT BLOCK NUMBER
C
C*****
      DIMENSION U(180), FILTER(20), W(0:20), T(80), S(180)
      INTEGER FRMSIZ
      DO 20 J=1,80
      X   T(J) = 0.0
      X20  CONTINUE
      IF(NSIL1 .EQ. 1)GO TO 30
      DO 25 I=1,20
      X   W(I) = 0.0
      25  CONTINUE
      30  DO 500 N=1, ISIZE
      X   IF(NDEGP .LT. (NFILP + FRMSIZ))GO TO 307
      DO 300 K=1, NORDER
      X   READ BINARY(3,END=305)FILTER(K)
      300  CONTINUE
      READ BINARY(3,END=305)G
      NFILP = NFILP + FRMSIZ
      GO TO 307
      305  LASTD = 1
      307  CONTINUE
      TOTAL = 0.0
      DO 400 K=1, NORDER
      X   TOTAL = TOTAL - W(K) * FILTER(K)
      400  CONTINUE
      W(0) = TOTAL + G * U(N)
      X   T(N) = W(0)
      IF(NSIL1 .EQ. 1)GO TO 425
      S(N) = 0.0
      GO TO 436
      425  S(N) = W(0)
      436  CONTINUE
      C   IF((IN .LT. 83).OR. (IN.GT. 84))GO TO 437
      C   TYPE"U(N) ",U(N),"S(N) ",S(N),"N ",N,"BLOCK",IN,NSIL1
      C   TYPE"SIL1",NSIL1

```

```

437      DO 450 I=1,NORDER
          MKD = NORDER + 1 - I
          W(MKD) = W(MKD-1)
450      CONTINUE
          W(0) = 0.0
          NBEGP = NBEGP + 1
500      CONTINUE
X         IFSC=0
X         MO = 0
X         NO = 1
X         CALL GRPH2("THROAT",NG,T,U,ISIZE,MO,YM,YA,IFSC)
X         ACCEPT="CONTINUE? ",NDOG
600      RETURN
          END

```

```

C*****
C THIS SUBROUTINE CREATES A NORMAL RANDOM SEQUENCE WHICH WILL
C BE AN INPUT TO THROAT
C
C INPUTS: FRMSIZ-LENGTH OF THE OUTPUT SEQUENCE U(N)
C          G      -SCALING VALUE (TYPICALLY 1000.0)
C          IX      -DOUBLE PRECISION SEED (GOOD INITIAL VALUE IS 203.0)
C          SIL2    -(1-UNVOICED, 0-SILENT)
C          RATIO   -SCALE FACTOR (TYPICALLY .1) FRACTION OF G USED
C                   RELATIVE TO VOICED SPEECH
C
C OUTPUT: U      -A NORMALLY GENERATED RANDOM SEQUENCE OF LENGTH FRMSIZ
C                IF SIL2=1 ELSE A SEQUENCE OF 0'S (SAME LENGTH).
C
C*****
SUBROUTINE UNVOCD(U, FRMSIZ, G, IX, SIL2, RATIO)

DOUBLE PRECISION    U1, V, W, T, X, E, E2, E10, E3, Z, P, P1, F, P2, P3, P4, P5, P1
X , P12, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A12, A13, A14, A15, A16, A17, A18
DOUBLE PRECISION INTEGER IX
INTEGER FRMSIZ, SIL2
DIMENSION U(180)
DATA A1/.884070402298758D0/,
X   A2/.131131635444180D0/,
X   A3/.986655477086949D0/,
X   A4/.958720824790463D0/,
X   A5/.630834801921960D0/,
X   A6/.755591531667601D0/,
X   A7/.034240503750111D0/,
X   A8/.911312780288703D0/,
X   A9/.479727404222441D0/,
X   A10/.10547366102207D0/,
X   A12/.872834976671790D0/,
X   A13/.049264496373128D0/,
X   A14/.5955071380159401D0/,
X   A15/.805577924423817D0/,
X   A16/.053377549506886D0/,
X   A17/.973310954173898D0/,
X   E/2.216035867166471D0/,
X   A18/.180025191068563D0/,

C
C***SCALE FOR SILENCE
C
IF(SIL2.NE.0)GO TO 500
DO 400 I=1,180
U(I) = 0.0
400 CONTINUE
GO TO 2500
500 CONTINUE
C
C*** CALCULATE THE NORMAL FUNCTION
C
P1 = 3.1415926536D0
P12 = (P1*2.0)**(-.5)
E2 = E**2.0
E3 = E2/2.0
DO 2200 N = 1,FRMSIZ
U1 = DRAND(IX)
IF(U1.GT.A1)GO TO 1000
V = DRAND(IX)

```

```

X = E * (A2 * U1 + V - 1)
GO TO 9000
1000 IF(U1 .LT. A17)GO TO 1200
1005 V = DRAND(IX)
W = DRAND(IX)
T = E3 - DLOG(W)
E10 = T * V**2
IF(E10 .GT. E3)GO TO 1005
IF(U1 .LT. A3)GO TO 1010
X = -(2.0 * T)**.5
GO TO 9000
1010 X = (2.0 * T)**.5
GO TO 9000
1200 IF(U1 .LT. A4)GO TO 1500
1300 V = DRAND(IX)
W = DRAND(IX)
Z = V - W
T = E - A5 * DMIN1(V,W)
P = DMAX1(V,W)
IF(P .LT. A6)GO TO 1800
P1 = A7 * DABS(Z)
F = P12*DEXP(-T*T/2.0)-A18*(E-DABS(T))
IF(P1 .LE. F)GO TO 1900
GO TO 1300
1500 IF(U1 .LE. A8)GO TO 1700
1600 V = DRAND(IX)
W = DRAND(IX)
Z = V - W
T = A9 + A10 * DMIN1(V,W)
P2 = DMAX1(V,W)
IF(P2 .LE. A12)GO TO 1800
P3 = A13 * DABS(Z)
F = P12 * DEXP(-T*T/2.0)-A18*(E-DABS(T))
IF(P3 .LE. F)GO TO 1800
GO TO 1600
1700 V = DRAND(IX)
W = DRAND(IX)
Z = V - W
T = A9 - A14 * DMIN1(V,W)
P4 = DMAX1(V,W)
IF(P4 .LE. A15)GO TO 1800
P5 = A16 * DABS(Z)
F = P12 * DEXP(-T*T/2.0)-A18*(E-DABS(T))
IF(P5 .LE. F)GO TO 1800
GO TO 1700
1800 IF(Z .LT. 0.0)GO TO 1700
X = -T
GO TO 2000
1900 X = T
2000 CONTINUE
9000 CONTINUE
U(N) = (G) * SNGL(X) * RATIO
X TYPE U(N)
2200 CONTINUE
2300 CONTINUE
RETURN
END

```



```
500      TIME = TIME + 1 0  
X        WRITE(12,900)I,U(I)  
X900     FORMAT(1X,15,1X,G16.7)  
600      CONTINUE  
         RETURN  
         END
```



```

C*****
C
C PROGRAM: SIFT
C AUTHOR: WILL JANSSEN AND CRAIG MCKOWN
C DATE: 19 JULY 83 UPDATED 26 AUG 83
C LANGUAGE: FORTRAN5
C
C FUNCTION: THIS PROGRAMS SETS UP AND RUNS THE SIFT PITCH DETECTOR
C AS DESCRIBED BY MARKEL AND GRAY WITH A SILENCE
C DETECTOR ADDED.
C
C LOAD COMMAND LINE: RLDR PITCHER& SIFTA DIRECT AUTO ENER
C IOF @FLIDE
C
C RUN LINE: SIFT FILE1/I FILE2/O FILE3/I
C INPUTS: SPEEFL-IS THE INPUT SPEECH FILE AND IS REPRESENTED
C BY FILE1 ABOVE
C PARAM -THIS INPUT FILE (OUTPUT OF SETUP) CONTAINING
C 25 RECORDS OF REAL DATA THEN 25 RECORDS OF
C INTEGER DATA. THIS DATA IS USED TO INITIALIZE
C THIS FILE. DATA FROM THIS FILE USED IN THIS
C PROGRAM:
C RECORD#6-RPARAM(6) IS THE SILENCE THRESHOLD
C
C OUTPUTS: DUMMY -THIS IS THE OUTPUT PITCH FILE WHERE EACH
C RECORD REPRESENTS THE PITCH INFORMATION
C FOR EACH 80 POINTS. THE RECORD FORMAT IS
C ON LINE 049.
C*****
C
C INTEGER SPEEFL(7), PARAM(7), DUMMY(7), RUMMY(7)
C INTEGER VAL(768), TRPIT(400), NE(400), NSPMAX(256)
C DIMENSION SPCH(400), PBUF(100), PITCH(3), RPARAM(25), IPARAM(25)
C DIMENSION PIT(-1:400)
C
C MAXSET = 400
C MAXPT = 80
C
C DO 30 I=1,400
C NE(I) = 0
C 30 CONTINUE
C NFILES = 3
C CALL IOF(NFILES, MAIN, SPEEFL, DUMMY, PARAM, RUMMY, MS, S1,
C X S2, S3, S4)
C CALL OPEN(1, SPEEFL, 1, IER)
C IF (IER.NE. 1) TYPE "OPEN FILE ERROR ", IER
C CALL OPEN(3, PARAM, 3, IER)
C IF (IER.NE. 1) TYPE "OPEN FILE ERROR PARAM ", IER
C DO 40 I=1,25
C READ FREE(3) RPARAM(I)
C 40 CONTINUE
C DO 41 I=1,25
C READ FREE(3) IPARAM(I)
C 41 CONTINUE
C
C *** FIND MAX VALUE OF SPEECH
C
C IBIG = 88
C MBLOCK = 1

```

```

NV = 0
N600 = 0
N500 = 0
1041 CONTINUE
CALL RDBLK(1, NV, NSPMAX, MBLCK, IENDS)
DO 1042 J=1, 256
    N200 = IABS(NSPMAX(J))
    IF(N200 .GT. N500) N500 = N200
    N600 = N600 + 1
1042 CONTINUE
NV = NV + 1
IF((NV .LT. IBIG).AND. (IENDS .NE. 9)) GO TO 1041
CALL REWIND(1)
S200 = 32000.0/FLOAT(N500)
THRESH = RPARAM(6)
PITCH(1)=0.0
PITCH(2)=0.0
PITCH(3)=0.0
NSET = 0
S = 1
K = 0
70 CONTINUE
IF (NSET .GE. MAXSET) GO TO 160
NSET = NSET + 1
NPOINT = 0
CALL RDBLK(1, K, VAL, 3, IER)
IF (IER .EQ. 9) GO TO 160
IF (IER .NE. 1) TYPE "READ FILE ERROR ", IER
F = S + 399
DO 100 J=S,F
    NPOINT=NPOINT+1
    SPCH(NPOINT) = (FLOAT(VAL(J))/204.8) * S200
100 CONTINUE
C
C*** DETERMINE IF ENERGY THRESHOLD IS EXCEEDED
C
    CALL ENER(SPCH, THRESH, NEN)
    NE(NSET) = NEN
X    TYPE"NEN= ", NEN
C
C***CALL TO THE SUBROUTINES WHICH PERFORM PITCH ANALYSIS
C
    CALL SIFTA(SPCH, PITCH)
    DELAY = NSET - 2
    IF (PITCH(3) .EQ. 0.0) GO TO 110
    PIT(DELAY) = (PITCH(3) - 1)*4.
    GO TO 120
110 PIT(DELAY) = 0.0
120 CONTINUE
S = S + MAXPT
IF (S .LE. 256) GO TO 70
S = S - 256
K = K+1
GO TO 70
160 CONTINUE
MSET=NSET ; TOTAL # OF SETS
CALL CLOSE(1, IER)
IF (IER .NE. 1) TYPE "CLOSE FILE ERROR ", IER
DO 180 I = 1, MSET - 2
    TRPIT(I) = INT(PIT(I))

```

```

180  CONTINUE
      DO 190 I = MSET - 1, MAXSET
          TRPIT(I) = 0
190  CONTINUE
      CALL DFILW(DUMMY, IER)
      IF (IER.EQ.13) GOTO 200
      IF (IER.NE.1) TYPE "DELETE ERROR ", IER
200  CALL CFILW(DUMMY, 2, IER)
      IF (IER.NE.1) TYPE "CREATE FILE ERROR ", IER
      CALL OPEN(2, DUMMY, 3, IER)
      DO 900 J=1, MAXSET
          NSIL = 1
          IF (TRPIT(J) .EQ. 0) NSIL = 0
          IF (NE(J) .EQ. 1) GO TO 300
          NSIL = 0
          TRPIT(J) = 0
300  WRITE(2, 849) NSIL, NE(J), TRPIT(J)
849  FORMAT(3X, 3(110, 3X))
900  CONTINUE
      IF (IER.NE.1) TYPE "WRITE BLOCK ERROR ", IER
      STOP
      END

```

```

C*****
C
C      SIFT ALGORITHM PROCESSING - STEP1
C
C      INPUT PARAMETERS:  SPCH(J)  (J=1,2,...,400)
C                        THE SPEECH SIGNAL TO BE PROCESSED FOR PITCH
C
C      OUTPUT PARAMETER:  PITCH(J)  (J=1,2,3)
C                        THE PITCH IN UNITS OF [ ]
C
C      NOTE: PARAMETERS FIXED FOR FS=8 KHZ
C*****

SUBROUTINE SIFTA(SPCH,PITCH)
DIMENSION SPCH(1),PBUF(100),AF(4),PF(4),DF(5),D(5),ABUF(33)
DIMENSION U(100),A(5),P(5),RC(5),PITCH(1)
DATA AF/1.,-2.340366,2.011900,-.614109/
DATA PF/.0357082,-.0069956,-.0069956,.0357082/
DATA P/1.,4*0./

C***  INITIALIZE MEMORY OF DIRECT TO ZERO
DO 10 J=1,5
  DF(J)=0.
  D(J)=0.
10  CONTINUE

C***  PRE-FILTER, DOWN-SAMPLER, DIFFERENCER AND HAMMING WINDOWER.
UPREV=0.
DO 20 J=1,400
  CALL DIRECT(AF,PF,3,DF,SPCH(J),SOUT)
  IF (MOD(J,4).NE.0) GO TO 20
  K=J/4
  PBUF(K)=SOUT
  U(K)=(SOUT-UPREV)*(.54-.46*COS((K-1)*6.28318/99.))
  UPREV=SOUT
20  CONTINUE

C***  COMPUTE INVERSE FILTER COEFFICIENTS
CALL AUTO(100,U,4,A,ALP,RC)

C***  PERFORM INVERSE FILTERING AND HAMMING WINDOW
DO 30 J=1,100
  CALL DIRECT(P,A,4,D,PBUF(J),FOUT)
  IF (J.LE.4) GO TO 30
  PBUF(J-4)=FOUT*(.54-.46*COS((J-5)*6.28318/95.))
30  CONTINUE

C***  SIFT ALGORITHM PROCESSING - STEP2
C***
C      INPUT PARAMETER:  PBUF(J)  (J=1,2,...,76)
C                        THE DOWN-SAMPLED, FILTERED ERROR SIGNAL
C
C***  PERFORM AUTOCORRELATION ON PITCH BUFFER
DO 25 JJ=1,33
  J=JJ-1
  NMJ=76-J
  SUM=0.
  DO 15 I=1,NMJ
    IPJ=I+J

```

```

          SUM=SUM+PBUF(I)*PBUF(IPJ)
15      CONTINUE
          ABUF(JJ)=SUM
25      CONTINUE

C***    OBTAIN PITCH VALUES FROM LAST THREE FRAMES
        P1=PITCH(1)
        P2=PITCH(2)
        P3=PITCH(3)

C***    GET PEAK WITHIN RANGE[6,32]
        L=6
        AMAX=ABUF(L)
        DO 35 J=6,32
            IF(ABUF(J).LE.AMAX) GO TO 35
            AMAX=ABUF(J)
            L=J
35      CONTINUE

C***    TEST FOR MAX EQUAL ZERO
        IF (AMAX.EQ.0.) GO TO 60

C***    TEST FOR LEFT HAND EDGE.
C***    IF ABUF(L) IS NOT A PEAK, SET UNVOICED.
        IF (ABUF(L).LT.ABUF(L-1)) GO TO 60

C***    PERFORM PARABOLIC INTERPOLATION
C***    ABOUT LOCATION L
        AA=ABUF(L-1)-ABUF(L)
        AA=(AA+ABUF(L+1)-ABUF(L))/2.
        BB=(ABUF(L+1)-ABUF(L-1))/4.
        AP=ABUF(L)-BB*BB/AA
        AL=L-BB/AA
        V=AP/ABUF(1)

C***    TEST WITH VARIABLE THRESHOLD
        IF (L.GE.19) GO TO 40
        DD =-1. *(L-6.)/13. +2.
        GO TO 50
40      CONTINUE
        DD =-1. *(L-19.)/13. +1
50      CONTINUE
        V=V/DD

C***    DECISIONS
        IF(V.GE..37) GO TO 70
        IF(P1.EQ.0.) GO TO 60
        IF(V.GE..32) GO TO 70
60      PO=0.
        GO TO 80
70      PO=AL
80      IF(ABS(P1-P3).LE..375*P3) P2=(P1+P3)/2.

