CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

Canada

A Microprocessor-Based Hearing Aid Device for Selected Classes of Sound Signals (A Feasibility Study)

Paraskevas Ioannou Michalopoulos

A Thesis

in

The Department

of

Electrical Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering at
Concordia University
Montréal, Québec, Canada

March 1986

ISBN 0-315-30669-6

ABSTRACT

A Microprocessor-Based Hearing Aid Device
for Selected Classes of Sound Signals
(A Feasibility Study)

Paraskevas Ioannou Michalopoulos

Παρασκευάς Ἰωάννου Μιχαλόπουλος

A feasibility study on a microprocessor-based device to help the hearing impaired persons has been conducted. The device should recognize selective classes of sound signals so as to help the hearing impaired persons to understand emergency sounds as well as to facilitate them in their communication with other persons.

Although emphasis was given in the design of the back end of the recognizer, the front end was not neglected. For the latter a user's control device was constructed whose purpose include the interfacing of the microphone output to the ADC input. The former is based mainly on the pitch comparison of the signals, supported by time domain elements such as the duration of the sound signals and by autocorrelation domain elements such as the absolute energy in the signals.

The algorithm (software) to be included in the device has been tested in a general purpose computer under various environmental conditions. The sound patterns that the device should be able to recognize include time-invariant sounds as well as sounds which are repetitive in time.

Finally, the computer requirements in memory and speed for the construction of the device are outlined.

Table of Contents

List of Figures

List of Symbols

Symbol	Title	Page
α	Coefficient of the Preemphasis Filter	41
a	LP Autocorrelation Coefficient	13
$\mathbf{a}$	LPAC's Vector	58
$e(n)$	Error of Estimation for a Sample	13
$E\{\}$	Expected Value	9
E_t	Error of Estimation for a Frame	14
E_m	Residual (min. error of estimation)	21
f_N	Nyquist Frequency	8
L	Resolution of PSD	62
N	Frame Length in Samples	17
O	Order of	59
p	Order of Estimation	10
$P(\omega)$	PSD of the Signal	27
$P'(\omega)$	Estimated PSD	28
$\mathbf{r}$	Vector of Autocorrelation Coefficients	58
$R[]$	Autocorrelation Function	9
$R()$	Autocorrelation Coefficient	18
$\mathbf{R}$	Autocorrelation Matrix	58
$s()$	Sample of the Signal	8
$s'()$	Estimated Sample	13
$w()$	Window Function	18
ω_s	Sampling Frequency	28
Δ	Determinant	59

Table of Abbreviations

ADC	Analog-to-Digital Converter
AR	AutoRegressive
ARMA	AR with Moving-Average Error
DFFT	Discrete FFT
FP	Floating Point
FPIPS	FP Instructions per Second
FRP	FP Processor
FS	Full Scale
FFT	Fast Fourier Transform
FT	Fourier Transform
FPIPS	Floating Point Instructions Per Second
FPP	Floating Point Processor
LP	Linear Prediction
LPAC	LP Autocorrelation Coefficients
MS	Mean Square
MSB	Most Significant Bit
PSD	Power Spectral Density
RV	Random Variable
SUR	Statistical Uncertainty of Results
TF	Transfer Function

CHAPTER 1

INTRODUCTION

The main objective of the present thesis is the search for an efficient algorithm to be implemented in a digital hearing aid device. Two additional objectives are: first, the testing of the algorithm, and second, the estimation of the computer memory and speed required for the implementation of the algorithm.

The work in this thesis is basically a feasibility study for a device which would recognize classes of sound signals that include such repetitive mechanical sounds as the telephone ring, or sounds which border noise in their characteristics and which include a repetitive knock, a series of steps, etc.

The hearing aids presently used for the hearing impaired persons are analog devices. These devices act as simple linear amplifiers of the incoming sound signals or they filter and sift the spectrum of the signal in a range where it can be audible by the hearing impaired person (carrier) [15]. The fundamental assumption in designing these analog devices is that the carrier still possesses the hearing sense but his dynamic range has been reduced. No general purpose device exists for aiding the hearing impaired persons and which device identifies the sound signals of the environment and gives the results of the identification with other means than sounds.

An ideal hearing aid device should 'understand' all the sound signals it receives, and it should pass the results of the 'understanding' to the carrier of the device. In addition, such a device should be easy to use, be cost effective, be real time, and be accurate. Such an ideal device, it seems, can not be built with the present knowledge and advancement of the technology. It is understood that

such an ideal device should be a complete substitute of the functions of the human ear, and of some functions of the human brain.

The first question asked before we attempted to formulate any problem was: Is there a systematic overall theory of hearing? E.C. Carterette [10] mentions that there are at least 21 'partial' theories. Of these theories, none is capable of completely explaining the physics of the hearing sensation, and thus none is generally accepted. It has been concluded [51], that there is not such an overall theory. The main reason for the absence of such a general theory is that the ear is a complex organ, and that the ear-brain communication is also as complex.

An important observation at this point is that audition is not a simple frequency analysis [10]. The whole process can be visualized as a peripheral analysis of the sound signals done by the ear, which is complemented by a central synthesis (classification) done in the left cerebral cortex. Apparently the ear is capable of analyzing compound sounds into complex, rather than simple, tones and of passing the results of this analysis to the brain [42].

Because of the complexity of the problem, it has been suggested, in 1939 [41], that it is not possible to construct a general purpose hearing aid device. The main argument for this suggestion is that by breaking down the sound signals into binary form, information critical to the ear in understanding the sound is lost for ever. For almost a decade afterwards research into the field of the imitation of the hearing sense came to almost a standstill. The new generation of microprocessors and peripherals, much superior to the previous generation both in speed and storage capability, gave a rebirth to this research. Another factor in the rebirth was the application in the sound recognition schemes of new coding techniques of the data, namely the linear prediction coding.

Having partially settled above the question of the overall theory of hearing,

the next step will be the understanding of some facts referring to the ear which have been established as correct. The ear is inherently non-linear and combination tones have real effect (i.e. the various components of a sound can be distinguished) [10]. A normal ear analyzes sound signals ranging in frequency from ~ 100 Hz to ~ 20 KHz [34]. The analysis of the various frequencies is done in spatially different parts of the ear [42,51]. It is generally accepted that frequencies above 5 KHz may be discarded without any significant loss in audibility of the signal [16,49]. If the cut-off frequency is ~ 3.3 KHz, as is the case when the carrier of the signal is a telephone line, then some loss of intelligibility has been observed [34]. Another fact is that the ear analyzes four components of the sound signals [9,16,42]. The first component is that of loudness, which represents the relative level of energy as well as the distribution of energy in a sound. The second component is that of roughness of the sound, which corresponds to the sound fluctuations. The third component is the pitch, which is an analysis of the frequencies in a sound. The last component is the sharpness or the quality of the sound signal, which corresponds to the characteristics of the spectral envelope of the signal. A conclusion of the above analyzing powers of the ear is that the ear is an organ which in combination with the brain is capable of analyzing sound signals in time domain, in frequency domain, and in autocorrelation (power) domain, all at the same time, or parallelly at a very short period of time.

How the results of the peripheral analysis of sounds, performed by the ear, are passed to the left lobe of the human brain for further processing is quite unknown. It is generally accepted that this communication is done in a complex and high level [42], i.e. the information passed is in compound form.

The above preliminary investigation referring to some aspects of the hearing sense limited the goals of our investigation. The algorithm to be investigated

should recognize two specific classes of sound signals. In the first class we find the almost time-invariant signals. Examples of these signals include a door-bell, the ring of a telephone, an emergency siren, a fire alarm, etc. The common characteristic of this class of sounds is that the sound generator upon being excited remains in the same excitation state, after a short transition period, for a sufficiently long period of time. In the second class, we find signals which are repetitive in time, other than the signals belonging to the first class. Examples of such signals are: the knock on a door, steps approaching, the cry of a baby, etc. The common characteristic of this class of signals is that they are time-variant, and that they have large transit ratio [transition period/stabilization period].

Considering that the algorithm is of limited capabilities, care was taken so that it could be easily expandable to other classes of signals. In addition to expandability, other desired characteristics of the algorithm include: precision, speed, limited computer memory required, and finally, ease of implementation in a digital computer environment.

For the formulation of the algorithm, a model building stage was necessary. This is done in chapter 2. In this chapter, an analysis of the overall problem is presented. The main questions to be answered were: what can be a realistic model? and, what can be a good realistic model? Basically, the work done in this chapter has a dual purpose. The first purpose is the smoothing of the signal, and the second is its parametrization. Both can be accomplished by using almost identical means. The parametrization component is necessary considering that, from an infinite set of inputs, we attempt to recognize (label) a certain (finite) number of inputs. The possible combinations a recognition algorithm should explore, if not parametrization is done, is certainly infinite. With the parametrization (estimation) of the signal the combinations required by the algorithm to possess is at most as many as the number of classes of sounds to be recognized.

The other topic in this chapter is the smoothing of the signal. It is highly desirable, for any recognition algorithm, to deal with signals which have been smoothed up to a certain level. The smoothing process is an invariably used process in all signal processing algorithms.

The parametrization of the signal can be done basically by using Fast Fourier Transform (FFT) or Linear Prediction (LP) coding. We have chosen the latter for two main reasons. The first reason is that in FFT a small error in the input data may result in a large error in the estimation of the data. This is a common problem in Fourier analysis, and it may stand true even when the signal-to-noise ratio is large. It is assumed that the error is due to some kind of noise [38]. The second reason is that the smoothing of the signal can be done better by using LP coding than FFT, and the spurious responses of the incoming signal are taken care of by using LP coding. Additional supportive factors for using LP coding is its accuracy and its speed of computation [50].

The theoretical formulation of the problem (model building) was necessary to be included in this thesis for the additional reason of understanding the effect the various variables (parameters) of the model have on the overall recognition scheme. This understanding gives us the capability of manipulating these variables so as to fit in the need of incorporating them in the overall algorithm without any significant loss in performance. As an example, the choice of the order of estimation is very critical in the accuracy and efficiency of the algorithm. This order can be estimated during the model building stage.

Ultimately, the various results of the analysis carried out in chapter 2 are reduced to categorical decisions in chapter 3. In this chapter, based upon the model already devised, we consider the implementation requirements. Subjects such as conversion and preemphasis and their effect on the overall recognition

results are covered in detail. Another subject, the 'feature' computation and extraction, is of great importance for the quality of the recognizer, and thus is also covered in detail. An observation referring to the signal's 'features' is that we have chosen, somehow arbitrarily, as 'features' a set of peaks for each frame to be processed. The peak-picking algorithm, which is the main component of the back-end of the recognizer, is supported by time domain components such as the duration of the signal, and also by autocorrelation domain components such as the relative power in a frame.

In chapter 4 we tabulate the results we took when we implemented the algorithm in a general purpose mini-computer. We also reason why the algorithm was successful and why the algorithm failed. In this chapter, we also state the computer requirements for the implementation of the algorithm.

CHAPTER 2

PARAMETRIC MODELING - 'FEATURE' COMPUTATION

2.1 Introduction

In this chapter, we present the parametric representation of discrete random signals, and the 'feature' extraction from these parameters. This presentation is based mostly on mathematical analysis. The work done in the past by other researchers was taken into account for the completion of this analysis. The results of our analysis will be used later in the model design.

For the parametric modeling, first, we examine the nature of the signals, and second, we elaborate on its three main stages, namely the best estimate, the order of the parameters, and the estimation of the parameters.

The 'feature' extraction subject is treated in a concise but complete way. Various possible 'features' are presented and their validity is examined. Taking into consideration that these 'features' will be processed digitally, we decide on the 'features' which could better fit in such an environment.

Many decisions pertaining to the design of our model will be taken in this chapter. They will be based upon the mathematical analysis presented in this chapter. The rest of the decisions, so that to complete the model design, will be taken in the next chapter. These decisions will be based upon the analysis carried-out in the present chapter and upon the needs arising when we implement our model in a digital environment.

2.2 Parametric Modeling of a Discrete Random Sequence

Suppose that there exists a one-input/output ideal system as depicted in Figure 2.1.

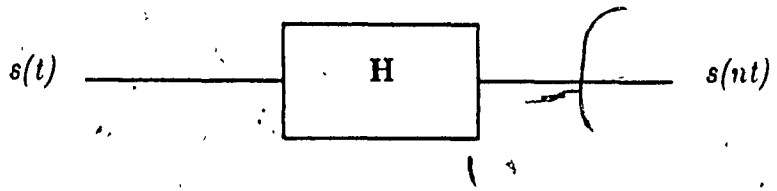


Figure 2.1

Ideal Impulse Sampler

The input, $s(t)$, to the system is a continuous random signal whose statistics we assume to be known over the interval $-\infty < t < +\infty$. The output is defined only at discrete points $\{ \ldots, -T, 0, T, 2T, \ldots \}$. Denoting the sequence of the output samples by $s(nT)$, these samples are equally spaced in time and may be written as

$$\ldots, s(-T), s(0), s(T), s(2T), \ldots \tag{2.1}$$

The sampling rate $(\frac{1}{T})$ is assumed to be at least twice the Nyquist frequency (f_N). In order to simplify our notation, we assume that T is one time unit. Thus, $s(n)$ and $s(nT)$ refer to the same thing, and they will be used interchangeably throughout this thesis. We observe, at this point, that the time series $s(n)$ $(-\infty < n < +\infty)$ consists of continuous samples defined over discrete points of the time axis. The so defined series $s(n)$ is called a discrete parameter process [39].

The conditioning of the continuous signal $s(t)$ must be extended further if we want to process the output of the system in a digital environment. For that we replace the ideal impulse sampler of Figure 2.1 with an ideal impulse sampler-quantizer. In this case, the quantized samples $s(nT)$ have discrete arguments and they are, also, themselves discrete. The so derived sequence

$$s(n) \quad (n = \ldots, -1, 0, 1, 2, \ldots)$$

(discrete, discrete-time random process) we will call for simplicity the "time-sample discrete signal" or simply the "signal".

From the statistics of the continuous random process $s(t)$ (which we have assumed to be known) we can determine the statistics of the discrete random process $s(n)$, by noting that the following relations hold :

$$E\{s(nT)\} = E\{s(t)\} \tag{2.2a}$$

and,

$$R[(n_1T, n_2T)] = R[(t_1, t_2)] \tag{2.2b}$$

where

$E\{*\}$ denotes the expected value, and

$R[*]$ denotes the autocorrelation of the random process [39].

An additional link between processes $s(t)$ and $s(n)$ is that if the process $s(t)$ is stationary (wide/strict sense) then the process $s(n)$ is also stationary (wide/strict sense). The inverse is not always true [39].

Given a random signal $s(n)$, our next aim is to represent this signal in a parametric form for further processing. The main reason for this is that the parametric representation of a signal results in efficient computer processing.

The processing speed and the memory required, are of main concern. Computer processing of the raw signal leads to relatively slow processing and requires, usually, more memory space than when we process a parametric model of the signal. In addition, although the raw signal $s(n)$ contains all the information attributed to the signal $s(t)$, its parametric representation, usually, does not. It has been found that the results taken are better (more accurate) when we process the latter than the former [50].

From the above outlined reasons, we conclude that the parametrization of a signal is a must. This invariably involves some kind of estimation of parameters. For the parametric representation of a signal, three main steps (stages) should be taken. These steps are:

(1) Decision of the best estimate

(2) Decision, taking into consideration decision (1), of the order of the parameters, p.

(3) Based on decisions (1) and (2), the estimation of the parameters.

The above three stages will be examined next in some detail. The examination will not be done in the order given above. In our analysis we will interchange the order of the second and the third stages. In implementing a parametric representation of a signal, the second stage should be cleared before the third stage.

The interchange of the order of the last two stages in our analysis will not affect our results, because when clearing stage three before stage two we assume that the order of estimation in itself is a variable.

B.E.1 — Best Estimate

In this stage we will examine first the nature (statistics) of the continuous signal $s(t)$. Second, from the statistics of $s(t)$ we will derive the statistics of $s(n)$. Finally, the decision for the best estimate will be taken. This decision will be based, first, on the nature of $s(n)$, and second, on the requirements for further processing.

The variation of $s(t)$ is, in our case, a function of a mechanical displacement (microphone). This mechanical displacement is, in turns, a function of the change in the pressure of the volume of air surrounding the microphone. The sound generator is responsible for the changes of the pressure level. Pictorially, we can approximate the generation of $s(t)$ as shown in Figure 2.2.

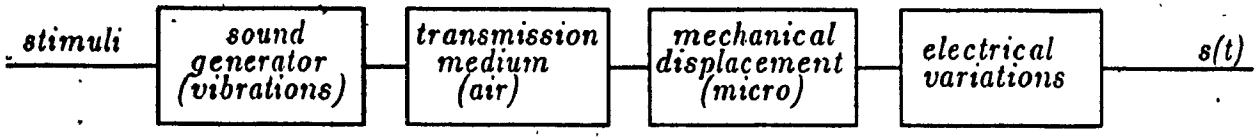


Figure 2.2

Generation of the Continuous Random Sound Signal $s(t)$

Each box in Figure 2.2 may be expanded or altered. As an example, air is not the only transmission medium. Any kind of concentration of matter may act as a sound medium. In any case, for simplicity in our analysis we consider Figure 2.2 as a valid and sufficient (for this work) representation of the generation of the signal $s(t)$.

Every step involved in the generation of $s(t)$ adds, invariably, some kind of noise. The variable "noise" should be taken into serious consideration in examining the nature of $s(t)$. For clarity and simplicity of our analysis we omit, at the present stage, this variable. We assume that the conditions are ideal in any step (i.e. noise does not exist). We will examine this subject later on (chapter 3) in detail.

Further, we assume that every system (represented by a box) in Figure 2.2 is linear, time-invariant, and causal. Under the assumptions mentioned above, we can conclude that $s(t)$ is proportional to the sound level. An approximation can be made at this point concerning the statistics of $s(t)$. This approximation is based mostly on observation. We approximate the simultaneous probability distribution of any random vector of $s(t)$

$$[\, s(t_1),\, s(t_2),\ldots,\, s(t_n)\,]$$

as being a Gaussian distribution. This means that any individual or joint probability density is also Gaussian. With the last approximation and taking into account equations 2.2a and 2.2b we can state that : the random variables of the process $s(n)$ are jointly and individually normally distributed. Assuming that the process does not have any deterministic component (bias), then all the random variables of the process and the process itself have zero mean [55]. An additional assumption is the stationarity of $s(t)$, which implies the stationarity of $s(n)$. This assumption will be discussed in detail in Section 2.2.2. As it turns out, for the case of normal stationary processes, the best linear prediction and the best prediction coincide [5]. Also, for normal processes with zero mean, the nonlinear and linear mean square estimates are identical [38]. Combining the above, we conclude that the mean square (MS) estimate for our signal can be as good as any other estimate.

From the last conclusion, we see that, from the analysis point of view, there are many classes of best estimates, which can be made to give identical (best) results. So, having this freedom of choice, we choose the best estimate which could be easily and efficiently implemented in a digital computer environment. Undoubtedly, the best estimate which fits better to the environment to be implemented is the estimate which leads to an autoregressive model [17].

To elaborate to some extent, let $s(n)$ be the random variable (RV) to be estimated by the RV $s'(n)$, and let $e(n)$ be the random error of the estimation. In the chosen best linear estimate we define $s'(n)$ and $e(n)$ in the following way:

$$s'(n) = \sum_{k=1}^{p} a_k \; s(n-k) \tag{2.3}$$

$$e(n) = s(n) - s'(n) \tag{2.4}$$

where, p is the order of estimation (to be determined), and a_k's are the estimation coefficients, also to be determined.

For the last two equations we make the following observations. The model defined by these two equations is not the only available linear model of an estimated signal $s'(n)$. As an example, $s'(n)$ or e (n) could be defined as:

$$s'(n) = \sum_{k=-p}^{p} a_k \; s(n-k)$$

$$e(n) = s(n-k) - s'(n)$$

The above equations so defined lead to a non-autoregressive model [22]. Equations 2.3 and 2.4 on the other hand give a model which is a special case of an autoregressive model with moving-average errors (ARMA) [17]. This special case is prefered over the general autoregressive model because the latter cannot be

easily and efficiently processed [17,30].

Summarizing this stage: based on some general assumptions and criteria, we have chosen the MS-linear estimation as the best estimate with the understanding that $s'(n)$ and $e(n)$ are defined as in equations 2.3 and 2.4.

2.2.2 *Estimation of Parameters.*

Based on the previous analysis, the aim of the present Section can be stated as follows [38] :

> "Estimate a random variable $s(n)$ by a random variable $s'(n)$, using linear mean square prediction (estimation), so that the error
>
> $$e(n) = s(n) - s'(n)$$
>
> to be minimum."

The random variable $s'(n)$ was defined in equation 2.3 as

$$s'(n) = \sum_{k=1}^{p} a_k \, s(n-k)$$

where p is the order of estimation, or, equivalently, it represents the number of estimation coefficients. So our aim is reduced to that of finding p random variables a_k $(k = 1,2,...,p)$ from the signal $s(n)$.

Using equation 2.3, the error, $e(n)$, is given by :

$$e(n) = s(n) - \sum_{k=1}^{p} a_k \, s(n-k) \qquad (2.5)$$

In the mean-square estimation we minimize the expected value of the square

of the error, E_t , which is given by

$$E_t = E\{e^2(n)\} = E\{[s(n) - \sum_{k=1}^{p} a_k\ s(n-k)]^2\} \tag{2.6}$$

Minimization of E_t with respect to all the coefficients a_l gives

$$\frac{\partial E_t}{\partial a_l} = 0\ ; \quad l = 1, 2, \ldots, p$$

or, equivalently

$$\frac{\partial}{\partial a_l}\left(E\{[s(n) - \sum_{k=1}^{p} a_k\ s(n-k)]^2\}\right) = 0\ ; \quad 1 \leq l \leq p \tag{2.7}$$

Carrying the derivation inside the expectation, which is permissible since the expectation applies to the random variables $s(n)$ $(0 \leq n \leq \infty)$, we take

$$E\{2[s(n) - \sum_{k=1}^{p} a_k\ s(n-k)]\,[-s(n-l)]\} = 0\ ; \quad 1 \leq l \leq p \tag{2.8}$$

or

$$E\{-s(n)\,s(n-l) + s(n-l)\sum_{k=1}^{p} a_k\ s(n-k)\} = 0\ ; \quad 1 \leq l \leq p \tag{2.9}$$

Using the linearity property of the expectation, equation 2.9 can be written as

$$E\{\sum_{k=1}^{p} a_k\ s(n-k)\,s(n-l)\} = E\{s(n)\,s(n-l)\}\ ; \quad 1 \leq l \leq p \tag{2.10}$$

or, again using linearity, equation 2.10 can also be written as

$$\sum_{k=1}^{p} a_k\ E\{s(n-k)\,s(n-l)\} = E\{s(n)\,s(n-l)\}\ ; \quad 1 \leq l \leq p \tag{2.11}$$

The last equation, clearly, defines a linear system of p equations (for $l = 1,2,...,p$) in p unknowns $(a_1, a_2, \ldots, a_p)$. This system can be easily solved if the expectations are known. The problem is that, at the moment, we do not have enough information about the second moments $E\{s(n-k)\,s(n-l)\}$ and $E\{s(n)\,s(n-l)\}$. In order to be able to calculate them we proceed therefore with the detailed examination of some signal characteristics.

The sound signals we are dealing with, are in general, time-varying (non-stationary) signals. However, a fundamental assumption for the present analysis is that, over a sufficiently long time interval, they can considered stationary. The stationarity of $s(t)$ implies the stationarity of $s(n)$ (Section 2.2). This assumption is based on the observation that any sound generator cannot change state in zero time interval (inertia of the system). This time interval, in which our fundamental assumption is approximately valid, is of the order of ms. The sampling rate is, usually, of the order of $(\mu s)^{-1}$, which shows that during the (assumed) stationarity a large number, on the average, of samples of the signal can be taken.

Our next aim is to approximate an optimal time interval for the stationarity of the signal, so as the model derived to be reliable. The optimal time interval, in which stationarity is assumed, is of stochastic nature, and depends on the mechanics of the sound generator and on the stimuli (Figure 2.2). In developing an algorithm to be implemented in a computer environment, it is highly desirable to approximate this optimal time interval with a deterministic quantity. For the determination of the optimal time interval, in addition to the signal stationarity, we take into account the statistical uncertainty of the results (SUR). We can define, loosely, as SUR the uncertainty which is associated with the failure to estimate the parameters of a model with precision. The SUR, clearly, increases when the sample size decreases, and it is eliminated (i.e. certainty) in the ideal

case of an infinite sample size. So, we can reduce the SUR at the expense of a longer time interval (i.e. larger sample size). If we increase the sample size beyond a limit, then, we run the risk of assuming stationarity, when stationarity does not apply any more. Thus, stationarity and SUR are two concepts contradicting one another. The optimal size of the sample records, the one that balances these two concepts, can be approximated experimentally. This experimental approximation leads directly to a deterministic sample size. For the approximation of the sample size we consider the nature of the signal and the sampling rate. It has been observed that the signals we are dealing with can be considered stationary for a time interval of up to 40 ms. The sampling frequency may range from 5 KHz up to 20 KHz. This gives a maximum sample size of between 200 and 800 samples respectively. Experimentally we found that the optimum and the maximum sample sizes may be approximated by the same quantity. For reliable results and computer efficiency we should use a sample size that is marginally smaller than the maximum permissible, due to the stationarity assumption, sample size. The final decision for the sample size (N) will take place in the next chapter. Presently, after the above analysis, we consider $200 \leq N \leq 800$ samples (but a fixed number).

From the previous analysis we can conclude that we can process the signal piecewise. We may process a frame of the signal each time. Because of the assumed stationarity of the signal, each frame (of length N) should contain all the information on the state of the sound generator for the corresponding time interval. The piecewise processing of the signal necessitates the truncation of the time series $s(n)$. This truncation of the signal may be viewed as a window through which we observe a finite (N) number of samples. It may, also, be viewed as taking pictures of the sounds generated at constant time intervals, and processing them separately. This truncation (framing) of the signal is

mathematically equivalent to multiplying the signal by a time window function of the form

$$w(nT) = u[(n-mN-1)T] - u[(n-(m+1)N-1)T] \tag{2.12}$$

where $u[*]$ is the unit step function, m is the frame number ($m = 0,1,2,\ldots$), N is the number of frame samples, and $0 < n \leq N$. In the next chapter, we will see that the window function, as defined above, is not the best choice. At this point, however, for simplicity, we assume the above definition for $w(nT)$.

Now, having presented all the assumptions, we return to the original objective of the analysis which was to derive the second moments in equation 2.11.

Assuming our signal to be ergodic (which implies stationarity), the sample serial correlation coefficients, or simply the autocorrelation, is defined as follows :

$$\begin{aligned} R(n_1,n_2) &= E\{s(n_1)\, s(n_2)\} \\ &= \lim_{M\to\infty} \frac{1}{M} \sum_{m=0}^{M} s(m+n_1)\, s(m+n_2) \end{aligned} \tag{2.13}$$

By framing the signal, as discussed previously (equation 2.11), we can write equation 2.13 as

$$R(n_1,n_2) = \left(\frac{1}{N}\right) \sum_{n=1}^{N} s(n+n_1)\, s(n+n_2) \tag{2.14}$$

Using equations 2.13 and 2.14 we can write equation 2.11 as follows :

$$\sum_{k=1}^{p} a_k\, R(n-k,n-l) = R(n,n-l)\,; \quad 1 \leq l \leq p \tag{2.15}$$

For stationary processes though, it is known [38] that

$$R(n_1,n_2) = R(n_1-n_2) \tag{2.16}$$

and using last relation equation 2.15 becomes

$$\sum_{k=1}^{p} a_k\, R(l-k) = R(l)\,; \qquad 1 \leq l \leq p \tag{2.17}$$

The above constitutes a linear system of p equations in p unknowns. The $R(l)$'s can be computed by the relation (combining equation 2.14 with 2.16)

$$R(l) = (\frac{1}{N}) \sum_{n=1}^{N} s(n)\, s(n+l) \tag{2.18}$$

For clarity we make several observations. Equation 2.13 is true for ergodic processes only. It states that the ensemble averages may be equated by the time averages of the process. For the definition of the above terms, and details about ergodicity we refer to [38].