C      IF(PO AND P1 ARE CLOSE) AND (P2 NOT 0)
C      BUT P3=0, THEN USE LINEAR EXTRAPOLATION FOR P2
C      (COMING OUT OF VOICED).
C***    IF (P3.NE.0.) GO TO 90
        IF(P2.EQ.0.)GO TO 90
        IF (ABS(PO-P1).GT.0.2*P1) GO TO 90

```

```

      P2=(2.*P1)-P0
C***  TEST FOR ISOLATED "VOICED" AND INCORRECT END OF "VOICED"
      90  IF (P1.NE.0.) GO TO 100
          IF (ABS(P2-P3).GT.(.375*P3)) P2=0.

C***  UPDATE FRAMES
      100 PITCH(3)=P2
          PITCH(2)=P1
          PITCH(1)=P0

C***  TRUE PITCH DELAYED BY TWO FRAMES EQUALS:
C***  (PITCH(3)-1)*5
      RETURN
      END

```

```

      SUBROUTINE ENER(SPCH,THRESH,NEN)
C*****
C
C  THIS SUBROUTINE DETERMINES WHETHER A FRAMES ENERGY
C  EXCEEDS A SILENCE THRESHOLD
C
C  INPUTS: SPCH  -THIS IS ONE FRAMF OF SPEECH
C           THRESH-THIS IS THE SILENCE THRESHOLD
C
C  OUTPUTS: NEN  -THIS IS A SILENCE INDICATOR WHERE A ZERO IS
C                FOR SILENCE AND A ONE FOR SPEECH
C
C*****
      DIMENSION SPCH(400)
      NEN = 1
      SUM = 0.0
      DO 100 J=1,400
        S1 = SPCH(J)/100.0
        SUM = SUM + S1 * S1
100    CONTINUE
      IF(SUM .LT. THRESH)NEN = 0
      RETURN
      END

```

```

C *****
C
C THE SUBROUTINE AUTO AS PRESENTED IN MARKEL & GRAY
C THIS SUBROUTINE COMPUTES THE AUTOCORRELATION THEN INVERTS
C THE MATRIX TO FIND COEFFICIENTS
C
C INPUTS: N      -SIZE OF THE AUTOCORRELATION
C          X      -SEQUENCE BEING AUTOCORRELATED
C          M      -ORDER OF THE FILTER MINUS ONE
C
C OUTPUTS: A      -INVERTED COEFFICIENTS
C          ALPHA  -THE CURRENT SYSTEM GAIN
C          RC     -THE REFLECTION COEFFICIENTS
C
C *****

```

```

      SUBROUTINE AUTO(N, X, M, A, ALPHA, RC)

      DIMENSION X(1), A(1), RC(1)
      DIMENSION R(21)

C
      NP=M+1
      DO 15 K=1, NP
        R(K)=0.
        NK=N-K+1
        DO 10 NP=1, NK
          R(K)=R(K)+X(NP)*X(NP+K-1)
10      CONTINUE
15      CONTINUE
      RC(1)=-R(2)/R(1)
      A(1)=1.
      A(2)=RC(1)
      ALPHA=R(1)+R(2)*RC(1)
      DO 40 MINC=2, M
        S=0.
        DO 20 IP=1, MINC
          S=S+R(MINC-IP+2)*A(IP)
20      CONTINUE
          RC(MINC)=-S/ALPHA
          MM=MINC/2+1
          DO 30 IP=2, MM
            IB=MINC-IP+2
            AT=A(IP)+RC(MINC)*A(IB)
            A(IB)=A(IB)+RC(MINC)*A(IP)
            A(IP)=AT
30      CONTINUE
            A(MINC+1)=RC(MINC)
            ALPHA=ALPHA+RC(MINC)*S
            IF(ALPHA) 50, 50, 40
40      CONTINUE
50      RETURN
      END

```



```

      SUBROUTINE DIRECT(A,P,M,D,XIN,XOUT)
C*****
C  THIS ROUTINE IMPLEMENTS THE DIRECT FORM FILTER.
C
C  INPUTS: A      -THIS IS THE SET OF FILTER COEFFICIENTS
C              FOR THE POLES
C           P      -THIS IS THE SET OF FILTER COEFFICIENTS
C              FOR THE ZEROS
C           M      -THIS IS THE ORDER OF THE FILTER MINUS ONE
C           D      -THIS IS THE SET OF INTERMEDIATE NODAL VALUES
C           XIN    -INPUT TO THE FILTER
C
C  OUTPUTS: XOUT  -THIS IS THE FILTER OUTPUT
C
C*****
      DIMENSION A(1),P(1),D(1)
C
      XOUT = 0.0
      D(1) = XIN
      DO 10 J = 1, M
         I = M + 1 - J
         XOUT = XOUT + D(I+1)*P(I+1)
         D(1) = D(1) - A(I+1)*D(I+1)
         D(I+1) = D(I)
      10  CONTINUE
      XOUT = XOUT + D(1)*P(1)
      RETURN
      END

```

```

C *****
C PROGRAM: PITCH2
C AUTHOR: WILL JANSSEN
C DATE: 22 APRIL 80
C LANGUAGE: FORTRAN5
C FUNCTION: THIS PROGRAM INPUTS A SPEECH FILE (NO PARTICULAR
C LENGTH REQUIRED) AND PRODUCES A FILE OF FRAME
C INFORMATION CONTAINING A VOICED/UNVOICED DECISION
C AND A SILENCE DECISION FOR EACH FRAME. THE BASIC ALGORITHM OF THIS
C PITCH DETECTOR COMES FROM J. J. DUBNOWSKI, R. W. SCHAFER, AND L. R.
C RABINER, "REAL-TIME DIGITAL HARDWARE PITCH DETECTOR," IEEE TRAN.
C ACoust., SPEECH, AND SIGNAL PROC., VOL. ASSP-24, NO. 1, FEB 76.
C
C LOAD COMMAND LINE: RLDR PITCH IOF LPF ENERGY SETMAX
C CLPPER AUTCOR @FLIB@
C
C RUN LINE: PITCH2 FILE1/I FILE2/O FILE3/I FILE4/I
C WHERE:
C FILE1-IS THE INPUT BINARY SPEECH FILE
C FILE2-IS THE PITCH DETECTOR OUTPUT INCLUDING
C A VOICED/UNVOICED INDICATOR, A
C SILENCE/SPEECH INDICATOR, AND A PITCH
C PERIOD. SEE LINE 849 FOR THE OUTPUT FORMAT
C FILE3-CONTAINS A SET OF INITIALIZATION PARAMETERS
C FILE4-CONTAINS A SET OF FILTER COEFFICIENTS FOR
C THE LOW PASS FILTER
C
C WHAT FOLLOWS IS A SHORT DESCRIPTION OF SOME OF THE VARIABLES
C USED IN THIS PROGRAM:
C
C PARAMETER LIST-(WHERE A SET IS DEFINED AS 1/3 OF A FRAME)
C NUMSEG-(I) NUMBER OF SEGMENTS(1/3 FRAME) PROCESSED BY PITCH
C FILSPI-(IS) FILE NAME OF INPUT SPEECH FILE
C OUTFRM-(IS) FILE NAME OF OUTPUT FRAME INFORMATION
C NSETS-(I) CURRENT SET NUMBER
C MXSETS-(I) MAXIMUM NUMBER OF SETS(DETERMINES SPEECH ARRAY SIZE)
C NPOINT-(I) CURRENT POINT NUMBER IN CURRENT SET
C MAXPNT-(I) MAXIMUM NUMBER OF POINTS PER ARRAY FILL(50 BLOCKS)
C SPEECH(A,B)-(R) THE ARRAY OF SPEECH DATA WHERE A IS THE SET
C NUMBER AND B IS THE POINT IN THE SET
C LSTSET-(I) IN THE CASE WHERE THERE IS LESS THAN 25 BLOCKS
C READ IN LSTSET IS THE LAST SET USED
C ZEROSE-(I) SET TO ONE IF ZEROING WAS REQUIRED FOR ANY SET
C TFRMAR-(R) THRESHOLD FRAME ARRAY CONTAINS MAX. VALUE IN EACH SET
C THEN SETS THRESHOLD IN EACH FRAME
C PNTSET-(I) NUMBER OF POINTS IN EACH SET
C LSTRND-(I) LASTROUND, WHEN COMPLETED THE PITCH DETECTOR IS DONE
C RPARAM-(R) REAL RUN PARAMETERS FROM PARAMETER FILE
C IPARAM-(I) INTEGER RUN PARAMETERS FROM PARAMETER FILE
C LSTFRM-(I) LAST FRAME IS 2 LESS THAN THE NUMBER OF FRAMES BEING
C WORKED ON
C AUTTHR-(R) THRESHOLD FOR DETECTING 2ND AUTOCORRELATION PEAK
C TMAUT-(R) THIS ARRAY HOLDS THE AUTOCORRELATION FUNCTION FOR ONE FRAME
C SPCH-(R) THIS ARRAY HOLDS ONE FRAME OF SPEECH INFORMATION FOR
C AUTOCORRELATION COMPUTATIONS
C POS-(I) LOCATION OF AUTOCORRELATION PEAK
C ISPECH-(I) THE INTEGER VAR THE SPEECH FILE IS READ INTO
C FRAMAR-(I) ARRAY OF SILENCE, VOICED/UNVOICED DECISIONS, & PITCH
C MAX-(R) VALUE OF AUTOCORRELATION PEAK
C OVLSET-(I) THE SET STARTING LOCATION

```

```

C      DIMENSION SPEECH(10,80), FRMAR(10), FILSPI(7), OUTFRM(7),
X      PARAM(7), RPARAM(25), IPARAM(25), FRAMAR(10,3), FLTRFL(7),
X      LPFSPF(7), ZROCOF(100), S1(2), S2(2), S3(2), S4(2), MAIN(7),
X      OLDSPC(120), TEMPSP(4,80)
      INTEGER FILSPI, OUTFRM, NUMSEC, NSETS, MXSETS, NPOINT, MAXPNT, LSTSET,
X      PNTSET, PARAM, BLOCKS, BEGSET, TLSTST, FRAMAR, ZERO SP, LSTRND,
X      FLTRFL, S1, S2, S3, S4, OVLSET, FOURPS, FILRND, NSPMAX(256),
X      NDEL(3,3)
      REAL SPEECH, RPARAM, LPFSPF
      NUMSEC = 10
      MAXPNT = 800
      MXSETS = 10
      NFC = 0
      DO 1001 J10=1,3
        DO 1000 J11=1,3
          NDEL(J10,J11) = 0
1000      CONTINUE
1001      CONTINUE

C
C***INITIALIZES ARRAY THAT STORES OLD SPEECH FOR WINDOW INFORMATION
C
      DO 8 K5=1,120
        OLDSPC(K5) = 0.0
      8      CONTINUE

C
C*** READ INPUT AND OUTPUT FILES FOR DATA***
C
      NFILES = 4
      CALL IOF(NFILES,MAIN,FILSPI,OUTFRM,PARAM,FLTRFL,MS,S1,
X      S2,S3,S4)
      CALL OPEN(1,FILSPI,1,IER)
      IF(IER.NE.1)TYPE"OPEN ERROR ON ",FILSPI(1),IER
      CALL OPEN(2,OUTFRM,3,IER)
      CALL OPEN(3,PARAM,1,IER)
      IF(IER.NE.1)TYPE"OPEN ERROR ON ",PARAM(1),IER
      DO 10 I=1,25
        READ FREE(3) RPARAM(I)
10      CONTINUE
      DO 11 I=1,25
        READ FREE(3) IPARAM(I)
11      CONTINUE
      CALL CLOSE(3,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ",PARAM(1),IER
      CALL OPEN(4,FLTRFL,1,IER)
      TYPE"PAST OPEN ON FLTRFL"
      IF(IER.NE.1)TYPE"OPEN ERROR ON ",FLTRFL(1),IER

C
C***READ IN NO. OF FILTER POINTS(MAX=120 POINTS)WHERE NFLTZR=0'S,
C      NFPOL=POLE'S
      READ FREE(4) NFLTZR
X      TYPE"NFLTZR=",NFLTZR
      READ FREE(4) NFPOL
      DO 13 I=1,NFLTZR
        READ FREE(4) ZROCOF(I)
13      CONTINUE
14      CONTINUE
      CALL CLOSE(4,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR ",FLTRFL(1),IER

```

```

C
C***FIND MAX VALUE OF SPEECH
C
      IBIG = 88
      MBLOCK = 1
      NV = 0
      N600 = 0
      N500 = 0
29    CONTINUE
      CALL RDBLK(1, NV, NSPMAX, MBLOCK, IENDS)
      DO 30 J=1, 256
        N200 = IABS(NSPMAX(J))
        IF(N200 .GT. N500) N500 = N200
        N600 = N600 + 1
30    CONTINUE
      NV = NV + 1
      IF((NV .LT. IBIG). AND. (IENDS .NE. 9)) GO TO 29
X     TYPE"THE FOLLOWING NO. OF POINTS WHERE CHECKED ", N600
X     TYPE"AND THE MAX. VALUE FOUND WAS ", N500
      CALL REWIND(1)
      S200 = 32000.0/FLOAT(N500)

C
C***INPUT 10 BLOCK SEGMENT-IF NOT POSSIBLE ZERO FILL WHATS LEFT OF
C   THE LAST SET**
C
      M100 = 0
      FILRND = 0
      OVLSET = 1
      BLOCKS = 0
      ZEROSP = 0
      CLPLEV = RPARAM(1)
      AUTTHR = RPARAM(2)
      MXSTPT = IPARAM(1)           ; USUALLY 80
      VOCTHR = RPARAM(3)
      SILTHR = RPARAM(10)
      FOURPS = 4 * MXSTPT         ; FOUR SETS
15    N = 1

C
C*** LOADS ZEROS FOR LAST RUN
C
      IF(FILRND .EQ. 0) GO TO 44
      DO 41 KN=5, 8
        DO 40 KN2=1, PNTSET
          SPEECH(KN, KN2) = 0.0
40      CONTINUE
41      CONTINUE
      LSTSET = 8
      LSTRND = 1
      GO TO 75
44      CONTINUE
      PNTSET = MXSTPT
      NSETS = OVLSET ; 1 FIRST TIME, ELSE 5
      NPOINT = 1
      LSTSET = MXSETS
      POINTS = MXSETS * MXSTPT