For equations 2.17 and 2.18 we make the following observations. First, the signal is given by

$$s(n) = \begin{cases} s(n) & ; \quad \text{for} \quad 0 < n \leq N' \\ 0 & ; \quad \text{elsewhere} \end{cases} \tag{2.19}$$

and second, the autocorrelation is an even function (by definition), i.e.

$$R(l) = R(-l).$$

Expanding equation 2.17 in k and l we take :

$$\begin{array}{lllll} R(0)a_1+ & R(1)a_2+ & \cdots & +R(p-1)a_p & = R(1) \\ R(1)a_1+ & R(0)a_2+ & \cdots & +R(p-2)a_p & = R(2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R(p-1)a_1+ & R(p-2)a_2+ & \cdots & +R(0)a_p & = R(p) \end{array} \tag{2.20}$$

In literature, the equations 2.20 are called the Yule-Walker equations, the a_k's are called the linear prediction (LP) coefficients, and the pivotal matrix is called a Toeplitz matrix. In the next chapter, we will outline an efficient algorithm for solving the linear system of equations 2.20.

Summarizing the second step for the parametric modeling of the signal, we see that : using linear MS estimation and assuming ergodicity for the process, to compute the parameters of estimations we have to solve a linear system of equations. The final remark is that equations 2.20 have dimensions not of the time domain, but of the correlation domain.

2.2.3 Order of Parameters

The last main stage to be examined for the parametric modeling of a signal is the determination of the order of the estimation parameters a_k $(k = 1,2,...,p)$, i.e. the determination of p . In literature the number of parameters proposed for processing equivalent signals varies from two [29] to forty nine (49) [44]. In the present analysis we assume that $p \ll N$, where N is the sample size. If $p \to N$ (or it is larger), then the parametric representation of a signal loses all its usefulness, because in such a case it is more efficient to process the raw signal without loss of signal characteristics, than to use a model where, in addition to efficiency, some characteristics of the signal are lost.

In the present, we examine some theoretical aspects dealing with the order of parameters. The final determination will be done in the next chapter.

Form equation 2.8 we see that the error is orthogonal to any of the p past samples i.e.

$$[s(n) - \sum_{k=1}^{p} a_k \; s(n-k)] \quad \perp \quad s(n-l) \; ; \quad 1 \leq l \leq p$$

and hence it is orthogonal to any linear combination of these samples. Let that linear combination be the estimated signal, thus,

$$[s(n) - \sum_{k=1}^{p} a_k \; s(n-k)] \quad \perp \quad \sum_{k=1}^{p} a_k \; s(n-k)$$

or the same

$$e(n) = [s(n) - s'(n)] \quad \perp \quad s'(n)$$

The expected MS error

$$E_l = E\{[s(n) - s'(n)]^2\}$$

$$= E\{[s(n) - s'(n)]\,[s(n)]\} + E\{[s(n) - s'(n)]\,[-s'(n)]\}$$

is minimum when

$$E\{[s(n) - s'(n)]\,[s'(n)]\} = 0$$

and it is given by

$$E_m = E\{[s(n) - s'(n)]\; s(n)\}$$

$$= E\{[s(n) - \sum_{k=1}^{p} a_k \; s(n-k)]\; s(n)\}$$

$$= E\{s^2(n)\} - E\{\sum_{k=1}^{p} a_k \; s(n)\, s(n-k)\}$$

$$= R(0) - \sum_{k=1}^{p} a_k \, R(k) \qquad (2.21)$$

In theory, E_m is a monotonically decreasing function of p [56] (i.e. $E_m \rightarrow$ constant). Experimentally, it has been observed that, for any signal, the E_m levels off after a certain value of p (Figure 2.3). In such a case, a further increase of the number of linear estimates has little effect on E_m. So, as the first constraint for the determination of the order of parameters we should choose p where the slope of the curve E_m vs p goes to zero. For the second constraint, we make an examination on where and for what the estimation coefficients could be used. For that, a somehow detailed analysis will take place.

Equation 2.5 can be written in the z-domain (assuming unity gain for the filter) as

$$E(z) = S(z)\,[1 - \sum_{k=1}^{p} a_k\, z^{-k}]$$

or

$$E(z) = A(z)\,S(z) \tag{2.22}$$

where we have defined

$$A(z) \equiv 1 - \sum_{k=1}^{p} a_k\, z^{-k} \tag{2.23}$$

From equation 2.22 we see that the MS error can be viewed as the output of a system whose transfer function is the prediction polynomial $A(z)$ and it is driven by $S(z)$. We write equation 2.22 as

$$S(z) = \frac{E(z)}{A(z)}$$

The above relation, depicted in Figure 2.4, shows that the signal $s(n)$ can be

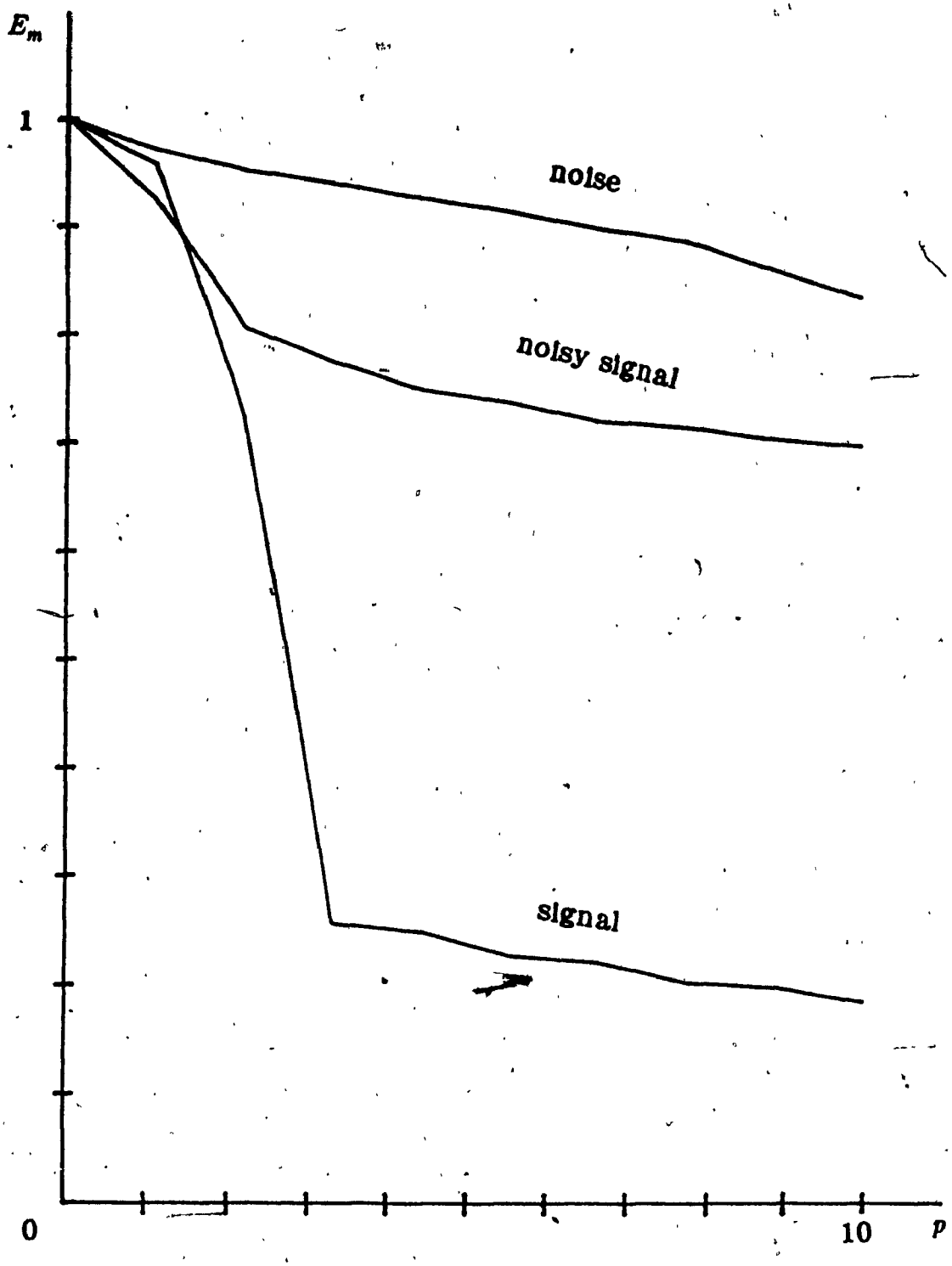


Fig. 2.3

E_m vs the Order of Estimation (p)

viewed as the output of a system whose transfer function is $\frac{1}{A(z)}$ and it is driven by $e(n)$, where $e(n)$ is "white" noise. So, over a short period of time, we can see that the impulse response of the filter $\frac{1}{A(z)}$ approximates the spectrum of the signal.

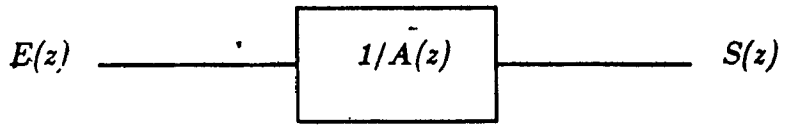


Figure 2.4

Frequency-domain Synthesis Model.

The power spectra, usually, have some peaks and the corresponding valleys between the peaks. The power concentration on the frequencies of the peaks is higher than the power distributed on the ones adjacent to the peaks. The local maxima of the spectral envelope of the signal $s(n)$ correspond to the poles of the function $\frac{1}{A(z)}$. For real estimation coefficients a_k, and assuming p an even number, we expect the number of peaks to be approximately one half minus one the order p of the polynomial $A(z)$. These peaks define the fundamental (F_0) and the secondary $(F_1, F_2, \ldots, F_{\frac{p}{2}-1})$ frequencies contained in the signal. So, assuming that we will process the signal in the frequency domain, the desired number of frequencies to be examined will set an additional constraint for the absolute minimum order of estimation. This absolute minimum, numerically is given by the double of the number of the desired frequencies plus two, i.e. $P_{\min} = 2(F_{desired}) + 2$.

The two already mentioned constraints for the determination of p refer to

the lower bounds of p. For the upper bound we have to take into consideration the desired confidence of the model. Thus, the desired precision of the model defines the third constraint. The examination of the third constraint will take place in the next chapter because in determining it we have to take into account some implementation needs.

Summarizing this stage, concerning the order of the estimate, we found that there are three main constraints for the determination of p. Namely, p must be chosen in such a way so that E_m to be close to the steady state. Also the required number of frequencies to be processed, will determine a requirement for the minimum value of p. The system's confidence and efficiency will determine the requirement for the maximum value for p. The combination of these three constraints will take place in chapter 3, where we will also give a deterministic value for p, which value will be used in our model.

2.3 Power Spectrum Derived from the Linear Prediction Coefficients.

We assume that the model for the parametric representation of the signal which we formulated in Section 2.2 is valid in every respect. The question now is : can we use the temporal variation of the linear prediction coefficients a_k's for matching purposes ? Unfortunately the answer is negative. Attempts to use the LP coefficients as 'features' of a sound did not give encouraging results. By 'features' we mean the models of a signal intermediate between the signal processor and the pattern matcher. As it is stated in [25],

> "... The luck of success may be partly due to the fact that the 'feature' space spanned by the LP coefficients is too complicated to introduce a simple and effective measure of distance between the elements in the space."

Other attempts were made to use the LP coefficients to derive the minimum prediction error (residual), E_m, using equation 2.21. The temporal variation of E_m can give a measure of distance between signal frames and templates. Again, the results were not as expected, and they were not as good as the results taken by using other schemes such as zero crossing measurements, FFT, etc [11,35].

From the above we see that 'feature' extraction is one of the most crucial steps in pattern matching algorithms. An algorithm may give better results than another algorithm, and the main (and most of the time, the only) reason is the superior 'feature' extraction part.

From experiments it was found that the temporal variation of formants contains all the information necessary to recognize the signal unambiguously [54]. Further, the temporal variation of the first three formants contains most of the

information to label the signal [54]. The formants correspond to the poles of the filter $\frac{1}{A(z)}$. They also correspond to the peaks of the power spectrum of the signal. The peak picking from the power spectrum is more computationally efficient than solving numerically $A(z)$ for its zeroes [25].

We have seen previously that the signal may be approximated by the output of a filter, which filter is driven by a "white" noise. The transfer function of the filter is

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k \, z^{-k}} \tag{2.24}$$

The power spectrum (spectral density) $P(\omega)$ of the signal is given by

$$P(\omega) = |S(z)|^2_{/z=e^{-j\omega T}}$$

where $S(z)$ is the two-sided z-transform of the $s(nT)$ and is defined as

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

It has been shown [30] that

$$P(\omega) = |H(z)|^2_{/z=e^{j\omega T}} \qquad \text{as} \quad p \to \infty \tag{2.25}$$

within a constant G^2 (G is the gain of the filter, which we have assumed to be unity). The last equation shows that we can approximate any signal spectrum by an all pole model.

For p relatively a small integer, we can assume that the approximated power spectrum of the signal will show the important peaks. This assumption fails when two or more peaks of $|H(e^{j\omega T})|^2$ are very close to each other.

So, in determining p we have to consider the above constraint also.

Let $P'(\omega)$ be the approximated spectrum of the signal $s(nT)$, i.e.

$$P'(\omega) = |H(z)|^2_{/z=e^{j\omega T}} = \frac{1}{\left|1 - \sum_{k=1}^{p} a_k z^{-j.\omega k T}\right|^2} \qquad (2.26)$$

We make the following two observations for the last equation. First, the function $P'(\omega)$ is periodic, with period the sampling frequency $\omega_s = \frac{2\pi}{T}$. Second, it is an even function, i.e.

$$P'(\omega) = P'(-\omega)$$

We can use these two observations in the implementation stage to reduce the number of calculations. From the first observation we derive that we need calculate $P'(\omega)$ only in the baseband, which extents from $\frac{-\omega_s}{2}$ to $\frac{+\omega_s}{2}$. The second observation shows that half the baseband ($\frac{-\omega_s}{2}$ to 0) is the mirror image of the other half of the baseband (0 to $\frac{+\omega_s}{2}$) on the $\omega=0$ axis. The same observation leads us to the conclusion that we can express equation 2.26 as a function of *cos* only. This is done in Appendix A.2, where we demonstrate that

$$P'(\omega) = \frac{1}{\rho(0) + 2\sum_{k=1}^{p} \rho(k)\cos(k\omega)} \qquad (2.27)$$

where

$$\rho(k) = \sum_{l=0}^{p-k} a_l\, a_{l+k} \quad \text{and} \quad a_0 = 1.$$

So, we can use the prediction polynomial $A(z)$ to take a smooth spectral behavior, from which we can extract the peaks to be used as the main 'features' of our model.

2.4 Conclusions

In this chapter we attempted a two-folded task. First, we built a model to parametrize the signal. Based upon several assumptions, the parametric model chosen is an autoregressive model, whose parameters are estimated by using linear mean-squared estimation. Second, having the model parameters, we examined several possibilities of efficiently using them, so that the 'features' derived should lead to a reliable and easy to implement pattern matching. We decided to use as 'features' of the signal its most significant formants. These formants will be calculated from the spectrum of the signal. So, the model parameters will be used to derive the spectrum of the signal.

Three subjects left incomplete in this chapter, namely the "noise", the order of the model (p), and the sample size (N), will be further discussed in the next chapter.

CHAPTER 3

SIGNAL PROCESSING

3.1 Introduction

Two main topics are covered in detail in this chapter, the noise variable, and the signal processing.

In the previous chapter, when we formulated an ideal model, we did not take into consideration the variable "noise". In the implementation stage, which is covered in this chapter, the noise superimposed on the signal is a critical factor for poor or good recognition results. For that, it is covered in detail. We have classified the noise into two classes, the transmission and the quantization noise. The transmission noise is generated in the analog side, and the quantization noise is generated in the digital side of the processing. Several assumptions, based upon the theory of the random noise, have been made, so that we pinpoint to the components of the noise which may substantially influence our results.

The other topic covered in the chapter is the digital computer aided signal processing. Three main stages of the signal processing, namely the signal preprocessing, the signal main processing, and the signal post processing are covered here in detail. The classification of the signal processing stages is done accordingly to the domain the variables we work with belong to. This is depicted in Figure 3.1. The signal preprocessing stage consists of the conversion, the preemphasis, the windowing of the signal, and the autocorrelation coefficients computation steps. The signal main processing stage consists of the end detection, the normalization, the linear prediction autocorrelation coefficients computation, and

the 'feature' extraction steps. The signal post processing stage consists of the fixed templates matching, the adaptive templates matching, and the final decision for the nature of the signal steps.

For each step mentioned above, we first explain why it is necessary that it be implemented, second we examine how it can be implemented in a digital environment, and third we estimate the computer time and memory required for the implementation.

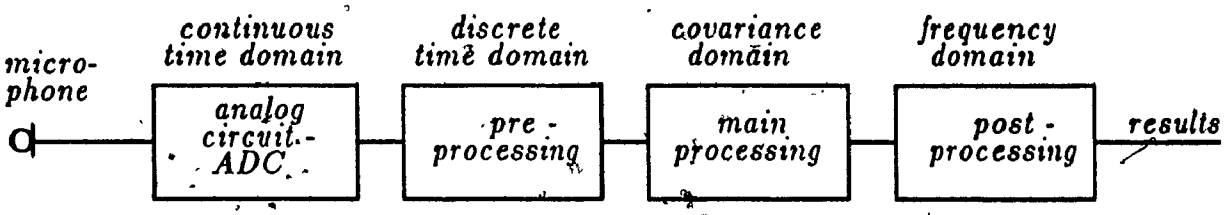


Figure 3.1

Stages for the Signal Processing

3.2 Noise

The subject of this section is to examine the effect of the noise appearing in the input of the digital environment on the processing. We will examine, briefly, how the noise is generated, how it propagates, and what measures we take to reduce its effects. Noise appearing in later stages of the signal processing will be examined within the context of the corresponding stages.

For the present examination, we can classify the noise appearing to the input of the computer as transmission and quantization noise. With this distinction we can write

$$s(n) = f(n) + v_t(n) + v_q(n) \tag{3.1}$$

where:

$s(n)$ is the data received,

$f(n)$ is the original signal, and

$v_t(n), v_q(n)$ are the transmission and the quantization noise respectively.

The examination of the transmission and of the quantization noise will take place separately. We will start with the transmission noise first.

The transmission noise is due to the transmission of the signal from the sound generator to the input of the Analog-to-Digital Converter (ADC). Assuming that the quantization noise is an additive noise, we can examine the transmission noise from the analog side of the ADC. In general, the transmission noise may be due to the environment (to the input of the microphone) or due to the inherent imperfections of the electrical circuits. So, the transmission noise ($v_t(t)$) can be expressed as

$$v_t(t) = v_n(t) + v_l(t) + v_u(t) \qquad (3.2)$$

where:

$v_n(t)$, $v_l(t)$ are the environmental and electrical noise respectively, and

$v_u(t)$ is an 'unknown' (at present) component of the transmission noise.

These three components of the transmission noise will be examined with some details in the order given by the equation 3.2.

The environmental (background) noise, $v_n(t)$, may be defined as the corruption of the original signal (message), $f(t)$, due to the environmental conditions, up to the input of the microphone. The sound waves received by the microphone may come directly from the sound generator or through deflection of the sound waves on walls and on various objects. These objects may be fixed in place (tables, partitions, etc.) or moving (chairs, humans, etc.) With various delays, the microphone will receive attenuated and distorted versions of the $f(t)$ in addition to the $f(t)$ itself. Also, in a non-controlled environment, as is our case, many other sounds of *a priori* unknown origin (possibly breathing, voices, air-conditioning, movements of objects, etc.) will reach the microphone. These kinds of noise, of environmental origin, will be an additive component to the original message, $f(t)$. From the above we see that we cannot make any valid assumption so that to be able to model the environmental noise. This observation leads to the conclusion that the information carried by the message can only be treated in a statistical manner.

Although the background noise cannot be modeled or estimated, it can be reduced down to a certain level. In a partly controlled environment (example is a room for general purpose and use), we found, experimentally, that the background noise is mostly concentrated in the 0 to 200 Hz frequency range, most of

the time. So, the low frequences can be eliminated by band-passing the signal through an analog bandpass filter whose lower cutoff frequency is around 200 Hz, and the upper cut-off frequency is approximately the Nyquist frequency (~ 3.2 KHz), so as to ensure, in addition, a common bandwidth for the recording conditions (sampling). The removal of the lower frequencies, naturally, reduces the range of our recognition system.

For the background noise whose frequences cannot be eliminated there are not many things that can be done. As long as the signal-to-noise ratio is high then the amount of information carried, and possibly extracted, from the signal is large. This amount of information is approaching zero very rapidly as the noise increases in intensity [56]. So, summarizing the above concerning the environmental noise, we see that the performance of any recognition system operating in a non-controlled environment, and hence ours, is highly dependent on the background noise. The performance may be increased in instances when the intensity of the background noise is small compared to the intensity of the signal i.e. high signal-to-noise ratio. The performance drops rapidly to zero for low signal-to-(environmental) noise ratio. Also, removal of the low frequency components of the environmental noise will not affect greatly the performance of the system because we expect the main frequency components of the signals to be recognized to be above the main frequency components of the background noise (i.e. above 200 Hz).

The second component of the transmission noise is the electrical noise $v_1(t)$. This noise is generated in all stages of propagation of the electrical signal from the microphone up to the input of the ADC. Concerning the microphone, we observed, from its response characteristics, that its response is non-linear for the various frequences. It responds, theoretically, to the sound waves from ~ 50 Hz to ~ 50 KHz. In useful, practical terms its response is between ~ 100 Hz and ~ 13

KHz. The important fact is that, even in this range (0.1 to 13 KHz) its response is non-linear but somehow it has a slightly zig-zag shape. Although we use for our experiments a high-quality, general purpose, omni-directional, moving-coil microphone, its frequency response varies almost stochastically within two levels of amplitude (Figure 3.2). This shows that, during the translation of the sound waves into electrical signals a noise, inherent to all microphones, is added to the signal [19]. For practical purposes, we can assume that this is a "white" noise, i.e. it has a constant spectrum ($S(\omega) = \text{constant}$).

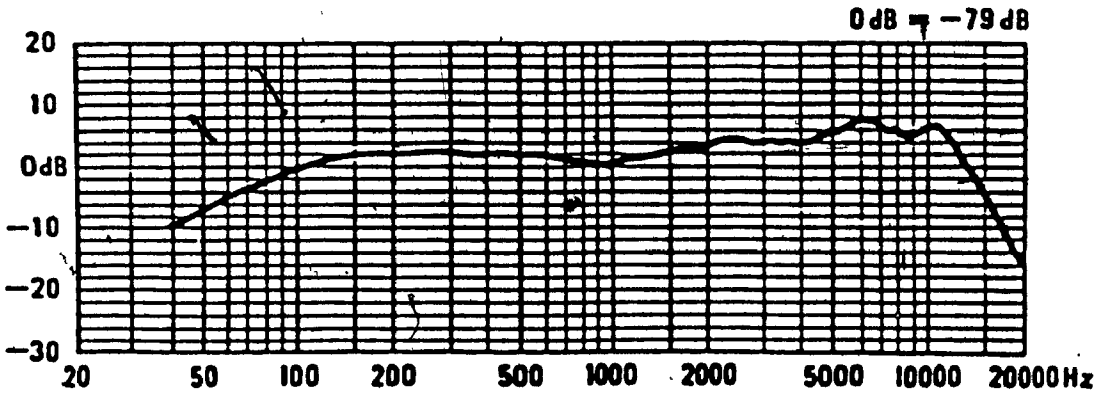


Figure 3.2

Microphone - Frequency Response

Considering all the electrical circuits and devices used (microphone included), we know that they suffer, invariably, from thermal (caused by the random thermal agitation of electrons in resistive elements) and from random noise (caused by any randomly occurring transient disturbances). The thermal noise is a form of "white" noise and can be treated as such. The random noise may result in "white" noise if there is a high rate of occurrences, or in impulse noise if the rate of occurrences is low. We can assume that the impulse noise is negligible

in comparison to the "white" noise of the electrical circuits involved, and so, it can be omitted from further consideration. So, summarizing for the second component of the transmission noise, we see that the main component of the electrical noise is a "white" noise, which has an approximately constant amplitude spectrum. Although, under several assumptions, we can reduce the "white" noise in the analog side of the ADC, this reduction will take place at the digital side of the ADC. The main reason for this is because our *a priori* knowledge about the incoming signal is limited. Another reason is that we can reduce the "white" noise very efficiently by using digital techniques, which may be adaptive.

The third, and last, component of the transmission noise was characterized previously as the 'unknown' noise $v_u(t)$ (equation 3.2). This 'unknown' noise is of environmental origin but of electromagnetic radiation nature. It is produced by electrical devices and wiring in the proximity of the analog circuits used for the experiments. As an example, we found, during the first stages of the experiments, that the terminals used emitted a low frequency but high intensity electromagnetic radiation. To reduce the effect of the 'unknown' noise, we had to shield the analog circuits. This led to virtual elimination of the 'unknown' noise.

So, summarizing about the transmission noise, we see that it has three components : the environmental, the electrical, and the 'unknown' noise. We can only eliminate effectively the 'unknown' noise by shielding the circuits used. The reduction of the electrical noise will be done using digital techniques (see preemphasis). The background noise is highly unpredictable, and its frequency components above $\frac{\omega_s}{2}$ can be eliminated selectively (low-pass or bandpass filtering of the signal). For the rest of the frequency components of the background noise, very little can be done.

The second main component of the noise appearing in the input of the digital environment is the quantization noise $v_q(n)$ which is the result of the quantization process, i.e. the conversion of the continuous input signal into a finite number of discrete values. In the quantization process, usually, the continuous signal is approximated to the nearest allowed digital level. The main concern with the quantization noise is that after quantization there is not any way we can reconstruct exactly the continuous signal. So, care should be taken to have a high signal-to-quantization-noise ratio, r_q . Assuming that all signal levels are equi-probable, and that the quantization error can be described by a uniform probability density function, then it has been shown [52] that

$$r_q = n^2$$

where n is the number of levels used, (i.e. $n = 2^m$), and m is the number of bits used for the quantization. This shows that we can increase r_q by increasing the word length of the ADC. It can be virtually eliminated by using more than 10 bits for the conversion. For the experiments we use a 12-bit ADC, where the most significant bit is reserved for the sign. That means that in our case $r_q = 66$ dB, and so the effects of the quantization noise may be considered negligible.

So, from the previous analysis for the various kinds of noise appearing in the output of the ADC, we see that the data received, $s(n)$, can be approximated by the relation,

$$s(n) \approx f(n) + v_n(n) + v_l(n) \qquad (3.3)$$

where

$f(n)$ is the original signal,

$v_n(n)$. is the environmental noise, and

$v_l(n)$ is the electrical noise.

For the above approximation we have assumed that all other components of noise are negligible, i.e. care has been taken to reduce them to an acceptable level so that they will not have any significant effect in the further processing of the signal $s(n)$. The electrical noise, $v_l(n)$, is a "white" noise. The nature of the environmental noise, $v_n(n)$, is unknown.

In section 3.3.2, while we will be examining preemphasis, we will outline how we can further reduce the effect of $v_n(n)$ and $v_l(n)$, in the estimation of $f(n)$ from $s(n)$.

3.3 Signal Preprocessing

In this section we will examine the steps needed to process the signal in the time domain. Thus, this section covers the signal processing from the output of the ADC up to, and including, the computation of the autocorrelation coefficients. Input in this stage of processing is the sampled data and the output is a vector of coefficients to the main processing stage.

The current stage is broken down into the following four distinct steps: conversion, preemphasis, windowing, and autocorrelation coefficients computation. For each step we reason its necessity, and we explain why some choices were made. The steps taken for the signal preprocessing are depicted in Figure 3.3. The first three steps may be characterized as "signal conditioning" steps. The last step may be characterized as an "intermediate 'feature' computation" step.