C
C*** CHECK FOR LAST SET IN CURRENT SERIES
C
45    IF (NSETS .GT. MXSETS) GO TO 75
55    IF (NPOINT .LT. (MXSTPT + 1)) GO TO 65

```

```

C
C*** COUNTS THROUGH SETS
C
      NPOINT = 1
      NSETS = NSETS + 1
      GO TO 45
C
C*** READS IN A RAW SPEECH FILE
C
45      READ BINARY(1,END=70) ISPECH      ; READ RAW SPEECH
      M100 = M100 + 1
      SPEECH(NSETS,NPOINT) = FLOAT(ISPECH)*S200
68      CONTINUE
      NPOINT = NPOINT + 1
      GO TO 55
70      LSTSET = NSETS
X      TYPE=LSTSET = ",LSTSET
      ZEROSP = 1
      FILRND = 1
      DO 71 I=NPOINT,MXSTPT      ; FILLS LAST SET WITH ZEROS
      SPEECH(NSETS,I) = 0.0
71      CONTINUE
C
C*** ZERO FILL TILL LAST FRAME
C
      IF(LSTSET .LE. 6)LSTRND = 1
      FILRND = 1
      IF(LSTSET .EQ. 10)GO TO 73
      LSTSET = LSTSET + 1
      DO 73 N50=LSTSET,10
        DO 72 N60=1,PNTSET
          SPEECH(N50,N60) = 0.0
72      CONTINUE
73      CONTINUE
75      CONTINUE
X      ACCEPT"LPF ?1-Y,0-N",IYS
X      IF(IYS .EQ. 0)GO TO 9000
      CALL LPF(SPEECH,LSTSET,ZROCOF,NFLTRZ,PNTSET,OLDSPC,
X      TEMPSP,OVLSET)
C      ACCEPT" ENERGY? 1-Y,0-N",IYS
C      IF(IYS .EQ. 0)GO TO 9000
      CALL ENERGY(SPEECH,LSTSET,FRAMAR,SILTHR,PNTSET,VOCTHR)
      TLSTST = LSTSET - 2      ; CALCULATE EXCEPT LAST 2 SETS
      BEGSET = 1
X      ACCEPT"DO SETMAX? 1-Y,0-N",IYS
X      IF(IYS .EQ. 0)GO TO 9000
      CALL SETMAX(SPEECH,LSTSET,TFRMAR,PNTSET,RPARAM(1),BEGSET)
X      ACCEPT" DO CLPPER? 1-Y,0-N",IYS
X      IF(IYS .EQ. 0)GO TO 9000
      CALL CLPPER(SPEECH,LSTSET,TFRMAR,PNTSET,BEGSET)
X      ACCEPT" DO AUTCOR? 1-Y,0-N",IYS
X      IF(IYS .EQ. 0)GO TO 9000
      CALL AUTCOR(SPEECH,LSTSET,AUTTHR,PNTSET,FRAMAR,BEGSET)
C***THE PURPOSE OF THIS CODE IS TO ACCOUNT FOR THE 2 SET OVERLAP
C      PROBLEM BETWEEN BLOCKS READ IN***
C      WRITES LAST TWO SETS INTO NEXT FIRST TWO SETS
      IF (LSTRND .EQ. 1)GO TO 800
      DO 800 J=1,4
        DO 790 K=1,PNTSET
          SPEECH(J,K) = TEMPSP(J,K)

```

```

790      CONTINUE
800      CONTINUE
          OVLSET = 5
          BLOCKS = BLOCKS + 1
X      TYPE"JUST RAN 640 SAMPLES OF SPEECH, SEQ. NO.: ",BLOCKS
          NSTTMP = LSTSET - 4
          DO 860 N5=1,NSTTMP
              NPO = FRAMAR(N5,3)
              NFC = NFC +1
              NP1 = NDEL(1,3)
              NP2 = NDEL(2,3)
              NP3 = NDEL(3,3)
805      IF(ABS(NP1-NP3).LE..375*NP3)NP2=(NP1+NP3)/2.0
C      IF(NPO AND NP1 ARE CLOSE) AND (NP2 NOT 0)
C      BUT NP3=0, THEN USE LINEAR EXTRAPOLATION OF NP2
C      (COMING OUT OF VOICED)
          IF(NP3.NE.0)GO TO 809
          IF(NP2.EQ.0)GO TO 809
          IF(ABS(NPO-NP1).GT.0.2*NP1)GO TO 809
          NP2 = (2.0*NP1)-NPO
C***  TEST FOR ISOLATED "VOICED" AND INCORRECT END OF "VOICED"
809      IF(NP1.NE.0)GO TO 810
          IF(ABS(NP2-NP3).GT.(.375*NP3))NP2=0
810      NDEL(3,3) = NP2
          NDEL(3,2) = NDEL(2,2)
          NDEL(3,1) = NDEL(2,1)
          NDEL(2,3) = NP1
          NDEL(2,2) = NDEL(1,2)
          NDEL(2,1) = NDEL(1,1)
          NDEL(1,3) = NPO
          NDEL(1,2) = FRAMAR(N5,2)
          NDEL(1,1) = FRAMAR(N5,1)
          IF(NFC.LT.3)GO TO 850
          WRITE(2,849)NDEL(3,1),NDEL(3,2),NDEL(3,3)
849      FORMAT(3X,3(I10,3X))
850      CONTINUE
860      CONTINUE
X      TYPE"M100= ",M100
          IF (LSTRND.EQ. 1)GO TO 9000
          GO TO 15
9000      CONTINUE
          WRITE(2,849)NDEL(2,1),NDEL(2,2),NDEL(2,3)
          WRITE(2,849)NDEL(1,1),NDEL(1,2),NDEL(1,3)
          CALL CLOSE(1,IER)
          IF (IER.NE. 1) TYPE"CLOSE FILE ERROR1",IER
          CALL CLOSE(2,IER)
          IF (IER.NE. 1) TYPE"CLOSE FILE ERROR2",IER
          TYPE "THE PROGRAM HAS NOW COMPLETED"
          STOP
          END

```

```

      SUBROUTINE LPF(SPEECH,LSTSET,ZROCOF,NFLTRZ,PNTSET,
X   OLDSPC,TEMPSP,OVLSET)
C*****
C   THIS SUBROUTINE IS A LOW PASS FILTER(MAX 100 POINTS) WITH
C   A CUT OFF FREQUENCY OF 900HZ. THE TYPE OF WINDOW CHOSEN FOR
C   THE FILTER IS USER DEPENDENT. THIS ROUTINE IS PASSED IN A FILE OF
C   FILTER COEFFICIENTS WHERE THE FILE IS AN OUTPUT OF KATHY WARD'S
C   WFILTER. THE FIRST NUMBER READ IS THE NUMBER OF ZERO COEFFICIENTS
C   (USUALLY 99),THE SECOND IS THE NUMBER OF POLE COEFFICIENTS(ALWAYS
C   ZERO, HOWEVER THE PROGRAM COULD BE UPDATED TO ACCEPT IIR FILTERS)
C   AND THE NEXT VALUES(USUALLY 99) ARE THE ZERO COEFFICIENTS
C   B(0) TO B(99). THE FORMULA USED FOR THE FILTER IS  $Y(N) =$ 
C   SUM(FROM K=0 TO M) OF B(K)*X(N-K) WHERE M IS THE NO. OF FILTER
C   POINTS.
C
C   INPUTS: SPEECH-THIS IS THE INPUT SPEECH FILE (ALSO THE OUTPUT)
C           LSTSET-THE NUMBER OF THE LAST SET TO BE PROCESSED
C           ZROCOF-THIS IS A SET OF FILTER COEFFICIENTS
C           NFLTRZ-THIS IS THE NUMBER OF ZERO COEFFICIENTS
C           PNTSET-THIS IS THE NUMBER OF POINTS IN 1/3 OF A FRAME
C           OLDSPC-THIS IS OLD SAVED SPEECH
C           TEMPSP-THIS TEMPORARY ONE DIMENSIONAL ARRAY HOLDS SPEECH
C                   FOR PROCESSING
C           OVLSET-INDICATES SET PROCESSING STARTING POINT
C
C   OUTPUTS: SPEECH-THIS IS THE OUTPUT SPEECH FILE (ALSO THE INPUT)
C
C*****
      DIMENSION ZROCOF(100),SPEECH(LSTSET,PNTSET),
X   SPHLPF(-119:800),OLDSPC(120),U(500),T(500),TEMPSP(4,PNTSET)
      INTEGER PNTSET,OVLSET
C***WRITE SPEECH FILE INTO A TEMPORARY FILE SPHLPF-SPEECH LPF
C***
C*** PUT 120 VALUES FROM LAST RUN INTO SPHLPF FOR THE WINDOW TO USE
C***
      DO 20 N3=1,120
          N4 = N3 - 120
          SPHLPF(N4) = OLDSPC(N3)
C   WRITE(12,29)SPHLPF(N4),N4
C29  FORMAT(3X,'SPHLPF= ',F10.2,3X,'N4= ',I10)
      20  CONTINUE
          J1 = 0
C
C***EXCEPT FIRST TIME UNFILTERED SPEECH IS LOADED INTO SPHLPF
C
          J2 = 0
          IF(OVLSET.EQ. 1)GO TO 30
          DO 28 M7=1,4
              DO 26 M8=1,PNTSET
                  J2 = J2 + 1
                  SPHLPF(J2) = TEMPSP(M7,M8)
      26  CONTINUE
      28  CONTINUE
          J1 = J2 ;SETS NEW STARTING POINT FOR SPHLPF
      30  DO 40 I=OVLSET,LSTSET
              DO 35 J=1,PNTSET
                  J1 = J1 + 1
                  SPHLPF(J1) = SPEECH(I,J)
C   WRITE(12,29)SPHLPF(J1),J1
      35  CONTINUE

```

```

40      CONTINUE
C
C***PUT LAST 120 VALUES OF SPHLPF INTO OLDSPC TO BE READY FOR NEXT RUN
C
      DO 45 N6=1,120
        N7 = J1 - 120 - PNTSET + 4
        IF (N7 .LE. 0)GO TO 47      ; NO SAVING REQUIRED, LAST RUN
        N8 = N7 + N6
        OLDSPC(N6) = SPHLPF(N8)
45      CONTINUE
47      CONTINUE
C
C*** SAVES NON LPF SPEECH IN A TEMPORARY ARRAY FOR NEXT LPF RUN
C
      IF(LSTSET .LT. 10)GO TO 78
      LTEMP = LSTSET - 4
      DO 78 M1=1,4
        M3 = LTEMP + M1
        DO 77 M2=1,PNTSET
          TEMPSP(M1,M2) = SPEECH(M3,M2)
77      CONTINUE
78      CONTINUE
C
C*** SET PLOT VARIABLES
C
      NGPONT = 500
      YMAX = 3000
      YMIN = -3000
      NG = 1
      MODE = 0
      IFSLC = 0
C*** SEND SPEECH THROUGH THE FILTER
      NSET = 1
      NSTPNT = 1
      DO 60 N=1,J1
        SUM = 0.0
C*** MULTIPLY SEQUENCE AND FILTER A1 DELAYS
        DO 50 K=1,NFLTR2
          NKDIFF = N - K
          SUM = ZROCOF(K) * SPHLPF(NKDIFF) + SUM
50      CONTINUE
52      IF (NSTPNT .LT. (PNTSET+1)) GO TO 55
          NSET = NSET + 1
          NSTPNT = 1
55      CONTINUE
C***PUT FILTERED SPEECH BACK INTO SPEECH FILE
      SPEECH(NSET,NSTPNT) = SUM
C
      WRITE(12,59)SUM,N
C59      FORMAT(3X,'SPEECH= ',F10.2,3X,'N=',I10)
          NSTPNT = NSTPNT + 1
X          IF(N .GT. 500)GO TO 60
X          T(N) = SUM
60      CONTINUE
X          CALL GRPH2("LPF",NG,T,U,NGPONT,MODE,YMIN,YMAX,IFSLC)
          RETURN
          END

```



```

SUBROUTINE ENERGY(SPEECH,LSTSET,FRAMAR,SILTHR,PNTSET,VOCTHR)
C*****
C THIS ROUTINE USES A RECTANGULAR WINDOW 120 SAMPLES LONG. THIS WAS
C CHOSEN AS A COMPROMISE TO REFLECT THE CHANGING PROPERTIES OF THE
C SPEECH SIGNAL YET NOT BE LESS THAN A PITCH PERIOD.
C
C INPUTS: SPEECH-THIS IS THE INPUT SPEECH FILE
C          LSTSET-THIS IS THE LAST SET TO BE PROCESSED
C          PNTSET-THIS IS HOW MANY POINTS IN 1/3 OF A FRAME
C          VOCTHR-THIS IS THE VOICED/UNVOICED THRESHOLD
C          SILTHR-THIS IS THE SILENCE THRESHOLD
C
C OUTPUTS FRAMAR-THIS IS A 2-D ARRAY THAT CONTAINS PITCH INFORMATION
C           WHERE THIS SUBROUTINE UPDATES THE SILENCE AND VOICED
C           INDICATOR
C*****

      DIMENSION SPEECH(LSTSET,PNTSET),SQRD(240),
X      SUM(240),FRAMAR(LSTSET,3),TEMP(500),U(500)
      INTEGER FRAMAR,CURRNT,LAST,TOTL,FIRST,LSTSET,
X      TWCpst,PNTSET
      LSTENG = LSTSET - 4
      FIRST = 0

C
C***INITIALIZE FRAMAR
C
X      M6 = 0
      DO 10 M1=1,LSTSET
        DO 9 M2=1,3
          FRAMAR(M1,M2) = 0
        9      CONTINUE
      10      CONTINUE
      TWCpst = 2 * PNTSET
      DO 100 I=1,LSTENG
        IF (I .GT. 1) FIRST = 1

C
C***FILL THE FIRST TWO SETS AND PUT THEM IN SQRD A TEMPORARY ARRAY
C
      IF (FIRST .EQ. 1) GO TO 25
      DO 20 K=1,PNTSET
        SQRD(K) = (SPEECH(I,K)/1000.0)**2 ,NORMALIZE SPEECH
        K1 = K + PNTSET
        SQRD(K1) = (SPEECH(I+1,K)/1000.0)**2
        K2 = K + TWCpst
        SQRD(K2) = (SPEECH(I+2,K)/1000.0)**2
      20      CONTINUE
      GO TO 35
C***PUT LAST 160 SQUARED VALUES IN PLACE OF FIRST 160***
      25      DO 30 L=1,TWCpst
        SQRD(L) = SQRD(L+PNTSET)
      30      CONTINUE
C***LOAD NEW 3RD SET ***
      DO 33 L1=1,PNTSET
        SQRD(L1+TWCpst) = (SPEECH(I+2,L1)/1000.0)**2
      33      CONTINUE
      35      CONTINUE
C
C***TOTAL UP SQUARES IN 119 POINT WINDOWS(COMPUTE NEW WINDOW ON 1 POINT
C SHIFTS)

```