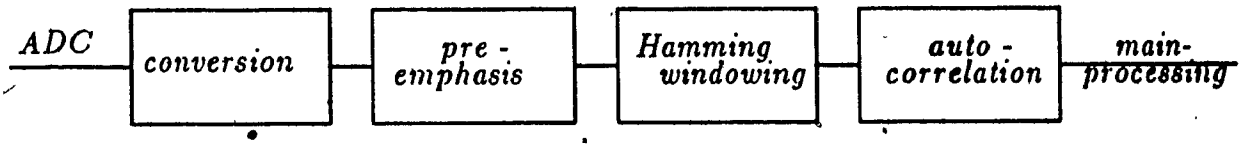


Figure 3,3

Signal Pre-processing Steps

3.3.1 Conversion

In this step the output of the ADC is conditioned in such a way so as to be compatible with the computer's internal representation of numbers. There are

two subjects to be discussed in here. First, we will reason why the conversion step was necessary, and second, we will describe the conversion used in the implementation of our algorithm.

The output of the ADC used is given by a series of 12 bits, in one's complement notation. The computer used for experimentation represents the signed binary integers in two or four bytes as depicted in Figure 3.4 in two's complement representation.

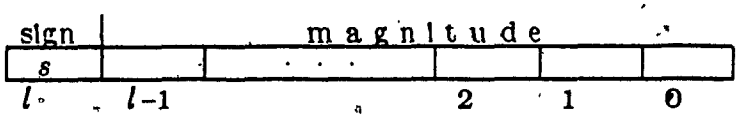


Figure 3.4

Data Representation in Two's Complement

The length $(l+1)$ is given by

$$(l+1) = 2^m \quad ; \quad m = 4,5$$

An easy conversion for a negative ADC output, at this point, so that the ADC output to be compatible with the computer's representation of integers (two's complement), could be to logically OR the ADC output with a binary constant of the form 1...10...0, where the length of the constant is equal to $(l+1)$ and the number of 1's is equal to $(l-k)$, and to add 1 to the result. Assuming that the preprocessing stage could be done by multiplication and addition/subtraction of integers (in preemphasis $\alpha = 1$, see next topic), this conversion seems the best choice.

By using integer arithmetic the preprocessing stage could be performed faster than by using floating point (FP) numbers arithmetic. The problem is that 'overflow' may occur. Assuming that we reserve 8 bits for the resolution of the

window used, and that the window length is 256 samples, then to cover the complete range of the autocorrelation coefficients we would need 45 bits (sign+(11+8)*2+8), which exceeds by far the available length of 32 bits. This limitation was the main reason to convert the output of the ADC into FP numbers, with format as depicted in Figure 3.5.

Figure 3.5

Floating Point Data - Single Precision

In Figure 3.5, s denotes the sign of the mantissa. The exponent is in excess of 200 (octal) notation, and the mantissa is 24 bits long.

The conversion (and normalization, for plotting purposes) of the ADC output into FP was performed using the formula

$$FP\ data = \begin{cases} float(\dfrac{ADC\ output}{2047.}) & ;\quad MSB = 0 \\ float(\dfrac{ADC\ output - 4095}{2047.}) & ;\quad MSB = 1 \end{cases} \tag{3.4}$$

3.3.2 Preemphasis

The second step in the signal conditioning stage is the preemphasis of the signal. During this step we pass the signal through a digital filter with the purpose of improving the signal-to-noise ratio (SNR). This step, or its equivalent, can be found in the literature under various names such as : matched filter, optimal filter, signal 'whitening', prewhitening, signal flattening, preemphasis,

amplification of high frequencies, etc [7,29,37]. The same technique, in its analog equivalent is used extensively in communication systems that employ frequency or phase modulation. In this thesis we adopt the term 'preemphasis'.

The improvement of the SNR, by using preemphasis, is highly desirable and it is invariably used for any sound signal processing. It has been concluded, based on experimental results [21,23], that the preemphasis of the signal is necessary for any signal processing. With proper preemphasis, the effect of the 'white' noise (equation 3.2) on the signal recognition is virtually eliminated. The low frequency background noise, which usually carries the most of the energy, may also be filtered out. This suggests a preemphasis filter with flat response at the lower frequencies, and an exponentially rising response at the higher frequencies (up to $\frac{\omega_s}{2}$) [52]. So, with proper preemphasis, only the background noise of high frequencies and the signal itself will be further processed.

It has been shown [21] that the preemphasis filter should be a simple first order inverse filter of the form :

$$H(z) = 1 + \alpha z^{-1} \tag{3.5}$$

Let d define the ratio of the highest to the lowest amplitude, with respect to ω, of the power spectral density, $P'(\omega)$ (equation 2.27), of the predicted signal, i.e.

$$d = \frac{\max_\omega P'(\omega)}{\min_\omega P'(\omega)} \tag{3.6}$$

Clearly, d is a measure of the spectral dynamic range [30]. An optimal value for α, in the sense of minimizing d, or equivalently maximizing the spectral flatness, is given by [30]:

$$\alpha = -\frac{R(1)}{R(0)} \tag{3.7}$$

where $R(0)$, $R(1)$ are defined by equation 2.18.

The coefficient α can be heavily quantized for any value between 0 and $-\frac{2R(1)}{R(0)}$ so as to reduce the dynamic range of the PSD [21]. Its numerical value depends on the signal being processed. An adaptive preemphasis filter could be the ideal solution. For that, we should first compute the $R(0)$, $R(1)$ of the signal and subsequently apply the preemphasis filter. The computation of the first two autocorrelation coefficients of the signal is a time consuming process (Table 3.1). For that, in implementing preemphasis we settle for a fixed value of α. Experimentally, we found that $\alpha = -0.96$, determined to be the average of the ratio $\frac{R(1)}{R(0)}$ of many signals tested. So, the preemphasis filter used is of the form :

$$H(z) = 1 - 0.96\, z^{-1} \tag{3.8}$$

and the preemphasized signal is given by:

$$s'(n) = s(n) - 0.96\, s(n-1) \tag{3.9}$$

For a random discrete-time signal passing through a digital filter with a transfer function (TF) $H(z)$ the following relations hold [3].

$$S_{out}(z) = H(z)\, H(z^{-1})\, S_{in}(z) \tag{3.10}$$

and

$$S_{out}(e^{j\omega T}) = |H(e^{j\omega T})|^2\, S_{in}(e^{j\omega T}) \tag{3.11}$$

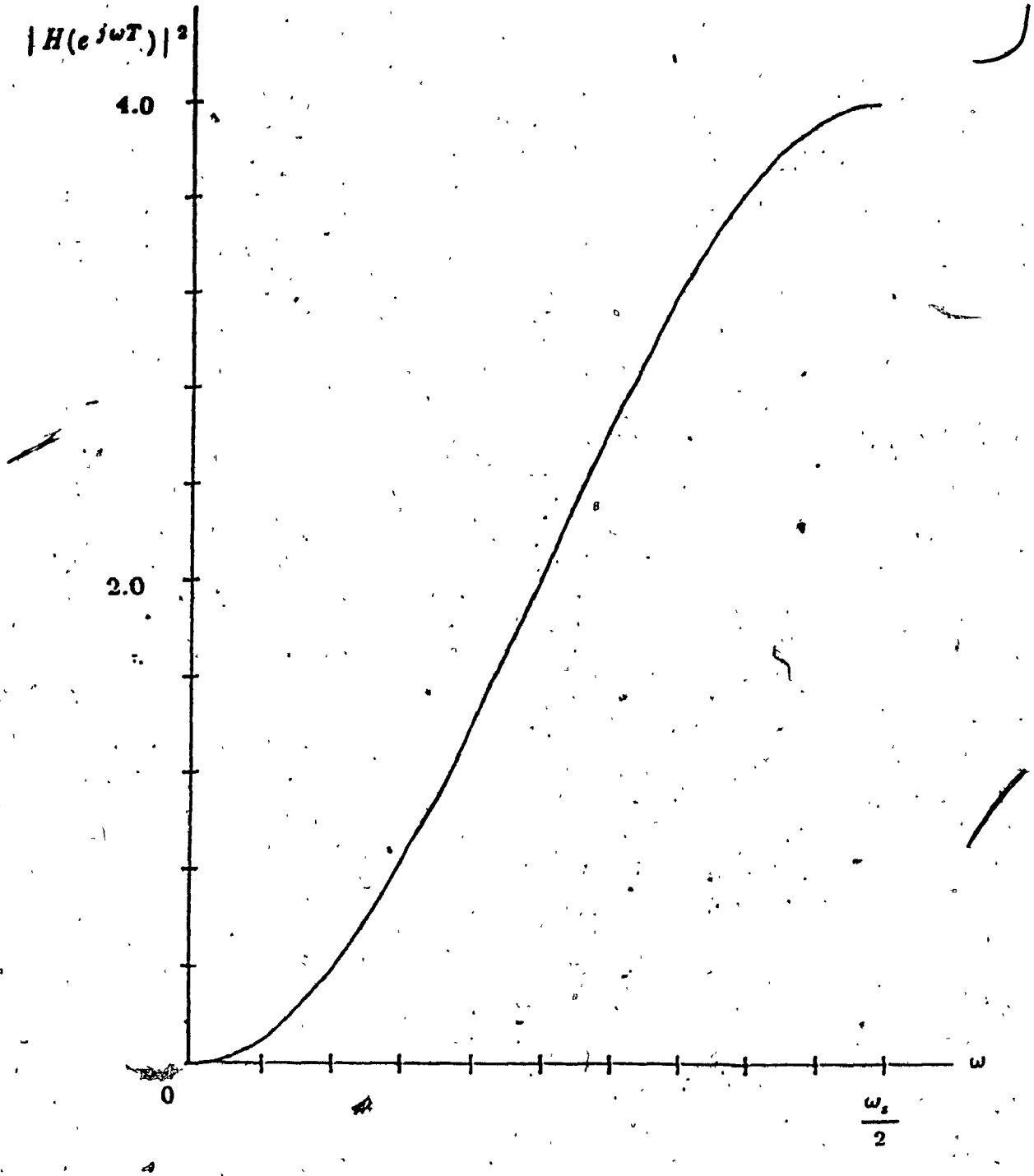


Figure 3.6

Amplitude Response of the Preemphasis Filter

Equation 3.11 shows that the PSD of a discrete-time random signal at the output of a digital filter is equal to the PSD of the input signal times the squared amplitude of the filter. From the above, we see that for the preemphasized signal we have :

$$S_{out}(e^{j\omega T}) = |1 - 0.96\, e^{-j\omega T}|^2 \; S_{in}(e^{j\omega T})$$

or

$$S_{out}(e^{j\omega T}) = [1.9216 - 1.92\, cos\,(\frac{2\pi\omega}{\omega_s})] \; S_{in}(e^{j\omega T}) \qquad (3.12)$$

by letting $T = \frac{2\pi}{\omega_s}$. The plotting of the bracketed term in equation 3.12 is shown in Figure 3.6, evaluated for $0 \leq \omega \leq \frac{\omega_s}{2}$. From this plot, we see that the low frequencies are almost eliminated and that the high frequencies are accentuated.

Summarizing this step, we see that the preemphasis of the signal is necessary for further signal processing. Preemphasis of the signal results in improvement of the SNR. The preemphasis filter is a first order inverse filter, which can be easily implemented. The implementation adds approximately 5% overhead in the overall signal processing (Table 3.1). There are some additional advantages of the preemphasis, such as system stability. These will be explained in section 3.4.2 (LP coefficients).

3.3.3 Windowing

This is the last step of the digital signal conditioning. In this step the

incoming signal (data) is partitioned (framed) by using a weighting function of the form:

$$w(n) = \begin{cases} w(n) ; & 0 \leq n \leq N-1 \\ 0 ; & \text{elsewhere} \end{cases} \qquad (3.13)$$

Windowing can be defined as the time-limited of a non-time-limited signal by using a bound limited multiplicative function (window).

As we have discussed in chapter 2, the assumed stationarity of the signal may be extended from 200 to 800 samples. Our model was built on the assumption that the signal processing is done piecewise (framewise), and that the temporal variation (from frame to frame) of its 'features' will be used for the recognition of the signal.

For the proper windowing of the signal, there are three subjects to be discussed. First, we discuss the desirable properties of the window function, second we give the reasons why a particular window function was chosen in the implementation of our model, and third, we define the numerical value for N (i.e. the sample size).

In time domain, the window function should have such a shape so as to smoothly taper (approximately to zero) the data at the sides of the frames. This is because, in estimating the first p values of the signal within a frame, we use the values of the signal outside the frame, which values we have set, arbitrarily, equal to zero (equation 2.19). If this tapering (smearing) is not done, then the expected error, in predicting the first p samples, could be large [50], which leads to a large overall predictive error. So, the first property of the window function is to have smooth and low side-lobes in the time-domain. Knowing that the 'features' will be computed from the PSD of the signal, then we should use a window function which could preserve the shape of the signal spectrum. Ideally,

the window function to perform that could have a Fourier transform a pulse at the origin. This, of course, corresponds to a 'white' noise (of amplitude $\frac{1}{2}$) in the time-domain. So, the second requirement is for the window function to have a pulse Fourier transform. These two requirements may define the window function. But they are mutually exclusive, in the sense that there is not any known mathematical function satisfying these two requirements simultaneously. A relaxation of the two requirements should lead in the search of a time-limited function which

I) In time-domain, could have smooth side-lobes and could taper the first p values of the signal close to zero, and

II) In frequency-domain, could have a narrow main-lobe, and smooth and low side-lobes.

This is equivalent to searching for a function which in both the time-domain (bounded) and in the frequency-domain (unbounded) from $-\omega_s$ *to* $+\omega_s$, has approximately the same shape. This shape is characterized by smooth and low side-lobes, and by a narrow main-lobe.