```

      DO 60 CURRNT=1,PNYSET
      LAST = CURRNT + 120
      TOTL = 0
      DO 55 N=CURRNT, LAST
      TOTL = TOTL + GORD(N)
55      CONTINUE
C
C
C***CREATE TEMPORARY ARRAY FOR PLOTTING
C
X      IF(M6 .GT. 500)GO TO 57 ;ARRAY ONLY HOLDS 500 VALUES
X      TEMP(M6) = TOTL
X      M6 = M6 + 1
X57     CONTINUE
C***SEE IF EXCEEDS SILENCE THRESHOLD***
      IF (TOTL .LT. SILTHR) GO TO 58
      FRAMAR(1,2) = 1
C
C***SEE IF EXCEEDS VOICED THRESHOLD
C
58      IF(TOTL.LT.VOCTHR)GO TO 60
      FRAMAR(1,1) = 1
      GO TO 90
60      CONTINUE
90      CONTINUE
X      TYPE"FRAMAR(1,1)= ",FRAMAR(1,1),I
100     CONTINUE
C
C*** PLOT ENERGY
C
X      NG = 1
X      MODE = 0
X      IFSLC = 0
X      M6 = M6 - 1
X      CALL GRPH2("ENERGY",NG,TEMP,U,M6,MODE,YMIN,YMAX,IFSLC)
      RETURN
      END

```

```

      SUBROUTINE SETMAX(SPEECH,LSTSET,TFRMAR,PNTSET,CLPLEV,BEGSET)
C*****
C   THIS SUBROUTINE CALCULATES THE MAXIMUM VALUE IN EACH SET
C   THEN CALCULATES THE FRAME THRESHOLD FOR CLIPPING BY THE FOLLOWING
C   FORMULA CLIPLEVEL = K*MIN(PEAK1,PEAK2) WHERE PEAK1 AND
C   PEAK2 ARE THE PEAK VALUES IN THE FIRST THIRD AND LAST THIRD OF A FRAME
C
C   INPUTS: SPEECH-THIS IS THE INPUT SPEECH FILE
C           LSTSET-THIS IS THE LAST SET TO BE PROCESSED
C           PNTSET-THIS IS HOW MANY POINTS ARE IN 1/3 OF A FRAME
C           CLPLEV-THIS IS THE CLIPPING LEVEL IN PERCENT OF MAX
C           BEGSET-THIS IS THE FIRST SET TO BE PROCESSED
C
C   OUTPUT: TFRMAR-THE ACTUAL VALUE THE SUBROUTINE CLPPER WILL USE FOR
C           CLIPPING THIS FRAME OF SPEECH
C*****
      DIMENSION SPEECH(LSTSET,PNTSET),TFRMAR(LSTSET)
      INTEGER PNTSET,BEGSET
      REAL MAXVAL,CLPPER
C
C*** FIND MAXIMUM VALUE IN EACH SET
C
      DO 70 NSET=BEGSET,LSTSET
        TFRMAR(NSET) = 0.0
        DO 60 N=1,PNTSET
          IF (SPEECH(NSET,N).GT.TFRMAR(NSET))TFRMAR(NSET)=SPEECH(NSET,N)
        60   CONTINUE
      70   CONTINUE
C***CALCULATE CLIPPING LEVEL IN EACH FRAME EXCEPT THE LAST TWO
      LSTFRM = LSTSET - 2
      DO 80 I=BEGSET,LSTFRM
C   CHOOSE THE SMALLER ONE UNLESS ZERO
        IF(TFRMAR(I+2) .LT. 10)GO TO 75
        IF(TFRMAR(I).GT.TFRMAR(I+2))TFRMAR(I) = TFRMAR(I+2)
C***SET CLIPPING LEVEL
      75   TFRMAR(I) = CLPLEV * TFRMAR(I)
X      TYPE=TFRMAR(I)=",TFRMAR(I),I
      80   CONTINUE
      RETURN
      END

```

```

      SUBROUTINE CLPPER(SPEECH,LSTSET,TFRMAR,PNTSET,BEGSET)
C*****
C THE PURPOSE OF THIS SUBROUTINE IS TO DO CENTER AND PEAK CLIPPING
C THE THRESHOLD IS SET FOR EACH FRAME IN SUBROUTINE SETMAX. IN A GIVEN
C FRAME VALUES ABOVE THE THRESHOLD ARE SET TO +1.0, VALUES BELOW - THE
C THRESHOLD ARE SET TO -1.0, AND VALUES IN BETWEEN ARE SET TO 0.0
C
C INPUTS: SPEECH-THIS IS THE INPUT SPEECH FILE (ALSO AN OUTPUT)
C         LSTSET-THIS IS THE LAST SET TO BE PROCESSED
C         TFRMAR-THIS IS THE CLIPPING THRESHOLD
C         PNTSET-THIS IS THE NUMBER OF POINTS IN 1/3 OF A FRAME
C         BEGSET-THIS IS THE FIRST SET TO BE PROCESSED
C
C OUTPUT: SPEECH-THIS IS THE SPEECH THAT HAS BEEN CLIPPED
C
C*****
      DIMENSION SPEECH(LSTSET,PNTSET),TFRMAR(LSTSET),TEMP(500),
X      U(500)
      INTEGER BEGSET,PNTSET
      LSTFRM = LSTSET - 2
      DO 70 NSET=BEGSET,LSTFRM
        DO 60 N=1,PNTSET
          IF (SPEECH(NSET,N) .LT. TFRMAR(NSET)) GO TO 50
          SPEECH(NSET,N) = 1.0
          GO TO 60
50        IF (SPEECH(NSET,N) .LT. (-TFRMAR(NSET))) GO TO 52
          SPEECH(NSET,N) = 0.0
          GO TO 60
52        SPEECH(NSET,N) = -1.0
60      CONTINUE
70    CONTINUE
C
C*** PLOT CLPPER
C
X      MB = 1
X      NG = 1
X      MODE = 0
X      IFSLC = 0
X      DO 105 M5=BEGSET,LSTFRM
X        DO 100 M6=1,PNTSET
X          IF(MB .GT. 500)GO TO 110
X          TEMP(M6) = SPEECH(M5,M6)
X          MB = MB + 1
X100      CONTINUE
X105      CONTINUE
X110      CONTINUE
X      MB = MB - 1
X      CALL GRPH2("CLPPER",NG,TEMP,U,MB,MODE,YMIN,YMAX,IFSLC)
      RETURN
      END

```

```

      SUBROUTINE AUTCOR(SPEECH,LSTSET,AUTHR,PNTSET,FRAMAR,BEGSET)
C*****
C   THIS ROUTINE COMPUTES THE SHORT TIME AUTOCORRELATION FUNCTION USING
C   THE FIRST 1/3 OF THE FRAME WITH LACS FROM N=16 TO 160(2 TO 20MSEC)
C   NEXT, THE FIRST PEAK AFTER R(0) IS DETERMINED, IF IT EXCEEDS THE
C   THRESHOLD(I.E. VOICED) THEN THE FRAME'S PITCH IS DETERMINED. OTHERWISE
C   THE FRAME IS CONSIDERED UNVOICED.
C
C   INPUTS: SPEECH-THIS IS THE INPUT SPEECH FILE
C           LSTSET-THIS IS THE LAST SET TO BE PROCESSED
C           AUTHR-THIS IS THE AUTOCORRELATION THRESHOLD WHICH USED TO
C               FIND THE FUNDAMENTAL PEAK
C           PNTSET-THIS IS THE NUMBER OF POINTS IN 1/3 OF A FRAME
C           BEGSET-THIS IS THE FIRST SET TO BE PROCESSED
C
C   OUTPUT: FRAMAR-THIS 2-D ARRAY CONTAINS PITCH INFORMATION WHERE
C           THIS SUBROUTINE ENTERS A PITCH VALUE IF VOICED
C*****
      DIMENSION SPEECH(LSTSET,PNTSET),SPCH(240),FRAMAR(10,3),
X   TMPAUT(0:160)
      INTEGER BEGWIN,ENDWIN,POS,FRAMAR,BEGSET,TWCPST,PNTSET,
X   CNSET1,CNSET2
      REAL MAX
      LSTFRM = LSTSET - 4
      N6 = 0
      TWCPST = 2 * PNTSET
      DO 170 NSET=BEGSET,LSTFRM
      FMIN = 10000      ; INITIALIZING FOR FIRST VALLEY
      N56 = 0
      N6 = N6 + 1
      RO = 0.0
C
C*** CALCULATE R(0) FOR THE FRAME
C
      DO 100 N=1,PNTSET
      RO = RO + SPEECH(NSET,N)**2
100    CONTINUE
      TMPAUT(0) = RO
C
C*** INITILIZE SPCH TO ZERO
C
      DO 103 M16=1,240
      SPCH(M16) = 0.0
103    CONTINUE
C
C*** CALCULATE R(K) FOR THE FRAME
C***PUT 3 SETS INTO NEW ARRAY SPCH
C
      MN = 0
      CNSET1 = NSET
      CNSET2 = CNSET1 + 2
      DO 110 I=CNSET1,CNSET2
      DO 105 J=1,PNTSET
      MN = MN + 1
      SPCH(MN), = SPEECH(I,J)
105    CONTINUE
110    CONTINUE
      DO 150 MN=1,2
      DO 150 K=0,TWCPST

```

```

      RN = 0.0
      DO 130 N=1, PNTSET
        RN = RN + SPCH(K+N) * SPCH(N)
130    CONTINUE
        TMPAUT(K) = RN
X      N5 = K
150    CONTINUE
        HALF = TMPAUT(0)/3.0
        DO 1000 NF=0, TWCPST
          IF(TMPAUT(NF).LT. HALF)GO TO 900
          TMPAUT(NF) = TMPAUT(NF) - HALF
          GO TO 950
        900    IF(TMPAUT(NF).LT. (-HALF))GO TO 950
              TMPAUT(NF) = 0.0
        950    TMPAUT(NF) = TMPAUT(NF) + HALF
        1000    CONTINUE
        153    CONTINUE
X      N5 = N5 + 1
X      IF((NSET.EQ. LSTFRM).AND. (N6.EQ. 6))GO TO 155
X      GO TO 150
X155    CALL PLOT10(TMPAUT, N5, N6, X0, Y0, 1.0)
X      N6 = 10
X158    CALL PLOT10(TMPAUT, N5, N6, X0, Y0, 1.0)
X      IF(NSET.EQ. LSTFRM)N6 = 10
          IF(FRAMAR(NSET, 1).EQ. 0)GO TO 169
          MAX = 0.0
C
C*** SEARCH FOR FIRST PEAK TO EXCEED THE THRESHOLD TTSTER
C
      DO 163 NST=1, 1
        IMULT = NIT - 1
        TTH = .05 * FLOAT(IMULT) + .5
        TTSTER = R0 * (1.0 - TTH)
        DO 162 NST=20, 160
          IF(TMPAUT(NST).LT. TTSTER)GO TO 162
          POS = NST
          MAX = TMPAUT(NST)
          GO TO 164
        162    CONTINUE
        163    CONTINUE
        164    CONTINUE
C*** NORMALIZE THE PEAK
        MAX = MAX/R0
        IF (MAX.LT. AUTTHR) GO TO 168
C*** VOICED DECISION = 1, UNVOICED = 0, POS = PITCH
        FRAMAR(NSET, 2) = 1
        FRAMAR(NSET, 3) = POS
        GO TO 170
        168    FRAMAR(NSET, 2) = 0
        169    CONTINUE
        170    CONTINUE
              RETURN
              END

```

```

C*****
C
C      PROGRAM SCALE.FR
C
C      THIS PROGRAM SCALES SPEECH FILES SO THAT THERE IS A MAX VALUE
C      OF 1900 AND CAN DE-EMPHASIZE SPEECH
C
C      INPUT: MUST BE A BLOCKED FILE
C*****
      DIMENSION S1(256)
      DOUBLE PRECISION IX
      INTEGER OUTFILE(7), INFILE(7), FILUFD(18), SPEECH(256)
      NNOSIZ = 0
      IX = DBLE(203)
      NNEWS = 0
      ACCEPT"WARNING: THE INPUT FILE MUST BE AN INTEGER FILE <CR>
X          AND BE IN BLOCKED FORM. <CR> <CR>
X          DO YOU WISH TO CONTINUE?(1-Y,0-N) ",NYZ
      IF(NYZ.EQ.0)GO TO 60
      ACCEPT"INPUT FILENAME: "
      READ(11,39)INFILE(1)
39      FORMAT(S13)
      OPEN 1,INFILE,ATT="CI",ERR=40
      FLIP = 1.0
      ACCEPT"OUTPUT FILENAME: "
      READ(11,39)OUTFILE(1)
      OPEN 2,OUTFILE,ERR=50
      NDE = 0
      ACCEPT"FLIP FILE (1-YES,0-NO) ",FLIPY
      IF(FLIPY.EQ.1)FLIP=-1.0
      ACCEPT"OUTPUT FILE SIZE? ",ISIZE
      ACCEPT"PERFORM NOISE ADDITION?(1-Y,0-N)",NNOIS
      IF(NNOIS.EQ.0)GO TO 53
      ACCEPT"SIZE OF MAX NOISE?(REAL) ",VNOSIZ
33      CONTINUE
      MBLOCK = 1
      N18 = 0
      NV = 0
70      N6 = 0
      S4 = 0
      DO 80 I=1,256
      S1(I) = 0.0
80      CONTINUE
      N5 = 0
100     CONTINUE
      CALL RDBLK(1,NV,SPEECH,MBLOCK,IENDS)
      DO 200 J=1,256
      IF(NNOIS.EQ.0)GO TO 120
      NNEWS = INT((SNGL(DRAND(IX))-5)*VNOSIZ)
      SPEECH(J) = SPEECH(J) + NNEWS
120     N2 = IABS(SPEECH(J))
180     IF(N2.GT.N5)N5 = N2
      N6 = N6 + 1
200     CONTINUE
      NV = NV + 1
      IF(NV.LT.ISIZE)GO TO 100
500     TYPE"THE FOLLOWING NO. OF POINTS WERE CHECKED ",N6
      TYPE"AND THE MAX. VALUE FOUND WAS ",N5
      CALL REWIND(1)

```

```

IX = DBLE(203)
S3 = 0.0
S4 = 0.0
S6 = 0.0
S2 = 1900.0 / FLOAT(N5) * FLIP
N6 = 0
NV = 0
600 CALL RDBLK(1, NV, SPEECH, MBLOCK, IENDS)
DO 700 J=1, 256
    IF(NNOIS.EQ.0)GO TO 650
    NNEWS = INT(SNGL(DRAND(IX))*VNOSIZ)
650    SPEECH(J) = SPEECH(J) + NNEWS
680    S1(J) = FLOAT(SPEECH(J)) * S2
    SPEECH(J) = INT(S1(J))
700 CONTINUE
    CALL WRBLK(2, NV, SPEECH, MBLOCK, IER)
    N6 = N6 + 1
    NV = NV + 1
    IF(NV .LT. ISIZE)GO TO 600
900 CONTINUE
    N15 = N6 * 256
    TYPE"THE FOLLOWING NO. OF POINTS WERE OUTPUT ",N6
    CALL CLOSE(1, IER)
    IF(IER .NE. 1)TYPE"CLOSE ERROR ON INPUT ", IER
    CALL CLOSE(2, IER)
    IF(IER .NE. 1)TYPE"CLOSE ERROR ON OUTPUT ", IER
    TYPE"PROCESSED: ",N6
    GO TO 60
50 TYPE"OPEN ERROR ON OUTPUT "
    GO TO 60
40 TYPE"OPEN ERROR ON INPUT "
60 STOP
END

```



```

C PROGRAM PLTTER
C*****
C
C THIS PROGRAM WORKS INTERACTIVELY WITH THE USER TO PRODUCE A
C PITCH PLOT OR A PLOT OF SPEECH. THE SPEECH PLOT REQUIRES A
C MAXIMUM VALUE OF 2000.0 TO HAVE CONSTANT VERTICAL AXIS.
C
C*****
      DIMENSION OUTFILE(7),T(512),V(512)
      INTEGER OUTFILE
      ACCEPT"FILE NAME? "
      READ(11,39)OUTFILE(1)
39      FORMAT(S13)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN ERROR ",IER
      ACCEPT"FILE OUTPUT FROM PITCH? 1-Y,0-N ",M100
      IF(M100.EQ.1)GO TO 1000
      DO 50 I=1,512
         T(I) = 0.0
         V(I) = 0.0
50      CONTINUE
      NP = 0
      DO 200 M=1,500
         DO 100 I=1,512
            N = 1
            IF(M100.NE.1)GO TO 75
            READ(1,65,END=300)J1,J2,J3
            FORMAT(3X,3(I10,3X))
            V(I) = FLOAT(J3)
            T(I) = FLOAT(J2)
            GO TO 100
75          READ BINARY(1,END=300)J3
            V(I) = FLOAT(J3)
100         CONTINUE
            NP = NP + 1
            NPTS = N
            SF = 1.0
            TYPE"RUNNING PLOT",NP,"NPTS",NPTS
            CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
            IF(NP.EQ.10)NP = 0
200        CONTINUE
300        CONTINUE
            IF(N.EQ.512)GO TO 400
            DO 400 M3=N,512
               V(M3) = 0.0
400         CONTINUE
            IF(NP.NE.0)GO TO 450
            NP = 1
            CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
            NP = 10
            DO 430 M3=1,512
               V(M3) = 0.0
430         CONTINUE
            CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
            GO TO 500
450        CONTINUE
            IF(NP.NE.6)GO TO 475
            CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
            GO TO 425
475        CONTINUE

```

```

      NP = 10
      CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
500   CALL CLOSE(1,IER)
      GO TO 5000
1000  DO 1100 I=1,512
      N=I
      IF(M100 .NE. 1)GO TO 1075
      READ(1,65,END=1300)J1,J2,J3
      FORMAT(3X,3(I10,3X))
      V(I) = FLOAT(J3)
      T(I) = FLOAT(J2)
      GO TO 1100
1075  READ BINARY(1,END=1300)J1,J2,J3
      V(I) = FLOAT(J3)
1100  CONTINUE
1300  CONTINUE
      NPTS = N
      NP = 1
      SF = 1.0
      TYPE"RUNNING PLOT",NP
      CALL PLOT10(V,NPTS,NP,X0,Y0,SF)
      NP = 10
      TYPE"RUNNING PLOT"
      CALL PLOT10(T,NPTS,NP,X0,Y0,SF)
1500  CALL CLOSE(1,IER)
5000  STOP
      END

```

```

C*****
C      PROGRAM:      SETUP
C      AUTHOR:       WILL JANSSEN
C      DATE:         17 APRIL 83
C      LANGUAGE:     FORTRAN3
C      FUNCTION:     THIS PROGRAM ALLOWS THE USER TO SETUP A FILE THAT
C                   CONTAINS INFORMATION REQUIRED TO RUN THE
C                   RECURSIVE LINEAR PREDICTIVE CODER(RLPC).
C                   THE PROGRAM WILL ALLOW THE USER THE FOLLOWING
C                   OPTIONS.
C                   1) CREATE A NEW FILE
C                   2) UPDATE AN OLD FILE
C                   3) PRINT PARAMETERS
C
C
C      LOAD COMMAND LINE: RLDR SETUP @FLID@
C
C      NOTE:          1) THE ARRAYS ARE SET TO MAX OF 25 VARIABLES
C                   EACH.
C                   2) THE REAL ARRAY IS CALLED RELVAR AND THE
C                   INTEGER ARRAY IS CALLED INTVAR
C*****
C
C      SETUP
C*****
C      DIMENSION RELVAR(25), INTVAR(25),OUTFILE(7)
C      INTEGER YES,YES2,SIZE,SIZE1,YES5
C***  SIZE =REAL ARRAY SIZE,SIZE1 = INTEGER ARRAY SIZE
C      SIZE = 25
C      SIZE1 = 25
C***  NEW OR OLD FILE ***
C      WRITE(10,29)
C      29  FORMAT(3X, 'THIS PROGRAM CREATES AN INPUT FILE FOR THE RLPC')
C      ACCEPT"ARE YOU CREATING A FILE FROM SCRATCH?(1-YES,0-NO)",YES
C      TYPE" "
C      IF (YES .EQ. 0)GO TO 30
C
C      C***  NEW FILE ***
C
C      ACCEPT"DOES THE NEW FILE CURRENTLY EXIST?(1-YES,0-NO)",YES2
C      TYPE" "
C      IF(YES2 .EQ. 1)GO TO 30
C      TYPE" THE FILE MUST BE CREATED BEFORE USING THIS PROGRAM"
C      GO TO 1000
C
C      C***  GET FILE NAME ***
C
C      30  ACCEPT"FILE NAME? "
C      READ(11,39)OUTFILE(1)
C      39  FORMAT(S13)
C      CALL OPEN(1,OUTFILE,3,IER)
C      IF(IER .NE. 1)TYPE"OPEN ERROR ",IER
C      TYPE" "
C
C      C***  INITIALIZE THE ARRAYS,NEW FILES-SET = TO 0,OLD FILES-READ IN
C      OLD FILES***

```

```

C      IF (YES .EQ. 0) GO TO 50
      DO 45 I=1, SIZE1
45     RELVAR(I) = 0.0
      DO 47 I=1, SIZE1
47     INTVAR(I) = 0
      GO TO 60
50     READ (1, 901) (RELVAR(I), I=1, SIZE1)
      READ (1, 902) (INTVAR(I), I=1, SIZE1)
C
C*** UPDATE ARRAYS ***
C
      TYPE*          <CR>
X      IF YOU CHOOSE TO CHANGE A VARIABLE ENTER Y <CR>
X      OTHERWISE JUST DO A CARRIAGE RETURN. <CR>
X      <CR>"
60     CONTINUE
      TYPE*THE CURRENT VALUE OF CLPLEV (IN PITCH2) IS : ", RELVAR(1)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5000
      ACCEPT"<CR> INPUT NEW REAL VALUE : ", RELVAR(1)
C
5000    TYPE*CURRENT AUTOCORRELATION THRESHOLD (IN PITCH2) IS : ", RELVAR(2)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5001
      ACCEPT"<CR> INPUT NEW VALUE. ", RELVAR(2)
C
5001    TYPE*THE CURRENT VALUE OF VOICED/UN THRESH (PITCH2) IS: ", RELVAR(3)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5002
      ACCEPT" <CR> INPUT NEW REAL VALUE: ", RELVAR(3)
C
5002    TYPE*THE CURRENT VALUE OF WINDOW SHAPE (CODER) : ", RELVAR(4)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5003
      ACCEPT" <CR> INPUT NEW VALUE: ", RELVAR(4)
C
5003    TYPE*CURRENT VALUE OF SPEECH SCALE-(IN CODER): ", RELVAR(5)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5015
      ACCEPT" <CR> INPUT NEW VALUE: ", RELVAR(5)
C
5015    TYPE*CURRENT VALUE OF SILENCE THRESH-(SIFT) IS: ", RELVAR(6)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5216
      ACCEPT" <CR> INPUT NEW VALUE: ", RELVAR(6)
C
5216    TYPE" VALUE OF UNVOCD TO VOICED GAIN (SYNTH) IS: ", RELVAR(7)
      TYPE*CHANGE VALUE? <CR>      "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR .NE. 89) GO TO 5217
      ACCEPT" <CR> INPUT NEW VALUE: ", RELVAR(7)
C
5217    TYPE" VALUE OF GLOTTAL PULSE POS. (SYNTH) IS: ", RELVAR(8)

```

```

TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5218
ACCEPT" <CR> INPUT NEW VALUE ",RELVAR(8)
C
5218 TYPE" VALUE OF GLOTTAL PULSE NEG. (SYNTH) IS: ",RELVAR(9)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5219
ACCEPT" <CR> INPUT NEW VALUE. ",RELVAR(9)
C
5219 TYPE"CURRENT VALUE SILENT THR. (PITCH2) IS: ",RELVAR(10)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5013
ACCEPT" <CR> INPUT NEW VALUE: ",RELVAR(10)
C
5013 TYPE"CURRENT VALUE:NO. OF POINTS/SET (PITCH2) ",INTVAR(1)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5004
ACCEPT" <CR> INPUT NEW VALUE: ",INTVAR(1)
C
5004 TYPE" VALUE OF FILTER SPACINGS IS (CODER & SYNTH) : ",INTVAR(2)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5005
ACCEPT" <CR> INPUT NEW VALUE: ",INTVAR(2)
C
5005 TYPE"USE PRE/DE-EMPHASIS(1-Y,0-N) (CODER&SYNTH) : ",INTVAR(3)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5006
ACCEPT" <CR> INPUT NEW VALUE: ",INTVAR(3)
C
5006 TYPE"USE (0-IMPULSE,1-GLOTTAL PULSE) (SYNTH) : ",INTVAR(4)
TYPE"CHANGE VALUE? <CR> "
CALL RCHAR(ICHAR, IER)
IF(ICHAR.NE.89)GO TO 5007
ACCEPT" <CR> INPUT NEW VALUE: ",INTVAR(4)
C
5007 CONTINUE
C
C*** TYPE ARRAY ***
C
ACCEPT"DO YOU WANT TO HAVE THE ARRAY TYPED(1-YES,0-NO): ",YES
TYPE" "
IF(YES.EQ.0)GO TO 200
TYPE" CLPLEV: ",RELVAR(1)
TYPE" AUTOCORRELATION THRESHOLD: ",RELVAR(2)
TYPE"VOICED/UN THRESHOLD: ",RELVAR(3)
TYPE"WINDOW SHAPE VALUE: ",RELVAR(4)
TYPE"SPEECH SCALE-(IN CODER): ",RELVAR(5)
TYPE"SILENCE THRESHOLD: ",RELVAR(6)
TYPE"UNVOCD TO VOICED GAIN: ",RELVAR(7)
TYPE"GLOTTAL PULSE POS.: ",RELVAR(8)
TYPE"GLOTTAL PULSE NEG.: ",RELVAR(9)
TYPE"SILENCE THR. IN PITCH: ",RELVAR(10)
TYPE"NO. POINTS/SET: ",INTVAR(1)
TYPE"FILTER SPACING IS: ",INTVAR(2)

```

```

      TYPE"PRE/DE-EMPHASIS(1-Y,0-N): " ,INTVAR(3)
      TYPE"GLOTTAL PULSE(1-Y,0-N): " ,INTVAR(4)
C
C*** OUTPUT FILE ***
C
200  ACCEPT"IS THE INPUT FILE THE OUTPUT FILE(1-YES,0-NO): " ,YES2
      TYPE"
      IF (YES2 .EQ. 1)GO TO 75
      CALL CLOSE(1,IER)
      IF (IER .NE. 1)TYPE"CLOSE FILE ERROR1 " ,IER
      ACCEPT"FILE NAME? "
      READ(11,69)OUTFILE(1)
69    FORMAT(S13)
      CALL OPEN(1,OUTFILE,3,IER)
      IF (IER .NE. 1)TYPE"OPEN ERROR ON OUTPUT FILE " ,IER
75    CALL REWIND(1)
      WRITE (1,901)(RELVAR(1),I=1,SIZE1)
      WRITE (1,902)(INTVAR(1),I=1,SIZE1)
      CALL CLOSE(1,IER)
      IF (IER .NE. 1)TYPE"CLOSE FILE ERROR2 " ,IER
      TYPE"
      ACCEPT"PRINT ARRAY ON PRINTRONIX ?(1-Y,0-N) " ,YES
      TYPE"
      IF (YES .EQ. 0)GO TO 1001
      WRITE(12,1499)OUTFILE(1)
      CALL FGDAY(1MON,1DAY,1YEAR)
      CALL FOTIME(1HOUR,1MIN,1SEC)
      WRITE(12,1311)1MON,1DAY,1YEAR
      WRITE(12,1312)1HOUR,1MIN,1SEC
1311  FORMAT("0","DATE " ,1X,12,"/",12,"/",12)
1312  FORMAT("0","TIME " ,1X,12,":",12,":",12)
      WRITE(12,1500)RELVAR(1)
      WRITE(12,1501)RELVAR(2)
      WRITE(12,1502)RELVAR(3)
      WRITE(12,1503)RELVAR(4)
      WRITE(12,1504)RELVAR(5)
      WRITE(12,1505)RELVAR(6)
      WRITE(12,1506)RELVAR(7)
      WRITE(12,1507)RELVAR(8)
      WRITE(12,1508)RELVAR(9)
      WRITE(12,1509)RELVAR(10)
      WRITE(12,1600)INTVAR(1)
      WRITE(12,1601)INTVAR(2)
      WRITE(12,1602)INTVAR(3)
      WRITE(12,1603)INTVAR(4)
1499  FORMAT(1X,S13)
1500  FORMAT("0"," CLIPPING LEVEL " ,F12.5)
1501  FORMAT("0"," AUTOCORRELATION THRESHOLD " ,F12.5)
1502  FORMAT("0"," VOICED/UNVOICED THRESHOLD " ,F12.5)
1503  FORMAT("0"," WINDOW SHAPE A=" ,F12.5)
1504  FORMAT("0"," SPEECH SCALER " ,F12.5)
1505  FORMAT("0"," SILENCE THRESHOLD (SIFT PITCH) " ,F12.5)
1506  FORMAT("0"," UNVOICED TO VOICED GAIN " ,F12.5)
1507  FORMAT("0"," GLOTTAL PULSE POS. SLOPE " ,F12.5)
1508  FORMAT("0"," GLOTTAL PULSE NEG. SLOPE " ,F12.5)
1509  FORMAT("0"," SILENCE THR. IN PITCH " ,F12.5)
1600  FORMAT("0"," POINTS/SET " ,I6)
1601  FORMAT("0"," FRAME SEPERATION " ,I6)
1602  FORMAT("0"," PRE/DE-EMPHASIS(1-Y,0-N) " ,I6)
1603  FORMAT("0"," GLOTTAL PULSE(1-YES,0-IMPULSE) " ,I6)

```

```
900  FORMAT(3X, 'TEST1', F10.5)
901  FORMAT(3X, F10.3)
902  FORMAT(3X, I10)
1000  TYPE 'PROGRAM COMPLETED'
1001  STOP
      END
```

```

C*****
C
C      PROGRAM      KEVAL
C      AUTHOR      KATHY DIXON WARD WITH MODIFICATION BY WILL JANSSEN
C      DATE        8 APRIL 83
C      LANGUAGE    FORTRAN 5
C
C      FUNCTION:    THIS PROGRAM CALCULATES THE LINEAR MAGNITUDE,
C                   LOG MAGNITUDE, PHASE, AND IMPULSE RESPONSE OF
C                   A DIGITAL FILTER FOR N POINTS.  OPTIONS FOR
C                   PLOTTING THE RESULTS ON THE TEKTRONIX 4010 AND
C                   STORING THE RESULTS IN FILES- OUT1(MAGNITUDE,
C                   PHASE) AND OUT2(IMPULSE RESPONSE) ARE AVAILABLE.
C                   INPUT DATA TO THE PROGRAM IS A FILE CONTAINING
C                   M,N,A(I),B(I),AND AO FROM THE FOLLOWING FORM
C                   OF THE FILTER TRANSFER FUNCTION:
C
C                   
$$H(Z) = AO * (B(0) + B(1) * Z^{*-1} + \dots + B(M) * Z^{*-M}) /$$