The rectangular window (equation 2.12) does not satisfy any of the two requirements for the function's shape. In the time-domain, it does not taper the p initial values of a frame. In the frequency-domain, although it has a narrow main-lobe, it has high side-lobes, i.e. it has high ripple ratio, which is the ratio of the maximum side-lobe amplitude over the main-lobe amplitude. For a time window of 62 samples this ratio is approximately equal to 0.22 [3]. This, relatively high, ratio may lead to spectral distortion. A window function which satisfies both relaxed requirements is the Hamming window. In discrete time-domain the Hamming window function is given by:

$$w(n) = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) & ; \quad 0 \leq n < N \\ 0 & ; \quad \text{elsewhere} \end{cases} \tag{3.14}$$

The plots of the window chosen in time and in frequency domain are shown in Figure 3.7, and Figure 3.8. The Hamming window satisfies the time domain requirement, in the sense that it tapers the first p samples in a frame. It, also, satisfies the frequency domain requirement, in the sense that its Fourier transform has smooth side-lobes, and it has small ripple ratio (=0.0082, for 62 points, i.e. the maximum side-lobe has an amplitude 42 dB below its peak value) [3]. Its drawback, in comparison to rectangular window, is that it has a wider main-lobe, and it also adds more attenuation. Considering that the Hamming window function satisfies the two requirements (with some drawbacks), we have chosen this function to frame our signals.

So, after the windowing step, the signal to be further processed is given by:

$$s'(n) = s(n)\ w(n) \tag{3.15}$$

where $s(n)$ in the last equation is the preemphasized signal.

Observing equation 3.14, we see that it is symmetrical over the point $\frac{(N-1)}{2}$. So, in implemehting the data windowing, we reserved $\frac{(N-1)}{2}$ main memory locations to store the values of the window function up to the value $n = \frac{(N-1)}{2}$. Because these values are fixed for a given N, they are calculated once, and they are stored in main memory before the signal processing starts.

To determine the value for N, we know (chapter 2) that, for a given sampling frequency (in our case $f_s = 6.25$ KHz), the maximum value for N

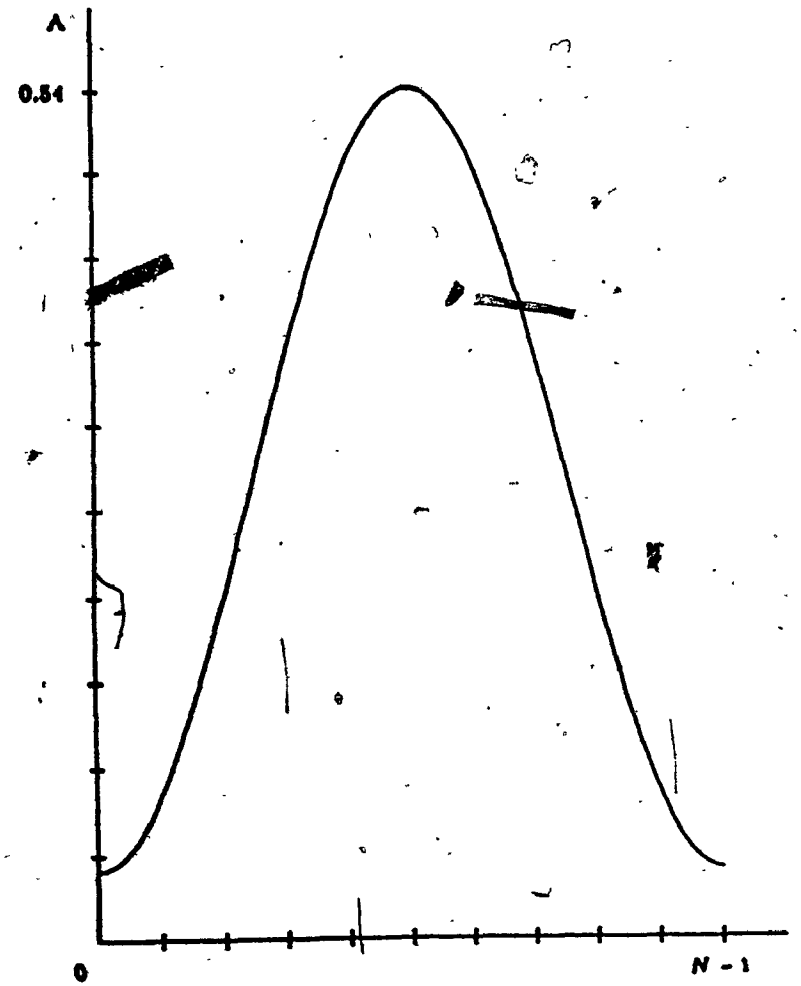


Figure 3.7

Hamming Window - Time Domain $N = 256$

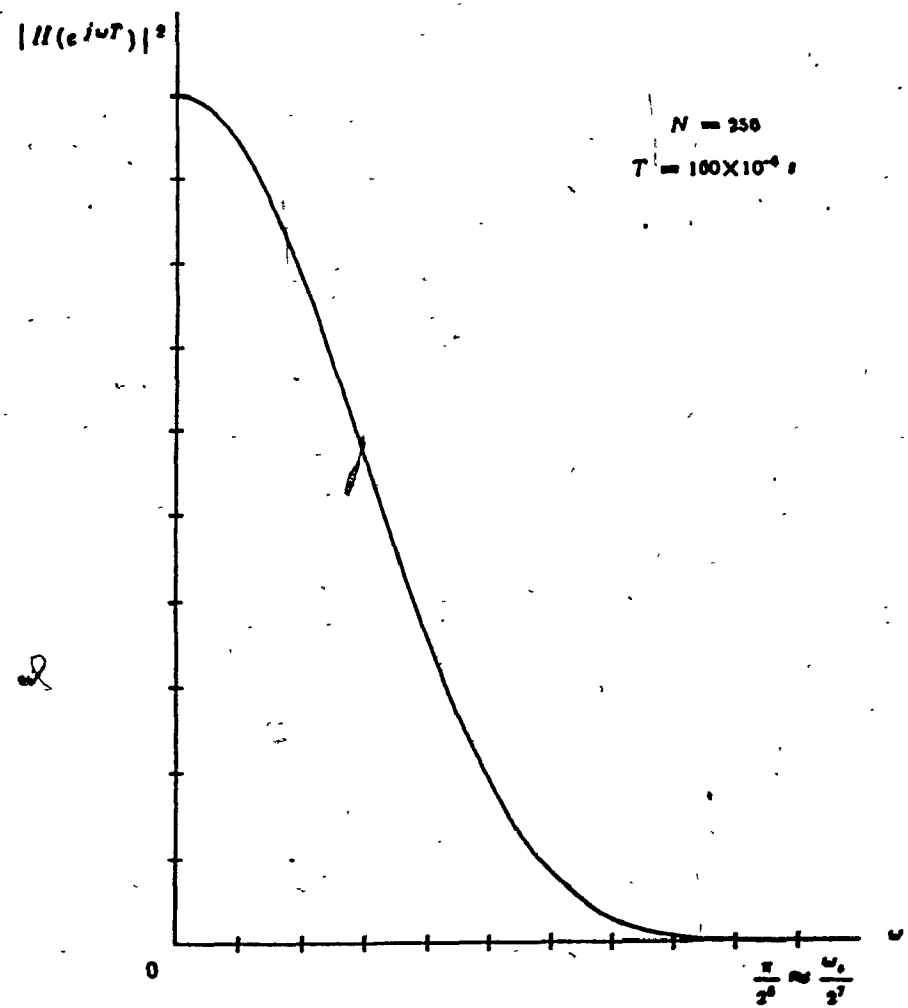


Figure 3.8

Hamming Window - Frequency Response

should be around 250 samples (assuming stationarity of the signal up to 40 ms). In the computer used for the signal processing, the record length is 512 data points. This led us to chose $N = 256$, which corresponds to half a record length, and it is also in accordance with the analysis carried out previously. For this value of N, 128 memory locations are reserved for storage of the values of the window function.

Summarizing this step, we see that the Hamming window function minimizes the mean square error and preserves the PSD shape. In this step we also determined the numerical value for the frame size N ($N = 256$). Implementation requirement for data windowing is 128 main memory locations. Windowing adds (Table 3.1) approximately 5% overhead in the overall signal processing.

3.3.4 *Autocorrelation Coefficients Computation*

The computation of the autocorrelation coefficients (R) of the preemphasized and windowed signal is the last step of the signal preprocessing. Two subjects are discussed here to cover this step. First, we give an explanation of what the autocorrelation coefficients are, and second, we determine the order of the prediction, p. This determination is done in conjunction with the analysis carried in chapter 2, and with the requirements of the environment of implementation.

Equation 1.18

$$R(l) = \left(\frac{1}{N}\right) \sum_{n=1}^{N} s(n)\, s(n+l)$$

may be modified as:

$$R^*(l) = \sum_{n=0}^{N-l-1} s^*(n)\; s^*(n+l) \;;\quad 0 \le l \le p \qquad (3.16)$$

where first, the factor $\frac{1}{N}$ has been dropped because this factor appears in both sides of the equation 2.17 and hence it can be omitted, second, the upper limit is $(N-l-1)$, because when we window the signal we, in effect, set the values outside the range zero to $(N-1)$ equal to zero, and third, the signal function $s(.)$ has been substituted by the windowed function $s^*(.)$ (equation 3.15). Obviously, $R^*(l)$ and $R(l)$ are the same within a factor, i.e.

$$R^*(l) = N\ R(l) \tag{3.17}$$

(assuming that preemphasis does not alter the validity of equation 2.18, and neglecting it's effect on the values of $R(.)$'s). So, instead of computing $R(l)$, we compute $R^*(l)$, which we call by the same name, and we will use the same notation (i.e. autocorrelation coefficient and the term $R(l)$ will refer to both $R(l)$ and $R^*(l)$).

The autocorrelation coefficients and the PSD $(S(\omega))$ of a sampled signal (under the assumptions of chapter 2) are related as follows [36,52]:

$$S(\omega) = \sum_{l=0}^{N-1} R(l)\ e^{-jl\omega T} \tag{3.18a}$$

and

$$R(l) = \frac{1}{\Omega} \int_{-\Omega}^{\Omega} S(\omega)\ e^{jl\omega T}\ d\omega \tag{3.18b}$$

or

$$R(l) = \frac{1}{N} \sum_{n=0}^{N-1} S(\omega)\ e^{jl\omega nT} \tag{3.18c}$$

where $\Omega = \frac{\pi}{T}$ (i.e. the Nyquist frequency).

For real signals the exponential terms, in the above equations, reduce to cosine terms. From equation 3.18a we see that the PSD of a sampled signal is the Fourier transform of its autocorrelation, i.e. the autocorrelation (in time domain) is the equivalent operation to Fourier transform (in frequency domain). From equation 3.18b we see that $R(0)$ represents the (average) power of the signal. So, the computed value for $R(0)$, in any frame, indicates if there is a detectable signal or only noise present. $R(0)$ may also be used to detect the beginning and the end of a signal. This property of $R(0)$ was used in the experimental part.

As we mentioned previously, for the computation of the autocorrelation coefficients we use equation 2.16. There are $(p+1)$ autocorrelation coefficients to be computed for each frame of data. So, at this point, we shall determine the order of the prediction p. In implementing equation 3.16, approximately $N(p+1)$ multiplications and $N(p+1)$ additions are required. As we see from Figure 3.10 and Table 3.1, the curve of the number of arithmetic operations required for the computation of the autocorrelation coefficients vs the number of poles has a constant slope (slope $= 2N$) for $p << N$. This slope decreases slightly for larger p. The slope of the other steps is at most N. So, in determining the value of p, the overhead being inserted for the computation of autocorrelation coefficients is considered heavily. From the above analysis we see that the overhead added to the system for the computation of the autocorrelation coefficients is relatively high. Hence, the upper bound for p should coincide with the absolute minimum bound required, so as to increase the efficiency of the system. From Figure 2.3 we see that the curve of the minimum normalized error vs the number of poles has a small slope for $p > 4$. So, $p = 4$ is the lower minimum requirement. Experimentally, we have found that we take good recognition results by using only the fundamental and the first two harmonic

frequencies of the signal. We reserved an additional frequency for tolerance. With that, $p = 10$ is the upper minimum requirement, which conforms with the requirement $p \geq 4$. Increasing p beyond 10, we found experimentally only a slight improvement in the results, and for purely stationary signals we did not find any improvement. Considering the overhead added for the computation of the autocorrelation coefficients we decided to set $p = 10$. This decision was supported also by considering the additional memory required for increasing p. For each frame processed, we reserve $(p+1)$ memory locations for the autocorrelation coefficients and p locations for the linear prediction coefficients a. So, for the whole signal, which may be extended over many frames, the saving in computer memory may be considerable. Hence, with the above reasoning we choose $p = 10$, which represents the minimum requirement.

From Table 3.1 we see that, for $p = 10$, the load to the computer for the calculation of the autocorrelation coefficients only is ~ 54% of the total load. From that we see that a parallel computer architecture should be considered for the signal processing.

Summarizing this step, we see that the computation of the autocorrelation coefficients is a compute intensive process. Mainly because of that, we selected as the order of prediction the upper minimum value defined from the analysis in chapter 2, i.e. we selected $p = 10$. The values of the autocorrelation coefficients will be passed to the next step of the signal processing, where they will be used to compute the linear prediction coefficients of the predictive filter. The $R(0)$'s will be also used as an end-detector.

3.4 Signal Main-Processing

In the second stage of the signal processing we cover the steps taken for the signal processing working first on the covariance domain, and second on the frequency domain. This stage covers the signal processing from the computation of the autocorrelation coefficients up to the computation of the 'features'. The input to this processing stage is a vector of autocorrelation coefficients and the output is a vector of 'features'. By 'features' we mean the location on the frequency axis of the relative maximum power concentration.

Four steps characterize this stage, namely: the end detection, the normalization of the autocorrelation coefficients, the computation of the LP autocorrelation coefficients, and the computation of the 'features'. These steps are depicted in Figure 3.9. Each step is examined, in detail, separately. For each step we reason for its necessity. We also reason why some choices were made. In analyzing the steps of this stage we take heavily into consideration the computer time/memory required for implementation.