C                   
$$(A(0) + A(1) * Z^{*-1} + \dots + A(N) * Z^{*-N})$$

C
C      LOAD COMMAND LINE: RLDR KEVAL PLOT10 PLOT5 LB GRPH LB @FLID@
C
C      NOTE:        1) THE LINK DP4F GRPH.LB MUST EXIST.
C                   2) THE MAXIMUM NUMBER OF POINTS (NP) MUST BE
C                      LESS THAN OR EQUAL TO 500.
C*****
C
C      COMPLEX Y, SUM1, SUM2, SUM3
C      DIMENSION RMAGLN(500), DEG(500), U(500), T(500), B(256), A(256)
C      DIMENSION OUTFILE(7), RMAGLG(500), H(501), OUT1(7), OUT2(7)
C      DIMENSION FILEOUT(7)
C      INTEGER DB
C      PI=3.14159
C      M = 0
C      NSOO = 1      , ELIMINATES THE STORE OPTION
C      NFIRST = 0
C      NPP = 1
C      ITEND = 0
C      JCOUNT = 0
C      NDOING = 0
C      ISTARO = 0
C      IYEA = 0
C      DATA PPLT, IMR, LIN, DB/O, O, O, O/
C
C*****INPUT FILE*****
C
C      TYPE*BASICALLY THIS PROGRAM CAN PROVIDE ONE OF 3 DIFFERENT<CR>
X      OUTPUTS.  THE INPUT FILE CONTAINS A SET OF LPC PREDICTOR <CR>
X      COEFFICIENTS AND THE OUTPUT CAN BE IN THE FOLLOWING FORMS. <CR>
X      1)TEKTRONIX PLOTS OF IMPULSE RESPONSE, PHASE RESPONSE, <CR>
X      LINEAR MAG. RESPONSE, AND LOG MAG. RESPONSE FOR EACH <CR>
X      FILTER SET; <CR>
X      2)AN OUTPUT FILE CAN BE CREATED AS AN INPUT TO A 3-D <CR>
X      PLOT PROGRAM <CR> <7> "
C      ACCEPT"CREATE 3-D FILE(1-Y,0-N) ",N,NNOT
C      IF(N,NNOT.EQ.0)GO TO 36
C      ACCEPT"HOW MANY COEF. SETS ",N,LONG

```



```

ACCEPT"FILE NAME (OUTPUT)? "
READ(11,40)FILEOUT(1)
CALL DFILW(FILEOUT,1ER)
IF(1ER.EQ.13)GO TO 10
IF(1ER.NE.1)TYPE"DELETE ERROR ",1ER
10 CONTINUE
CALL CFILW(FILEOUT,2,1ER)
IF(1ER.NE.1)TYPE"CREATE LOG MAG FILE ERROR ",1ER
CALL OPEN(4,FILEOUT,3,1ER)
IF(1ER.NE.1)TYPE"OPEN ERROR ON LOG MAG OUTPUT ",1ER
ACCEPT"(1-FOR YOUR CHOICE OF START,0-IND. PLOTS)",IYEA
IF(IYEA.EQ.0)GO TO 12
ACCEPT"HOW MANY IN ".NCIN
12 CONTINUE
NYS = 0
GO TO 38

C
C***CHECK FOR PRINTRONICS PLOT

36 NYS=0
38 ACCEPT"FILTER COEFFICIENTS FILE NAME? "
READ(11,40)OUTFILE(1)
40 FORMAT(S13)
CALL OPEN(3,OUTFILE,1,1ER)
IF(1ER.NE.1)TYPE"OPEN ERROR ",1ER
READ BINARY(3)NREAD
46 READ BINARY(3)(A(1),I=2,NREAD+1)
N = NREAD
READ BINARY(3,END=9595)A0
A(1) = 1.0
B(1) = 1.0
GO TO 9596
9595 ITEND = 1
9596 CONTINUE
IF((IYEA.EQ.1).AND.((JCOUNT+1).LT.NCIN))GO TO 90

C
C*****SET UP FOR IMPULSE RESPONSE*****
C
H(1)=B(1)/A(1)
IF(N.EQ.M) GO TO 62
IF(N.LT.M) GO TO 60
DO 57 I=M+1,N
37 B(I+1)=0.
D=N
GO TO 63
60 DO 61 I=N+1,M
61 A(I+1)=0.
62 D=M
C
C*****CALCULATE MAGNITUDE, PHASE, IMPULSE RESPONSE*****
C
63 IF(NFIRST.EQ.1)GO TO 70
ACCEPT"ENTER NUMBER OF POINTS (MAX=100) ",NP
IF(NYS.EQ.1)NFIRST = 1
IF(IYEA.EQ.1)NFIRST = 1
70 DO 90 I=1,NP
Y=CMPLX(0,(PI/NP)*(I-1))
SUM1=0.
SUM2=0.
DO 72 J=1,M+1

```

```

72      SUM1=SUM1+(B(J)*(CEXP(Y*(1-J))))
      DO 74 K=1,N+1
74      SUM2=SUM2+(A(K)*(CEXP(Y*(1-K))))
      IF(SUM2.EQ.0.)TYPE"***WARNING--PROGRAM ATTEMPTING TO
X EVALUATE A POLE***"
      IF(SUM2.EQ.0) SUM2=.1E-10
      SUM3=A0*(SUM1/SUM2)
      X=REAL(SUM3)
      RMAGLN(I)=(CABS(SUM3))**2
      W=RMAGLN(I)
      IF(W.EQ.0) W=.1E-10
      RMAGLG(I)=10.*ALOG10(RMAGLN(I))
      IF(X.EQ.0) X=.1E-10
      DEG(I)=ATAN2(AIMAG(SUM3),X)
      SUM4=0.
      DO 88 L=1,I
      IF(D.GE.I)GO TO 88
      A(I+1)=0.
      B(I+1)=0.
88      SUM4=A(I-L+2)*H(L)+SUM4
      H(I+1)=(-SUM4+B(I+1))/A(I)
90      CONTINUE
      JCOUNT = JCOUNT + 1
93      IF((IYEA.EQ.1) .AND. (JCOUNT.GE.NCIN))GO TO 95
      GO TO 99
95      IF(JCOUNT.GT.(NCIN+NLONG))GO TO 175
      WRITE BINARY(4) (RMAGLG(KI),KI=1,NP)
      TYPE"WROTE OUT ROW", (JCOUNT-NCIN)

C
C*****PLOT OPTIONS*****
C
99      IFSCCL=0
      NG=1
      N=NP
      IF(NYS.EQ.1)GO TO 2000
      IF(IYEA.EQ.1)GO TO 46
      IF(NFIRST.EQ.1)GO TO 100
      ACCEPT"LINEAR MAGNITUDE PLOT? (2-CONNECTED POINTS,1-VERTICAL
X LINES,0-NO) ",LIN
100     CONTINUE
      IF(LIN.EQ.0) GO TO 106
      IF(LIN.EQ.1) MODE=1
      IF(LIN.EQ.2) MODE=0
      DO 104 I=1,N
104      T(I)=RMAGLN(I)
      CALL GRPH2("LINEAR MAGNITUDE",NG,T,U,N,MODE,YMIN,YMAX,IFSCCL)
      CALL GCHAR(ICHAR,IER)
      IF(IER.NE.1) TYPE"CONTINUE CHARACTER ERROR ",IER
106      IF(NFIRST.EQ.1)GO TO 107
      ACCEPT"LOG MAGNITUDE PLOT? (2-CONNECTED POINTS,1-VERTICAL
X LINES,0-NO) ",DB
107     CONTINUE
      IF(DB.EQ.0) GO TO 111
      IF(DB.EQ.1) MODE=1
      IF(DB.EQ.2) MODE=0
      DO 109 I=1,N
109      T(I)=RMAGLG(I)
      CALL GRPH2("LOG MAGNITUDE",NG,T,U,N,MODE,YMIN,YMAX,IFSCCL)
      CALL GCHAR(ICHAR,IER)
      IF(IER.NE.1)TYPE"CONTINUE CHARACTER ERROR ",IER

```

```

      IF((NWNOT .EQ. 1) AND. (IYEA .EQ. 0))GO TO 110
      GO TO 111
110   ACCEPT"START 3-D FILE HERE(1-Y,0-N)",NDGY
      NFIRST = 1
      IF(NDGY .EQ. 0)GO TO 46
      IYEA = 1
      IF(ISTARO .EQ. 1)GO TO 1101
      NCIN = JCOUNT
      ISTARO = 1
1101  GO TO 93
111   IF(NFIRST .EQ. 1)GO TO 112
      ACCEPT"PHASE PLOT? (2-CONNECTED POINTS,1-VERTICAL
X LINES,0-NO) ",PPLT
112   CONTINUE
      IF(PPLT.EQ.0) GO TO 116
      IF(PPLT.EQ.1) MODE=1
      IF(PPLT.EQ.2) MODE=0
      DO 114 J=1,N
114   T(J)=DEQ(J)
      CALL GRPH2("PHASE",NG,T,U,N,MODE,YMIN,YMAX,IFSCL)
      CALL GCHAR(ICHAR,IER)
      IF(IER.NE.1)TYPE"CONTINUE CHARACTER ERROR ",IER
116   IF(NFIRST .EQ. 1)GO TO 117
      ACCEPT"IMPULSE RESPONSE PLOT? (2-CONNECTED POINTS,1-VERTICAL LINES,
X 0-NO) ",IMR
117   CONTINUE
      IF(IMR.EQ.0) GO TO 170
      IF(IMR.EQ.1)MODE=1
      IF(IMR.EQ.2)MODE=0
      DO 119 I=1,N
119   T(I)=H(I)
      CALL GRPH2("IMPULSE RESPONSE",NG,T,U,N,MODE,YMIN,YMAX,IFSCL)
      CALL GCHAR(ICHAR,IER)
      IF(IER.NE.1)TYPE"CONTINUE CHARACTER ERROR ",IER
      GO TO 170

C
C***PLOT OPTION FOR PRINTRONIX
C
2000  CONTINUE
      SF = 1.0
      NPTS = NP
      DO 2010 J=1,NP
          T(J) = RMAQLQ(J)
2010  CONTINUE
      TYPE"RUNNING PLOT ",NPP,"NPTS",NPTS
      CALL PLOT10(T,NPTS,NPP,X0,Y0,SF)
      IF((ITEND .EQ. 1) AND. (NPP .NE. 5))GO TO 2015
      IF(ITEND .EQ. 1)GO TO 2012
      IF(NPP .EQ. 10)NPP=0
      NPP = NPP + 1
      AGN = 1
      GO TO 174
2012  NPP = NPP + 1
      CALL PLOT10(T,NPTS,NPP,X0,Y0,SF)
2015  NPP = 10
      CALL PLOT10(T,NPTS,NPP,X0,Y0,SF)
      GO TO 175

C
C*****REPEAT OPTION*****
C

```

```

156 IF (ITEND EQ 1) GO TO 175
170 ACCEPT "RUN AGAIN? (1=YES, 0=NO) ", AGN
    NFIRST = 1
174 IF (AGN EQ 1) GO TO 46
175 CALL CLOSE(3, IER)
    IF (IER NE 1) TYPE "CLOSE FILE ERROR ", IER
    IF (IYEA EQ 0) GO TO 211
    CALL CLOSE(4, IER)
    IF (IER NE 1) TYPE "CLOSE ERROR ON OUTPUT FILE ", IER
211 CONTINUE
    STOP
    END

```

```

C*****
C
C      THIS ROUTINE CREATES AN OUTPUT SEQUENCE OF A DIGITAL
C      FILTER WITH A SIN WAVE INPUT. THE INPUT COEFFICIENTS CAN
C      COME FROM A FILE OR BE INPUT FROM A TERMINAL.
C
C      BE SURE YOU ARE ON THE TEKTRONIX 4010-1
C      LOAD LINE:RLDR FORMANT EXPAND POLYMLT PLOT10 PLOTS.LB GRPH.LB @FLIB(
C
C      COMMENTS:
C      AEVAL-REQUIRES AN INPUT IN THE FOLLOWING FORM
C      M-ORDER OF THE NUM
C      N-ORDER OF THE DEN
C      A(I)-DENOMINATOR COEFFICIENTS
C      B(I)-NUMERATOR COEFFICIENTS
C      AO-NUMERATOR GAIN
C
C      THE INPUT FILE FOR POLES AND ZEROS IS SET UP AS FOLLOWS
C      FIRST ENTRY-NUMBER OF ZEROS(INTEGER)
C      SECOND ENTRY-NUMBER OF POLES(INTEGER)
C      ZEROS IN PAIRS-FIRST THE REAL PART
C      -SECOND THE IMAG. PART
C      POLES IN PAIRS-FIRST THE REAL PART
C      -SECOND THE IMAG. PART
C      GAIN
C
C      APR-ARRAY OF REAL ROOTS
C      API-ARRAY OF INTEGER ROOTS
C      GAIN-OVERALL GAIN
C      NPOLE OR NZER-NUMBER OF POLES OR ZEROS RESPECTIVELY
C*****
C PROGRAM FORMANT

      DIMENSION OUTFILE(7),INFILE(7),T(256),A(0:50),B(0:50),APR(50),
X      API(50),BPI(50),W(0:50),BPR(50),POLY(51),OUT2(7),Y(256)
      INTEGER MNK(256),YES
      TYPE"THE MAX COEFFICIENTS ALLOWED ARE 50 EACH"
      GAIN = 1.0
      NPOS = 0
      S = 1.0
      NFILES = 0
C
C*** ZERO INTERMEDIATE FILTER POINTS
C
      DO 15 K=0,50
        W(K) = 0.0
      15 CONTINUE
      PI = 3.1416
      ACCEPT "FILE NAME (OUTPUT)? "
      READ(11,39)OUTFILE(1)
      39 FORMAT(S13)
      CALL DFILW(OUTFILE,IER)
      IF(IER.EQ.13) GO TO 10
      IF (IER.NE.1) TYPE "DELETE ERROR ",IER
C
C*** INSERT NO. OF BLOCKS
C
      10 ACCEPT"HOW MANY BLOCKS? (30 OR LESS) ",IBLCKS
      CALL CFILW(OUTFILE,3,IBLCKS,IER)

```

```

      IF (IER.NE.1) TYPE "CREATE FILE ERROR ",IER
      CALL OPEN(1,OUTFILE,3,IER)
      IF (IER.NE.1) TYPE "OPEN ERROR ",IER
      N = 256 * IBLCKS
C
C***INPUT INFORMATION(IMPULSE FREQUENCY INPUT)
C
      ACCEPT"WHAT IS THE IMPULSES INPUT FREQUENCY? ",IFREQ1
      NPER = INT(8000.0/FLOAT(IFREQ1))
C
C*** SEE IF INPUT IS IN A FILE
C
      ACCEPT"ARE FILTER COEFICIENTS IN A FILE?(1-YES,0-NO) ",YES
      IF(YES.EQ.0)GO TO 30
      ACCEPT"FILE NAME(INPUT)? "
      READ(11,39)INFILE(1)
      CALL OPEN(3,INFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN ERROR ON INFILE ",IER
      ACCEPT"INPUT POLES AND ZEROS(1-YES,0-NO)? ",YES
      IF(YES.EQ.1)GO TO 20
C
C*** INPUT DIRECT FORM
C
      READ FREE(3) MNUM
      READ FREE(3)NDEN
      MNUM = MNUM
      NDEN = NDEN
      READ FREE(3)(B(I),I=0,MNUM)
      READ FREE(3)(A(I),I=0,NDEN)
      READ FREE(3)AO
      GO TO 58
C
C***INPUT POLES
C
20  READ FREE(3) MZER
      READ FREE(3) NPOLE
      IF(MZER.EQ.0)GO TO 21
      DO 21 I=1,MZER
        READ FREE(3) BPR(I)
        READ FREE(3) BPI(I)
21  CONTINUE
      DO 22 I=1,NPOLE
        READ FREE(3) APR(I)
        READ FREE(3) API(I)
22  CONTINUE
      READ FREE(3) AO
      MNUM = MZER +1
      NDEN = NPOLE + 1
      GAIN = 1.0
      IF(MZER.EQ.0)B(0)=1.0
C
C  FIND THE POLE POLYNOMIAL
C
      CALL EXPAND(APR,API,GAIN,NPOLE,POLY)
      AO = AO/POLY(NDEN)
      DO 25 IV=1,NDEN
        POLY(IV) = POLY(IV)/POLY(NDEN)
25  CONTINUE
      DO 24 IV=1,NDEN
        NVER = NPOLE + 2 - IV

```

```

          A(IV) = POLY(NVER)
24  CONTINUE
      GAIN = 1.0
      CALL EXPAND(BPR,BPI,GAIN,MZER,POLY)
      DO 26 IV=1,MNUM
          NVER = MNUM + 2 - IV
          B(IV) = POLY(NVER)
26  CONTINUE
      ACCEPT"SAVE THE RESULTS IN DIR. FORM (1-Y,0-N)? ",YES
      IF(YES.EQ. 0)GO TO 58
      ACCEPT"FILENAME?(OUTPUT) "
      READ(11,39)OUT2(1)
      CALL DFILW(OUT2,IER)
      IF(IER.EQ. 13)GO TO 27
      IF(IER.NE. 1)TYPE"DELETE ERROR ",IER
27  CONTINUE
      CALL CFILW(OUT2,2,IER)
      IF(IER.NE. 1)TYPE"CREATE FILE ERROR OUT2 ",IER
      CALL OPEN(2,OUT2,3,IER)
      IF(IER.NE. 1)TYPE" OPEN ERROR ON OUT2 ",IER
      WRITE(2,99)MNUM
      WRITE(2,99)NDEN
      DO 111 JB=0,MNUM
          WRITE(2,98)B(JB)
111  CONTINUE
      DO 112 JB=0,NDEN
          WRITE(2,98)A(JB)
112  CONTINUE
      WRITE(2,98)AO
98  FORMAT(1X,F20.5)
99  FORMAT(1X,I4)
      CALL CLOSE(2,IER)
      IF(IER.NE. 1)TYPE"CLOSE ERROR ON OUT2 ",IER
      GO TO 58
30  NFILES = 1
      ACCEPT"WILL THE INPUT BE POLES AND ZEROS(1-YES,0-NO) ",YES
      IF(YES.EQ. 1)GO TO 41
C
C***INPUT DIRECT FORM FROM CONSOLE
C
      TYPE "THE TRANSFER FUNCTION IS SET UP AS FOLLOWS "
      TYPE " "
      TYPE "H(Z)=AO*(B(0)+B(1)*Z**(-1)+...+B(M)*Z**(-M))/"
      TYPE "      (A(0)+A(1)*Z**(-1)+...+A(N)*Z**(-N)) "
      TYPE " "
      ACCEPT"M=? ",MNUM
      ACCEPT"N=? ",NDEN
C
C***NOTE: THE VALUES ARE SHIFTED UP IN THE ARRAY
C
      DO 35 KFIL=0,MNUM
          KD = KFIL
          TYPE"B(",KD," ) = "
          ACCEPT B(KFIL)
35  CONTINUE
      DO 40 KFIL=0,NDEN
          KD = KFIL
          TYPE"A(",KD," ) = "
          ACCEPT A(KFIL)
40  CONTINUE

```

```

ACCEPT"A0? = ",A0
GO TO 58

C
C***INPUT POLES AND ZEROS FROM CONSOLE
C
41  B(0) = 1.0
    ACCEPT"HOW MANY POLES? ",NPOLE
    ACCEPT"HOW MANY ZEROS? ",MZER
    DO 42 KFIL=1,NPOLE
        ACCEPT"REAL PART OF POLE= ",APR(KFIL)
        ACCEPT"IMAG PART OF POLE= ",API(KFIL)
42  CONTINUE
    IF(MZER .EQ. 0)GO TO 43
    DO 43 KFIL=1,MZER
        ACCEPT"REAL PART OF ZERO= ",BPR(KFIL)
        ACCEPT"IMAG PART OF ZERO= ",BPI(KFIL)
43  CONTINUE
    ACCEPT"GAIN = ",A0
    MNUM = MZER
    NDEN = NPOLE
    GAIN = 1.0
    IF(MZER .EQ. 0)MNUM=0
X   TYPE"RUN EXPAND POLES"
    CALL EXPAND(APR,API,GAIN,NPOLE,POLY)
X   TYPE"RAN EXPAND POLES"
    A0 = A0/POLY(NDEN+1)
    DO 44 IV=1,NDEN+1
        POLY(IV) = POLY(IV)/POLY(NDEN+1)
44  CONTINUE
    DO 45 IV=1,NDEN+1
        NVER = NPOLE + 2 - IV
        IV5 = IV - 1
        A(IV5) = POLY(NVER)
45  CONTINUE
X   DO 47 MN5=0,NDEN
X       TYPE"MN5 = ",A(MN5)
X 47  CONTINUE
    GAIN = 1.0
    IF(MZER .EQ. 0)GO TO 46
X   TYPE"RUN EXPAND ZEROS "
    CALL EXPAND(BPR,BPI,GAIN,MZER,POLY)
X   TYPE"RAN EXPAND ZEROS"
    IF(MZER .EQ. 0)GO TO 46
    DO 46 IV=1,MNUM+1
        NVER = MNUM + 2 - IV
        IV5 = IV - 1
        B(IV5) = POLY(NVER)
46  CONTINUE
    IF(MZER .NE. 0)GO TO 48
    B(1) = 1.0
48  ACCEPT"SAVE RESULTS IN DIRECT FORM?(1-Y,0-N) ",YES
    IF(YES .EQ. 0)GO TO 58
    ACCEPT"FILENAME(OUTPUT)? "
    READ(11,39)OUT2(1)
    CALL DFILW(OUT2,IER)
    IF(IER .EQ. 12)GO TO 56
    IF(IER .NE. 1)TYPE"DELETE ERROR ",IER
56  CONTINUE
    CALL CFILW(OUT2,2,IER)
    IF(IER .NE. 1)TYPE"CREATE FILE ERROR ",IER

```



```

CALL OPEN(2,OUT2,3,IER)
IF(IER.NE.1)TYPE"OPEN ERROR ",IER
WRITE(2,99)MNUM
WRITE(2,99)NDEN
DO 1110 JB=0,MNUM
    WRITE(2,98)B(JB)
1110 CONTINUE
DO 1120 JB=0,NDEN
    WRITE(2,98)A(JB)
1120 CONTINUE
WRITE(2,98)A0
CALL CLOSE(2,IER)
IF(IER.NE.1)TYPE"CLOSE ERROR ",IER
58 CONTINUE
DO 75 J=1,IBLCKS
    K = J - 1
    DO 65 L = 1,256
        IF(NPOS.NE.NPER)GO TO 59 ;SETS AN IMPULSE AT INPUT FREQ
        S = 1.0
        NPUS = 0
59 CONTINUE
        NPOS = NPOS + 1
        TEMP = 0.0
        DO 60 KV=1,NDEN
            TEMP=-A(KV)*W(KV)+TEMP
60 CONTINUE
            W(0)=TEMP+S*A0
            TEMP1 = 0.0
            DO 61 KV=0,MNUM
                TEMP1=B(KV)*W(KV)+TEMP1
61 CONTINUE
            IF(NDEN.LT.MNUM)GO TO 62
            MEND = NDEN
            GO TO 63
62 MEND = MNUM
63 DO 64 IKR=1,MEND
            MKD = MEND + 1 - IKR
            W(MKD) = W(MKD) - 1)
64 CONTINUE
            W(0) = 0.0
            Y(L) = TEMP1
            S = 0.0
            MNK(L) = INT(TEMP1)
65 CONTINUE
        CALL WRBLK (1,K,MNK,1,IER)
        IF (IER.NE.1) TYPE "WRITE BLOCK ERROR ",IER
        IF (J.NE.1) GO TO 75
        DO 70 L=1,256
            T(L)=Y(L)
70 CONTINUE
75 CONTINUE
X TYPE"COMPLETED OUTPUTING THE IMPULSE RESPONSE "
CALL CLOSE(1,IER)
ACCEPT"DO YOU WANT A PLOT?(1-Y,0-N) ",NYES
IF(NYES.NE.1)GO TO 5600
ACCEPT"USE PRINTRONICS PLOTTER?(1-Y,0-N) ",NO
IF(NO.EQ.0)GO TO 3000
NP = 1
SF = 1.0
NPTS = 256

```

```

      CALL PLOT10(T,NPTS,NP,XO,YO,SF)
      NP = 10
      CALL PLOT10(T,NPTS,NP,XO,YO,SF)
      GO TO 5600
3000  IFSCCL = 0
      MODE = 0
      NG = 1
      N = 256
      CALL GRPH2(OUTFILE,NG,T,U,N,MODE,YM,YA,IFSCCL)
      IF(NFILES.EQ. 1)GO TO 5600
      CALL CLOSE(3,IER)
      IF(IER.NE. 1)TYPE"CLOSE ERROR ".IER
5600  CONTINUE
      STOP
      END

```

```

      SUBROUTINE EXPAND(ROOTR,ROOTI,GAIN,NF,POLY)
C*****
C
C   THIS SUBROUTINE EXPANDS ROOTS INTO A POLYNOMIAL
C
C   INPUTS: ROOTR -THIS IS A SET OF REAL ROOTS
C            ROOTI -THIS IS A SET OF IMAGINARY ROOTS
C            GAIN  -SYSTEM GAIN
C            NF    -IS A CHECK FOR NO FILTEN (FOR NF=0)
C
C   OUTPUT: POLY  -THIS IS THE OUTPUT POLYNOMIAL
C
C*****
      DIMENSION ROOTR(50),ROOTI(50),POLY(51),BPOLY(51),APOLY(3)
      CLOSE = 0.0000001
      DO 2 I=1,51
        POLY(I) = 0
        IF(NF .NE. 0)GO TO 4
        POLY(I) = GAIN
        RETURN
      4   I=1
        IF(ABS(ROOTI(I)).GT.CLOSE)GO TO 10
        NBP=1
        BPOLY(1)=1
        BPOLY(2)=-ROOTR(I)
        IF(NBP.EQ.NF)GO TO 800
        I=I+1
        GO TO 20
      10  NBP=2
        BPOLY(1)=1
        BPOLY(2)=-2*ROOTR(I)
        BPOLY(3)=ROOTR(I)**2+ROOTI(I)**2
        IF(NBP.EQ.NF)GO TO 800
        I=I+2
        GO TO 20
      20  IF(ABS(ROOTI(I)).GT.CLOSE)GO TO 30
        NAP=1
        APOLY(1)=1
        APOLY(2)=-ROOTR(I)
        GO TO 40
      30  NAP=2
        APOLY(1)=1
        APOLY(2)=-2*ROOTR(I)
        APOLY(3)=ROOTR(I)**2+ROOTI(I)**2
      40  CALL POLYMLT(APOLY,BPOLY,POLY,NAP,NBP,NP,1,0,1,0)
        N= NP+1
        IF(NP.GE.NF)GO TO 990
        DO 45 J=1,N
          BPOLY(J)=POLY(J)
          NBP=NBP
          I=I+NAP
          GO TO 20
      880 DO 881 I=1,3
      881 POLY(I)=BPOLY(I)
          N=NBP+1
      990 DO 991 I=1,N
      991 POLY(I)=POLY(I)*GAIN
        RETURN
      END

```

```

SUBROUTINE POLYMLT(A, B, C, NA, NB, NC, AK, BK, CK)
C*****
C
C
C   THIS SUBROUTINE COMPUTES A POLYNOMIAL
C
C*****
      DIMENSION A(51), B(51), C(51)
      NC=NA+NB
      IF(NC.LE.50)GO TO 2
      TYPE'DEGREE OF RESULT IS GREATER THAN 50 '
      RETURN
2     DO 1 I=1, 51
1     C(I)=0
      NAA=NA+1
      NBB=NB+1
      DO 3 I=1, NAA
      DO 3 J=1, NBB
3     C(I+J-1)=C(I+J-1)+A(I)*B(J)
      CK=AK*BK
      RETURN
      END

```

APPENDIX C

User Manual

The user manual is divided into five parts. The first part will contain the run lines for each of the vocoder parts including the pitch detector, speech analyzer and speech synthesizer. The run line will contain generic file names such as FILE1, etc.

The second, third and fourth parts will contain all the information required to run the pitch detector, speech analyzer and speech synthesizer respectively. The final section will describe how to use the utility programs which convert various vocoder information into convenient analysis forms such as plots.

Run Lines

The run lines for the vocoder will be shown below using FILE# as a generic file. The user can substitute any legal file name (see Data General user manuals for details) for a FILE#. Once a particular file number has been chosen below (for example FILE5) that file with number will always refer to the same file. This was done so that it would be clear when a particular file is used. Following the run lines a description of the files will be given so that the user can substitute his own files in cases where programs other than those created in this thesis are used. For example, any

other pitch detector could be used provided the pitch period file was of the same format. The run line for the AUTO pitch detector (program PITCH2) is as follows:

```
PITCH2 FILE1/I FILE2/O FILE3/D FILE4/F
```

The run line for the SIFT pitch detector is:

```
SIFT FILE1/I FILE2/O FILE3/D
```

The run line for the speech analyzer (program SIFT) is:

```
CODER FILE1/I FILE3/D FILE5/P
```

The run line for the speech synthesizer is:

```
SYNTH FILE3/D FILE5/P FILE6/S FILE2/O
```

Any of the four programs including PITCH2, SIFT, CODER or SYNTH can be run by typing in their respective run line on the Data General's command line.

A description of each file's content and how it is formatted is given below:

FILE1: This file contains the input speech that will be processed by the vocoder. The speech is an integer file in a binary format (requiring either a binary read or a read block to input the data). The data is in two's complimentary form with an absolute maximum of 32,768. FILE1 is used as an input to both pitch detectors and the speech analyzer.

FILE2: This file contains a silence/speech indicator, a voiced/unvoiced indicator if speech and the pitched period for a voiced speech segments. One record contains three integers which is the pitch information for each 80 speech samples. The first integer in the record is a one if the

speech segment is voiced and a zero if unvoiced or silence. The second integer is a one if the speech segment is speech or a zero for silence. The third integer is the pitch period and ranges in values from 0 to 160. If the speech segment is not voiced then the pitch period will equal zero. The Fortran format for each record is:

```
FORMAT (3X,3(I10.3X))
```

FILE2 can be an output of either pitch detector and an input to both the speech analyzer and synthesizer.

FILE3: This file contains the initialization parameters that are used by all the vocoder programs. The file is composed of 50 records where the first 25 are real and the second 25 are integers. Not all of the records are used, but are available for future expansion. The program SETUP (to be described in the utility section) is used to update this file. What follows is a list of the records and what initialization parameters they represent:

(Real Records)

- 1) The clipping level used in program PITCH2;
- 2) The autocorrelation threshold used in program PITCH2;
- 3) The voiced/unvoiced threshold used in PITCH2;
- 4) The variable alpha (the window shape) used in CODER;
- 5) The speech scale used in CODER;
- 6) The silence threshold used in SIFT;
- 7) The unvoiced to voiced gain used in SYNTH;

- 8) The glottal pulse positive slope size used in SYNTH;
- 9) The glottal pulse negative slope size used in SYNTH;
- 10) The silence threshold used in PITCH2;
- 11) thru 25) are not currently used.

(Integer Records)

- 26) The number of points in each 1/3 frame used in PITCH2;
- 27) The number of points between each filter update used in CODER and SYNTH;
- 28) The pre/de-emphasis switch (1-if used, 0-if not) used in CODER and SYNTH;
- 29) The glottal pulse or impulse switch (1-for glottal pulse, 0-for impulse) used in CODER and SYNTH;
- 30) thru 50) are not currently used.

FILE4: This is a file of filter coefficients that are the output of a program WFILTER which is a user program on the Data General System. These filter coefficients are used in the input low pass filter (LPF) in PITCH2. The first record (read free's are used so location on each line of the file doesn't matter) is the integer number of zeros in the LPF and is always assumed to be zero. The second record is always the integer zero. The maximum number of zeros allowed are 120, but 99 are usually used. The rest of the records in the file are real filter coefficients.

FILE5: This file contains the LPC predictor coefficients which are the output of the speech analyzer (CODER) and the input to the speech synthesizer (SYNTH). The file is a binary file where the first value is the integer order (p) of the predictor coefficients. The rest of the file consists of sets of real predictor coefficients with p predictor coefficients per set.

FILE6: This file contains the output speech produced by the speech synthesizer (SYNTH). This is a binary file consisting of integers.

Pitch Detectors

How to run the pitch detectors has been described in the run line section of this thesis. In addition, the way the software is organized and algorithm descriptions were given in appendix A and chapter III respectively. This section will provide the load lines and a short description of each subroutines primary function.

The load line for AUTOC is:

```
RLDR PITCH2 IOF LPF ENERGY SETMAX CLPPER AUTCOR @FLIB@
```

A short description of each subroutine follows:

IOF-This subroutine inputs the file names on the load line and makes them available to the program PITCH2.

LPF-This subroutine low pass filters the input speech (usually at 900 Hz).

ENERGY-This subroutine calculates the energy in a frame and compares it to a silence threshold. If the energy exceeds the threshold then it is considered speech otherwise

it is silence.

SETMAX-This subroutine finds the maximum value in the first and last third of the frame. The smallest of these two values is the clipping level.

CLPPER-This subroutine does infinite peak and zero clipping. Any value larger or smaller than the clipping level or its negative, respectively is set to a -1.0 if negative or 1.0 if positive. Any values in between are set to 0.0.

AUTCOR-This subroutine calculates autocorrelation for a frame then finds the fundamental peak.

The load line for SIFT is:

```
RLDR SIFT SIFTA DIRECT AUTO ENER IOF @FLIB@
```

A short description of each subroutine follows:

SIFTA-This subroutine performs inverse filtering on the speech.

DIRECT-This subroutine implements a direct form filter and is called by SIFTA for low pass filtering the input speech.

AUTO-This subroutine calculates the autocorrelation function and is called by SIFTA.

ENER-This subroutine computes the energy in a frame and compares it to a silence threshold. If the energy exceeds the threshold then it is considered speech otherwise it is silence.

IOF-This subroutine inputs the file names on the load line and makes them available to the program SIFT.

Speech Analyzer

How to run the speech analyzer has been described in the run line section of this thesis. The way the software was organized and the algorithms described was given in appendix A and chapter III respectively.

This section will provide the load lines and a short description of each subroutines primary function.

The load line for CODER is:

```
RLDR CODER RLPC INFIL IOF @FLIB@
```

A short description of each subroutine follows:

RLPC-This subroutine recursively calculates the autocorrelation of the input speech.

INFIL-This subroutine inverts the autocorrelation function to produce a set of predictor coefficients.

IOF-This subroutine inputs the file names on the load line and makes then available to the program CODER.

Speech Synthesizer

How to run the speech synthesizer has been described in the run section of this thesis. The way the software was organized and the algoritms described was given in appendix A and chapter III respectively. This section will provide the load line and a short description of each subroutines primary function.

The load line for SYNTH is:

```
RLDR SYNTH VOICED THROAT UNVOC D IOF @FLIB@
```

A short description of each subroutine follows:

VOICED-If a segment of speech has been determined to be

voiced this subroutine produces a glottal pulse or an impulse.

THROAT-This subroutine uses as an input the glottal pulse for voiced speech or noise for unvoiced speech. The appropriate input is put through a digital filter whose filter coefficients are the current LPC predictor coefficients.

UNVOCD-If a segment of speech has been determined to be unvoiced this subroutine outputs normally generated noise.

IOF-This subroutine inputs the file names on the load line and makes them available to the program SYNTH.

Utility Programs

The utility programs described in the user manual are interactive and don't use load lines for inputting file names. The only requirement to run these programs is to type in the program name, then carriage return. Some of the files described earlier (for example, FILE2) will be used here because the utility programs operate on various vocoder output files. The five utility programs used are SCALE1, PLTTER, SETUP, KEVAL and FORMANT.

The first program, SCALE1, uses the vocoder synthesized speech output and scales the speech to the proper magnitude for use with the D/A converter. The second program, PLTTER, either plots the pitch detector output or a speech file. The third program, SETUP, allows the user to update a file of parameters that are the initialization values for the vocoder. The fourth program, KEVAL, lets the user create a

file which is an input file for the 3-D plotter program PLOT3D. Note: PLOT3D was not created in this thesis; for details on using it see the book of user programs. The final program, FORMANT, allows the user to create an artificial vowel using poles or a set of filter coefficients.

SCALE1 requires one input speech file which can either be FILE1 or another speech file formatted in the same manner. The output is a speech file that has been scaled to a maximum of 2000.0. One additional option allows uniformly distributed noise to be added to the speech file. The speech file plus noise is still scaled to a maximum of 2000.0.

What follows is an example of a console session using SCALE1 with noise being added to a speech file (the users entries are in parenthesis and added comments are in quotes):

(SCALE1)

WARNING: THE INPUT FILE MUST BE AN INTEGER FILE AND BE IN BLOCKED FORM.

DO YOU WISH TO CONTINUE? (1-Y, 0-N)

(1)

INPUT FILENAME:

(FILE1)

OUTPUT FILENAME:

(FILEOUT)

OUTPUT FILE SIZE?

(88) "This is in blocks and is the maximum
size allowed by AUDIOHIST"

PERFORM NOISE ADDITION? (1-Y, 0-N)

(1)

SIZE OF MAX NOISE? (REAL)

(500.0)

THE FOLLOWING NO. OF POINTS WERE CHECKED 22528

AND THE MAX VALUE FOUND WAS 4800

If the program had been run without noise the only difference would be that the question "SIZE OF MAX NOISE?" would not appear.

The next program to be described is PLTTER. One input file is required and the filename entered at the console. This file can be either the output file of a pitch detector for a pitch plot or a scaled speech file (maximum of 2000.0.) for a speech plot. If a pitch plot is required two plots are output on the PRINTRONIX printer. The top plot is a plot of the pitch period and the bottom is a plot of silence (a "1") or not silence (a "0"). The second possible output is a plot of the speech file. Each page has ten plots of speech except the last page which has the required number of plots to plot the remaining speech.

What follows is an example of a console session using PLTTER:

(PLTTER)

FILE NAME?

(FILE2)

FILE OUTPUT FROM PITCH? (1-Y, 0-N)

(1)

In the example above a pitch plot would have been produced. If a 0 had been entered instead of a 1 a speech plot would have been produced.

The next program, SETUP, requires one input file and either uses the input file as an output or uses an additional file for output. The purpose of this program is to allow the user to change the initial conditions of the vocoder by changing the initial conditions file. The input file is a file already containing initial conditions. If a new file is to be created the program sets all values to zero and then the user inputs initial conditions from the console. A list of the initial conditions can be output to the PRINTRONIX printer and/or the console. Finally the initial conditions can overwrite the input file or be put in a new file.

What follows is an example of a console session using SETUP:

(SETUP)

OUTPUT AND/OR INPUT FILE CURRENTLY EXIST? (1-YES, 0-NO)

(1) "All input and output files must exist before
using this program"

FILE NAME?

(PARAM) "This is the file containing initialization
data or the new file to be created if starting from scratch"

IF YOU CHOOSE TO CHANGE A VARIABLE ENTERY OTHERWISE DO

A CARRIAGE RETURN

FILTER COEFFICIENTS FILE NAME?

(FILE5) "The output file from the speech analyzer"

ENTER NUMBER OF POINTS (MAX=100)

(50) "This is the number of points to each log
magnitude plot"

"At this point the output file is created. What follows
is a continuation of the other choice if a 3-D plot isn't
wanted."

CREATE 3-D FILE (1-Y, 0-N)

(0)

THE CURRENT VALUE OF CLPLEV (in PITCH2) IS: .6

CHANGE VALUE?

(CR) "carriage return"

CURRENT AUTOCORRELATION THRESHOLD (in PITCH2) IS: .3

CHANGE VALUE?

(Y)

INPUT NEW VALUE

(.35)

THE CURRENT VALUE OF VOICED/UN THRESH

(in PITCH2) IS: 20.0

"This questioning for each record continues until all
the records have been typed out. For a complete list see
the description of FILE3 given earlier in this appendix.
What follows is the rest of this session after all records
have been viewed or changed."

DO YOU WANT TO HAVE THE ARRAY TYPED (1-YES, 0-NO)

(0) "If a 1 would have been chosen the values
would have been typed on the console."

IS THE INPUT FILE THE OUTPUT FILE

(1) "This will write the updated values out to the
input file. A 0 response would have had the
values written out to a new file. In that
case the user would enter a file name for the
output file"

PRINT ARRAY ON PRINTRONIXS? (1-Y, 0-N)

(0) "A 1 choice produces a plot on the
PRINTRONIX of the data"

PROGRAM COMPLETED

The next program, KEVAL, has one input file and if the
option is selected on output file. The input file is a file
of predictor coefficients output from the vocoder. The
description of this file's format is given the description
of FILE5.

What follows is an example of a console session using
KEVAL:

(KEVAL)

BASICALLY THIS PROGRAM CAN PROVIDE ONE OF 3 DIFFERENT
OUTPUTS. THE INPUT FILE CONTAINS A SET OF LPC PREDICTOR
COEFFICIENTS AND THE OUTPUT CAN BE IN THE FOLLOWING FORMS.

1) TEKTRONIX PLOTS OF IMPULSE RESPONSE, PHASE RESPONSE,
LINEAR MAG. RESPONSE AND LOG MAG. RESPONSE FOR EACH FILTER
SET;

2) AN OUTPUT FILE CAN BE CREATED AS AN INPUT TO A 3-D

PLOT PROGRAM.

CREATE 3-D FILE (1-Y, 0-N)

(1) "At this point it will be assumed the user
wants a 3-D plot. The other case will be
treated later"

HOW MANY COEF. SETS

(50) "This means 50 log magnitude plots will be
output"

FILE NAME (OUTPUT)?

(FPLOT) "This is the input file for PLOT3D"

FILTER COEFFICIENTS FILE NAME?

(FILE5)

ENTER NUMBER OF POINTS (MAX=100)

(50) "This is how many points will be on each plot"

LINEAR MAGNITUDE PLOT? (2-CONNECTED POINTS, 1-VERTICAL
LINES, 0-NO)

(2) "A plot of connected points will appear on
the TETRONIX 4010 screen. Choosing a 1 would
produce a plot of vertical lines and 0 would
not provide a plot. After the user is
completed with the plot any key may be pressed
to continue."

"The next plot is the log magnitude plot. Then a phase
plot, and finally the impulse response plot. For each of
these plots the options 0 thru 2 are given. After viewing
the plots of interest the next console question follows."

RUN AGAIN? (1-YES, 0-NO)

(0) "The program ends"

If a 1 had been chosen, each of the plots picked the first time will be shown for each subsequent set of filter coefficients. This was done to avoid having to answer questions between plots. If the user does not wish to observe more plots enter 0 after the RUN AGAIN? question.

The final utility is FORMANT which is a program that was developed originally to create a file which is the output of an artificial vowel that is impulsed at a constant frequency. An input file can be used to either enter the vowel's filter coefficients in direct form or as poles and zeros. If the input is a file of poles and zeros then these are converted to coefficients in the direct form.

What follows is a console session using FORMANT:

THE MAX COEFFICIENTS ALLOWED ARE 50 EACH

FILE NAME (OUTPUT)?

(FVOWEL) "This is the file the artificial vowel
will be put in"

HOW MANY BLOCKS? (30 OR LESS)

(20) "FVOWEL will be 20 blocks long"

WHAT IS THE IMPULSE INPUT FREQUENCY?

(100) "Once every 100 samples the filter will be
impulsed"

ARE THE FILTER COEFFICIENTS IN A FILE (1-YES, 0-NO)?

(1)

FILE NAME (INPUT)?

(INFILE)

INPUT POLES AND ZEROS (1-YES. 0-NO)

(1)

"A pole zero file is in a record format. The first record is the number of zeros (z) and the second is the number of poles (p). The next z records are the zeros and then the next p records contain the poles."

SAVE RESULTS IN DIR. FORM (1-Y, 0-N)

(0) "If saved in direct form the file is configured to be an input to Kathy Wards program EVAL in the user's book"

COMPLETED OUTPUTTING THE IMPULSE RESPONSE

DO YOU WANT A PLOT? (1-Y, 0-N)

(1)

USE PRINTRONIX? (1-Y, 0-N)

(1) "The impulse response will be plotted on the PRINTRONIX. A choice of 0 would plot the impulse response on the TEKTRONIX 4010-1"

When the question asked ARE THE FILTER COEFFICIENTS IN A FILE answering no requires the user to enter data from the console. This method is self explanatory since the program asks for the data item by item.

VITA

Willis Janssen was born on 27 October 1951 in Minneapolis Mn. He graduated from Central High School in Pipestone Mn in 1969. From August of 1969 to May of 1973, he attended Southwest Minnesota State University where he earned a degree of Bachelor of Arts in physical science. Then for two years he taught physics, chemistry and mathematics at The Putney School, Putney Vt. In 1975 he joined the United States Air Force and worked at the Air Force Weapons Laboratory. During this time he attended the University of New Mexico. Where in 1979 he received the degree of Bachelor of Science in Electrical Engineering. Next, in Sept 1979 he received a commission in the United States Air Force. For the next three years he was assigned to the 552 Airborne Warning and Control Wing (AWACS) where he worked as a systems analyst. He then entered the School of Engineering of the Air Force Institute of Technology.

Permanent Address:RFD3 Box 99A

Putney Vt. 05346

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/83D-33	2. GOVT ACCESSION NO. 3. RECIPIENT'S CATALOG NUMBER A138008	
4. TITLE (and Subtitle) A RECURSIVE LINEAR PREDICTIVE Vocoder		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
7. AUTHOR(s) Willis A. Janssen Capt.		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE December, 1983
		13. NUMBER OF PAGES 360
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Approved for public release: LAW AFR 100-17. LYNN E. WCLAVER 7 Feb 84 Dean for Research and Professional Development Air Force Institute of Technology (A1C) Wright-Patterson AFB OH 45433		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Linear Predictive Coding (LPC) Vocoder		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A non-real time 10 pole recursive autocorrelation linear predictive coding vocoder was created for use in studying effects of recursive autocorrelation on speech. The vocoder is composed of two interchangeable pitch detectors, a speech analyzer, and speech synthesizer. The time between updating filter coefficients is allowed to vary from .125 msec between each update. The greatest change in quality was noted when changing from 20 msec/update to 10 msec/update		

Pitch period plots for the center clipping autocorrelation pitch detector (AUTOC) and simplified inverse filtering technique (SIFT) are provided. Plots of speech into and out of the vocoder are given. Formant versus time 3-D plots are shown.

Effects of noise on pitch detection and formants are shown. Noise effects the voiced/unvoiced decision process causing voiced speech to be re-constructed as unvoiced.

END

FILMED

384

DTIC