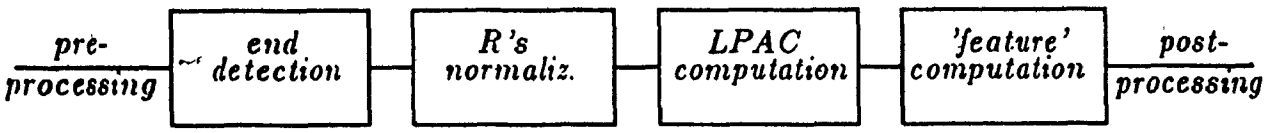


Figure 3.9
Signal Main Processing Steps

3.4.1 *End-Detection - Normalization*

The last step of the preprocessing stage is the computation of the autocorrelation coefficients R's. The computed values are passed to the main processing stage. In the first two steps in the main-processing stage we use the computed values of $R(0)$ as an end-detector, and we also normalize the autocorrelation coefficients with respect to $R(0)$. We included the end-detection and normalization steps in the main-processing stage for two reasons. First, because their operands belong to the covariance (autocorrelation) domain. This is in accordance with the classification of the stages according to the domain of the operands done previously. Second, in a multi-processor environment we could dedicate one of the processors to compute the autocorrelation coefficients only. In such a case, a relatively simple processor could perform this computation which is compute intensive.

In the end-detection step, the values of the $R(0)$'s are used to detect the beginning and the end of the signal. This detection is done framewise, i.e. we detect in which frame the signal starts/ends, but we do not have any information about the particular sample within the frame the signal starts/ends. The computed values of $R(0)$'s are also used to determine the existence of a signal in a frame. As we mentioned in the previous step, the value $R(0)$ represents the intensity (average power) of the signal in a frame. The end-detection is accomplished by comparing the value of the incoming $R(0)$'s with a threshold value. The threshold level value is fixed in our algorithm, and it represents the upper limit of the noise intensity found in the environment during the experimentation stage. We experimentally found that the noise intensity was affecting only the four least significant bits (out of 11) when sampled. So, the threshold level was set at the approximate value 0.0001 ($\sim (\frac{2^{11}}{2^4})^{-2}$).

By comparing the incoming values for $R(0)$'s with a threshold level, we take the following decision. If $R(0) >$ threshold then we proceed to the signal processing. If $R(0) \leq$ threshold then we signal to the previous stage to stop computing the rest of the autocorrelation coefficients for that frame.

In the normalization step we use the values for $R(0)$ to normalize all the autocorrelation coefficients. The following relation holds :

$$R(0) \geq |R(l)| \quad ; \quad 0 \leq l \leq p \tag{3.19}$$

Normalization of the autocorrelation coefficients does not effect the solution of the system of equations 2.19, but it does have an effect on the minimum error (equation 2.21). Knowing that the minimum error is a monotonic function of p and using equation 3.19 we take for the normalized total least-squared error

$$|E_m| \leq 1 \tag{3.20}$$

From the last equation we see that normalization of autocorrelation coefficients gives a common measure (in all frames) for the E_m. This common measure was used to test the validity of the implemented algorithm. The normalization step is also useful in the sense that, for various signal intensities, we take $R(0) = 1$. So, in further processing, we have a marginal speed-up because multiplication/division by $R(0)$ is not necessary.

Summarizing these two steps we see that our end-detection scheme uses a threshold level. The normalization of the autocorrelation coefficients results in the normalization of the minimum MS error. We implemented the normalization of the autocorrelation coefficients because it resulted in faster (in the order of a fraction of a percentage point) processing.

3.4.2 *Linear Prediction Autocorrelation Coefficients Computation*

In this step the normalized autocorrelation coefficients (R's) are used to compute the LP autocorrelation coefficients (LPAC) (a's). To cover this step we, first, explain the algorithm used for the computation of the LPAC's, and, second, we examine some aspects of the stability of the system of linear equations.

The computation of the LPAC's is done using equation 2.17,

$$\sum_{k=1}^{p} a_k \, R(l-k) = R(l) \; ; \quad 1 \leq l \leq p$$

which can be written as:

$$\mathbf{R}_{p \times p} \; \mathbf{a}_p = \mathbf{r}_p \tag{3.21}$$

where

$$\mathbf{R}_{p \times p} = \begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p-1) \\ R(1) & R(0) & R(1) & \cdots & R(p-2) \\ R(2) & R(2) & R(0) & \cdots & R(p-3) \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ R(p-1) & R(p-2) & R(p-3) & \cdots & R(0) \end{bmatrix}_{p \times p}$$

$$\mathbf{a}_p = [a_1 \; a_2 \; \cdots \; a_p]^t$$

and

$$\mathbf{r}_p = [R(1) \; R(2) \; \cdots \; R(p)]^t$$

The unknown a's are computed using the following (Durbin's) recursive formulae [17]

$$a_s^s = \frac{R(s) - \sum_{k=1}^{s-1} a_{s-1}^k \, R(s-k)}{1 - \sum_{k=1}^{s-1} a_{s-1}^k \, R(k)} \; ; \quad s = 1, 2, \ldots, p \tag{3.22a}$$

$$a_s^r = a_{s-1}^r + a_s^s \, a_{s-1}^{s-r} \; ; \; r = 1, 2, \ldots, s-1 \tag{3.22b}$$

with initial value :

$$a_1^1 = -R(1) \tag{3.22c}$$

In the above equations, the quantities $a_s^1, \ldots, -a_s^s$ are the LPAC's of order s, while the $a_2^2, \ldots, a_p^p$ are estimates of the partial LPAC's. The computation of a's (equation 3.21), which correspond to $-a_p^r$ $(r = 1,2,\ldots,p)$ in the recursive formulae, can be implemented in a computer environment very efficiently. The algorithm used for that is explained, in detail, in Appendix A.1.

As we see from Table 3.1 the number of calculations needed for the computation of the a's is of the order $O(p^2)$. This means that, for $p \ll N$, the time required for the computation of the LPAC's is very small compared to the time required to compute the R's $(O(Np))$. The load to the system for the computation of the a's is approximately 2.5% of the overall load, and the memory required to implement Durbin's algorithm is of the order of $O(p)$. From the above we conclude that we can solve the system of equations 3.21 relatively (to the other steps) very fast using a general purpose computer.

One criterion for the stability of the system 3.21 is the numerical value of the determinant of the square matrix. Observe that

$$\mathbf{a}_p = \mathbf{R}_{p\times p}^{-1} \, \mathbf{r}_p \tag{3.23}$$

where

$$\mathbf{R}_{p\times p}^{-1} = \frac{[cofactors]}{\Delta \mathbf{R}_{p\times p}} \tag{3.24}$$

So, if the determinant of the square matrix is a very small number, we should expect ill-conditioning of the system 3.23. The matrix $\mathbf{R}_{p\times p}$ is a positive

semidefinite matrix [50]. For the determinant of $\mathbf{R}_{p\times p}$ the following relations hold [37]:

$$0 \leq \Delta \mathbf{R}_{p\times p} \leq 1 \tag{3.25}$$

The upper equality is true for orthogonal data (i.e. $\mathbf{R}_{p\times p}$ is a diagonal matrix), and the lower equality is true fol linearly dependent data.

It has been found experimentally [24] that there is a significant autocorrelation between adjacent samples. This means that for most of the signals, we should expect

$$\Delta \mathbf{R}_{p\times p} \rightarrow 0 \tag{3.26}$$

The above relation introduces instability to the system of equations 2.17, i.e. a small fluctuation in the values of R's may have a large effect on the computed values for the LPAC's. This fluctuation may be the result of the word length of the computer (truncation/rounding). It has been suggested [45] that, given sufficient computer word length, the system is stable. We, experimentally, found that the system was unstable, but this did not have a big effect on the resulting shape of the PSD. These results were taken by using preemphasis. Without preemphasis, we found that the system was extremely ill-conditioned. Small variations on the R's did not only affect the a's but, also, the resulting PSD's did not have similarities. So, preemphasis of the signal results in decreasing the ill-conditioning of the system. This is an additional (to the reasons given in section 3.3.2) reason for implementing preemphasis. This implies that, for the same system stability, the computer word length required may be reduced by using preemphasis.

As a measure for testing the ill-conditioning of the system we can use the

variation of E_m (equation 2.21). We mentioned previously that the E_m is a monotonically decreasing function of p, i.e.

$$E_m^p \leq E_m^{p-1} \leq \cdots \leq E_m^1 \tag{3.27}$$

where the superscripts denote the order of estimation. So, we can calculate the various E_m^k's $(k = 1, 2, \ldots, p)$, and upon finding that relation 3.27 is not satisfied, we can deduct the ill-conditioning of the system. Observe that the partial LPAC's (i.e of order less than p) are calculated as an intermediate step on the recursive algorithm (equation 3.22b)). This test is not an exhaustive test and gives only a partial measure of the stability of the system. It was implemented in our algorithm because it can be done fast and requires only p memory locations for each frame processed.

To conclude this step we see that a fast algorithm (Durbin's) is used to compute the a's from the R's. The order of the calculations required is $O(p^2)$ and of the memory required is $O(p)$. The system appears to be ill-conditioned. A fast, but not exhaustive, test for the system instability was implemented. To complete this step we add to the computer system an additional load less than 2.5% of the overall load.

3.4.3 'Feature' Computation

This is the last step of the main processing stage. In this step we compute the 'features' for each frame of the signal, where we have defined as 'feature' a vector of the peak locations of the PSD. To complete this step, first, we evaluate the PSD of the signal in a frame using the LPAC's, and second, we make peak picking from the resulting PSD. A vector of LPAC's is the input in this step. It

outputs a vector of the location, on the ω-axis, of the PSD peaks. For the analysis below the computer time/memory requirements for this step are considered.

The PSD evaluation of a signal frame is done using equation 2.27. Rewriting this equation we take:

$$P'(n\,\Omega) = \frac{1}{\rho(0) + 2\sum_{k=1}^{p} \rho(k)\,\cos(kn\,\Omega)} \tag{3.28}$$

where

$$\rho(k) = \sum_{l=0}^{p-k} a_l\, a_{l+k}\;;\quad a_0 = 1$$

$$0 \le n \le \frac{L}{2} - 1$$

$$\Omega = \frac{2\pi}{L}$$

and L is the number of discrete points on the $\omega\ (= n\,\Omega)$-axis spanning the frequency range $0 \rightarrow \omega_s$. The function $P'(n\,\Omega)$ is evaluated on the upper unit semi-circle (i.e. on $e^{jn\Omega}$). This is done because the function $P'(n\,\Omega)$ is even and periodic (as we have mentioned in chapter 2).

For each frame we use $(\frac{L}{2})$ discrete frequency points to cover the baseband $0 \rightarrow \frac{\omega_s}{2}$. We have chosen $L = N = 256$. Hence, two adjacent points of the PSD are ~ 24 Hz apart (3125 Hz / 128 points). This resolution was a compromise between the computer time required for the evaluation of the PSD and the precision of the algorithm. As we see from Table 3.1, the slope of the curve [number of calculations required for the PSD evaluation] vs N is equal to $(\frac{p}{2} + 1)$. For

$p = 10$, this slope is equal to 6. If we double the points in which we evaluate the PSD, this could result in a six-fold increase in the required computer time for the evaluation of the $P'(n\,\Omega)$. So, the trade-off done in this step is between the speed of processing and the accuracy of the results.

Equation 3.28 shows that, for the evaluation of the estimated PSD we perform an autocorrelation on the LP autocorrelation coefficients with the addition that we weight the partial sums. The weighting function is $cos\,(kn\,\Omega)$ (or 1). From this we see that, in a multiprocessing environment, the processors dedicated for the computation of the R's and the ones dedicated for the computation of the $P'(n\,\Omega)$'s could have the same architecture. This architecture could be simple because of the limited number of instructions needed to perform the autocorrelation.

Again from equation 3.28 we observe two additional things. First, the values of the weighting function, used over and over for all frames, are fixed for all frames, and second, the values of the autocorrelation of the LPAC's, used over and over within a frame for various weights, are also fixed (for one frame). From the first observation we conclude that we can store in an array (of dimension $\frac{L}{2} \times p$) in memory the values of the weighting function. These values may be stored permanently in memory because they are used for all frames. From the second observation we conclude that a temporary storage is required (length $\frac{p^2}{2}$) to store the values of the autocorrelation. These values change from frame to frame.

The computed values of the PSD of a frame are stored temporarily in memory ($(\frac{L}{2})$ memory locations required). On this data we perform the peak picking. The algorithm implemented to perform the peak picking is simple and

works as follows: If in a given frequency the amplitude of the PSD is larger than the amplitude of the PSD in the two adjacent frequencies then we consider this as a peak. In an array we store the peak's frequency, their weight, and the total number of peaks found in a frame. This tri-dimensional array is passed to the post processing stage. The problem with this algorithm is that two characteristic frequencies located close to each other (in the range of 36 Hz) will not be distinguished. For pure sounds whose fundamental frequency is in the range of a few hundred Hz this problem does not appear, and the results taken are satisfactory. For more complex sounds, this problem can be easily detected in that not all peaks may be found (Tables 4.1, 4.2, 4.3).

Concluding this step we see that the PSD is computed using the LPAC's evaluated in the previous step. The peak picking is done on the PSD. 'Features' passed to the post processing stage is a vector defining the location of the peaks. In addition to this vector, the amplitude and the number of the peaks found as well as the normalized error of the frame are passed to the next stage. Depending on the accuracy of the results desired, implementation of this step may result in extremely large computer time consumption, and in a large number of memory locations required to store the data. In our implementation we chose a relatively small number of points to evaluate the PSD. After this choice, implementation of this step added approximately 28% load on the overall computer load. The memory required for storage of permanent and temporary data is of the order of $O(\frac{Lp}{2})$, or approximately 1500 memory locations.

	# of operations		Example: $L = N = 256$, $p = 10$ no. of operations		%		cumulative %	
Routine	*/	+-	*/	+-	*/	+-	*/	+-
Read and convert	N	$\frac{N}{2}$	256	128	4.9	2.8	4.9	2.8
Preemphasis	N	N	256	256	4.9	5.6	9.8	8.4
Window	N	-	256	-	4.9	-	14.7	8.4
Autocorrelation	$N(p+1)$	$N(p+1)$	2816	2816	53.8	61.3	68.5	69.7
LPAC Coefficients	p^2+2p-3	p^2+2p-3	117	117	2.2	2.5	70.7	72.2
Transfer Function	$\frac{L}{2}(p+2)$	$\frac{L}{2}p$	1536	1536	29.3	27.8	100	100
Totals (L = N)	$\frac{N(3p+10)}{2}+$ $+p^2+3p-3$	$\frac{N(3p+5)}{2}+$ $+p^2+2p-3$	5247	4597	100	100		

Legend :

L = resolution of the PSD

N = number of samples per frame

p = number of poles

Table 3.1

Statistical Analysis of the Required Average Number of Computer Operations per Frame

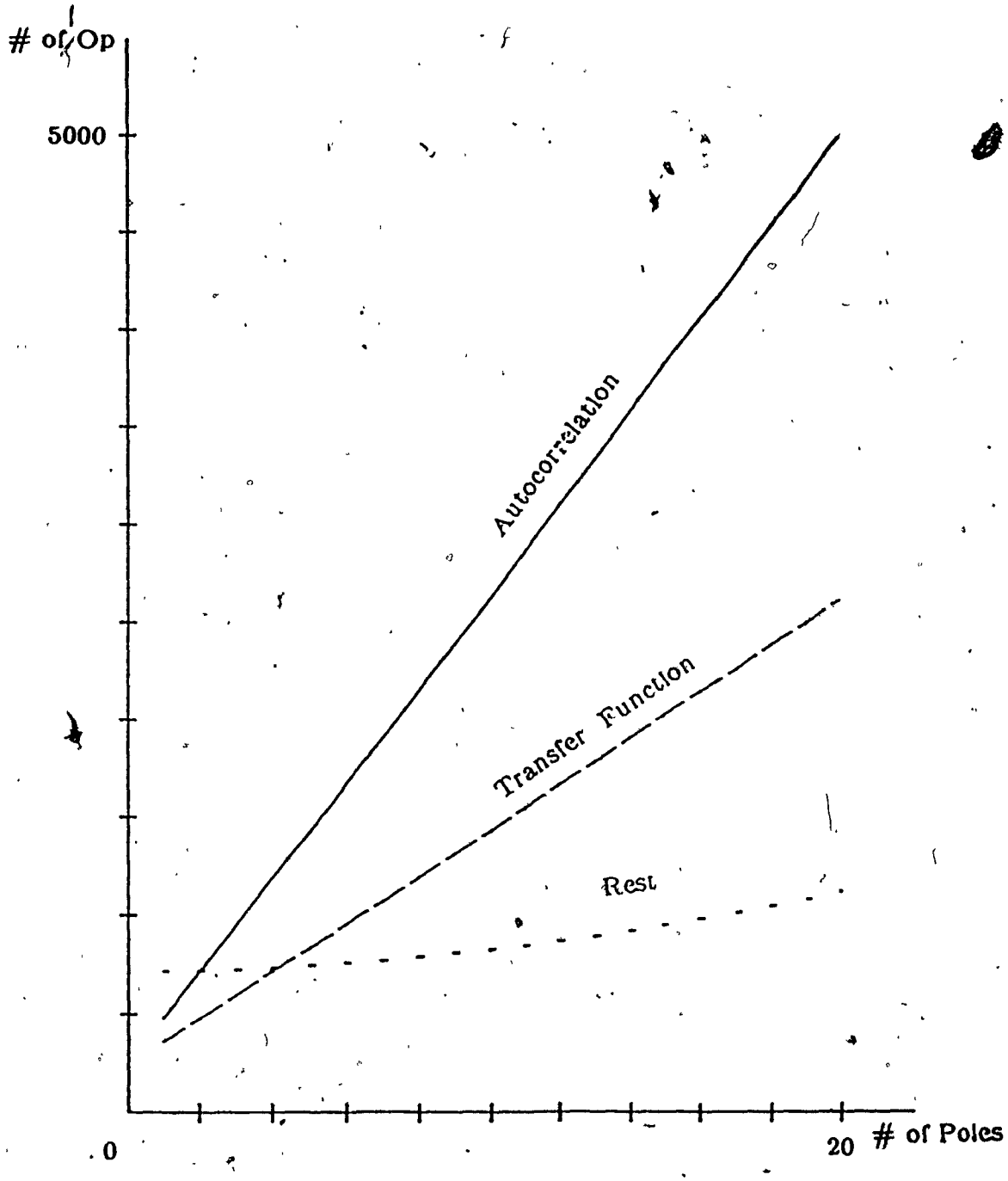


Figure 3.10

Computer Operations vs # of poles

3.5 Signal Post Processing

The signal post processing stage is the final stage of the overall signal processing. The input to this stage is a set of 'features' derived in the previous stages. The main 'features' of the signal is a vector of the peaks' frequency for each frame. Additional information referring to the signal and used in this stage include, first, the temporal range of the whole signal, second, the relative power in a signal frame, third, the amplitude of the normalized error in a frame, and fourth the absolute amplitude of the peaks. The above 'features' are used for possible signal identification (labeling). The decision taken by the algorithm for the nature of the signal is the output of this stage.

The overall search algorithm for the signal labeling is as depicted in Figure 3.11. The two branches of the search algorithm are the fixed and the adaptive template matching. The fixed template matching deals with time-invariant signals, and the adaptive template matching with repetitive sound signals. These two branches will be examined next separately.

3.5.1 Fixed Template Matching

The first test the post-processing algorithm performs on the signal frames 'features' is the comparison of the normalized minimum frame error with a threshold level. Experimentally we set this threshold value equal to 0.55. When the normalized error in a frame is larger than the preset threshold value, then the algorithm does not process the information for this frame further. The reason for this is that the parameters derived for a segment of the signal do not correspond

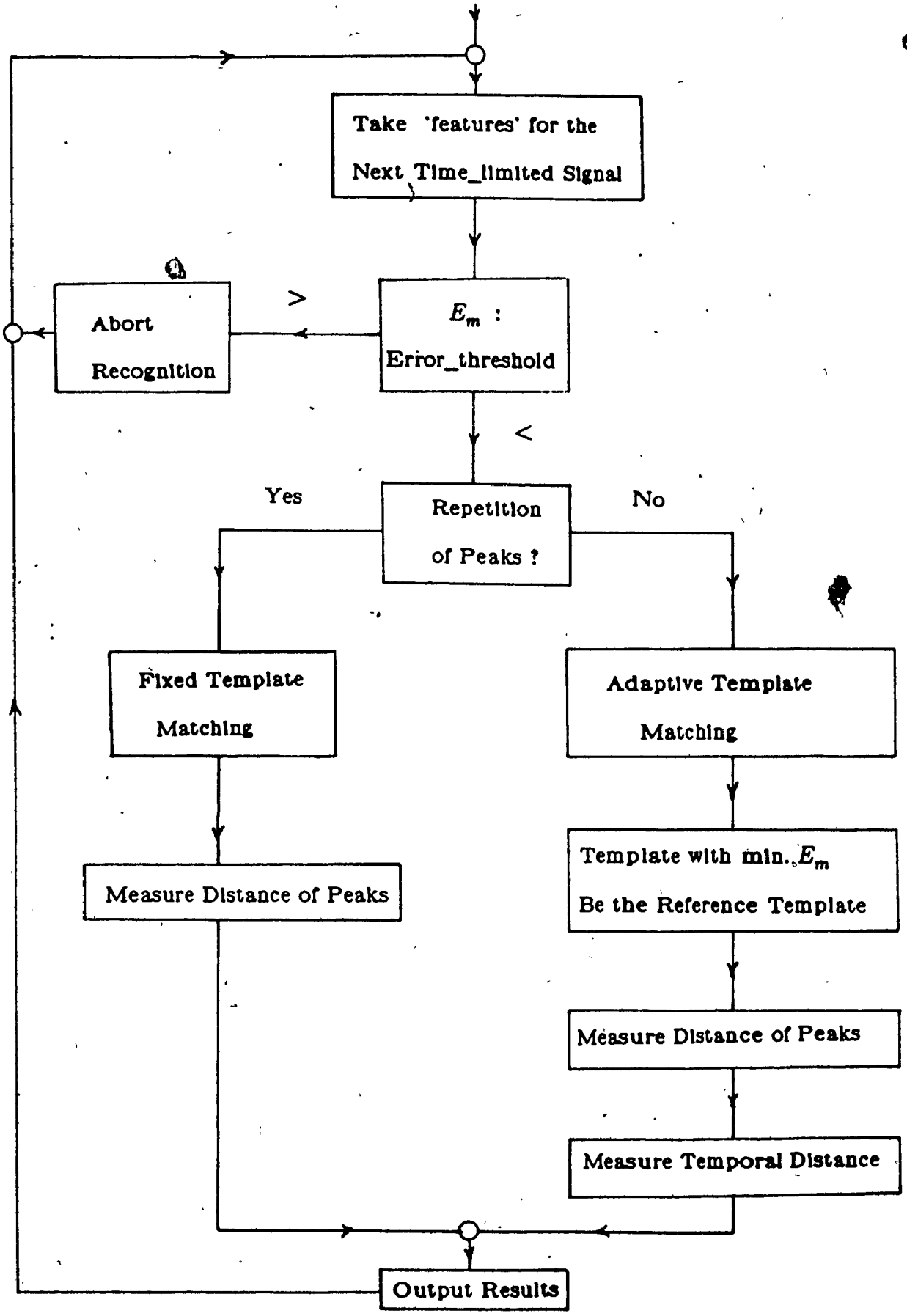


Figure 3.11

Compact Flowchart for the Signal Post Processing

to the real parameters of the signal segment. This becomes obvious when we consider that when the normalized minimum error is (ideally) equal to zero, then the parametric modeling is absolutely successful. On the other hand, when the normalized minimum error is equal to one, then the parametrization of the signal is absolutely unsuccessful.

Experimentally we found that there are two main factors which may influence the magnitude of the normalized minimum error in a frame. These factors are: first, is the signal-to-noise ratio, and second the dynamic range of the signal. The larger the signal-to-noise ratio or the dynamic range of the signal, the smaller is the normalized minimum error in a frame.

The algorithm considers that the parametrization of the signal is partly successful when $E_m \ll 0.55$, and it continues processing the 'features' of the frame, otherwise it aborts the processing and continues with the next frame.

The second test the algorithm performs is the checking for repetition of peaks in successive frames. When the algorithm detects a repetition of peaks, then it branches to the fixed template matching test, otherwise it branches to the adaptive template test.

In the fixed template testing, the signal 'features' are matched against a collection of reference templates. The reference templates, of which each one consists of a vector of the peak's frequency, and a vector of their magnitude, are stored in the memory of the machine. They have been created by the repetition of similar sounds and by averaging the outcome of the parametrization process.

Similarity of the reference template and of the test (signal) frame is defined as the distance of two vectors, having as components the location of the peaks on the ω -axis. The distance is measured in the following manner :

distance = || (reference template peaks) - (signal frame peaks) ||

The algorithm considers the signal frame similar to the template framer iff the distance is less than a threshold. The numerical value for the threshold has been determined experimentally and, in our case, it has been set equal to nine.

The acceptance or the rejection of the similarities is further tested by weighting the corresponding distances by the absolute magnitude of the peaks using the formula:

$$\text{score} = \frac{\sum <sf> <\text{template's absolute magnitude}>}{\sum <\text{template's absolute magnitude}>}$$

where

$$<sf> := \begin{cases} 0 & \text{for non similar frequencies} \\ 1 & \text{for similar frequencies} \end{cases}$$

Two frequencies (template's and signal's) are similar iff they are no more than 3 points apart. The score above is a weigthing function. It weigths the various absolute amplitudes stored in the templates with the similar frequencies.

The algorithm considers the reference template and the signal frame as similar when

$$\text{score} > \text{threshold\_score}$$

From experiments we found that the value 0.5 for the threshold_score gives good results, and thus we used this value in our algorithm. An observation at this point referring to the formula for the score is that the algorithm assumes similarity even when the reference template and the test template have only one peaks' frequency in common, with the understanding that the absolute amplitude of this frequency is at least as large as the total amplitude of the other peaks. This is a drawback of the algorithm, and it may be deceived in the respect that it will recognize a signal as similar to a certain template although the is not any

similarity. Experimentally we did not find this to be the case, but it may be.

The algorithm considers the similarity unambiguous when score $>$ 0.99. When $0.5 <$ score < 0.99 then this frame is conditionally accepted as similar to the reference template. In the case of the conditional similarity, the algorithm searches all the frames (from signal start to signal end) for conditional similarities and weights the total score in all frames. If the total score is at least half the possible maximum score, then this signal is accepted as similar to the reference signal.

An observation on the exhaustive algorithm is that in case of a non highly contaminated by noise signal (as the one shown in Figure 4.8) the algorithm recognizes the signal pattern from the first test and so it does not proceed with additional tests. But in case when the signal-to-noise ratio is relatively small, then the additional steps are necessary so that the final decision of the algorithm be as consistent as possible.

As it becomes clear from the previous discussion, the fixed template matching part of the algorithm deals with real time-invariant signals. The sound signals may include a telephone ring, a fire alarm, the honk of a car, a ring of a door bell, etc. The common characteristic of this set of signals is that as long as the sound generator is activated, then the sound produced has almost constant frequency components. Experimentally we found that at the very beginning and at the very end of the signal, when we have a transition period, the signal is not time-invariant (as expected). This was taken into account by the algorithm, and in some cases the boundary signal frames were disregarded. Another experimental observation is that when the signal starts in approximately the middle of a frame, then the frequency components in that frame are, usually, sharper than the frequency components in the frames which are composed entirely from the

time-invariant signal.

For the implementation of the fixed template test of the algorithm, the computer memory required is kp , where k is the number of time invariant signals whose 'features' have been stored in the machine. The CPU time required is negligible compared to the time required for the rest of steps for the signal processing (a fraction of a percentage point of the overall load).

3.5.2 Adaptive Template Matching

The adaptive template matching part of the algorithm attempts to recognize time-varying signals which are repetitive in time. In the adaptive template matching algorithm, fixed reference templates have not been stored in the machine. For the implementation of this algorithm, the 'features' for all the frames of the signal are needed. Also, the relative power of the frames is used. The algorithm picks one frame, among all the frames in the signal, with the smallest normalized minimum error, and considers it as the reference template. It tests the 'features' of the so chosen frame with the 'features' of the template with the second smallest normalized minimum error. The distance of the 'features' is measured the same way as in the fixed reference template algorithm. If this distance is less that the threshold level, then it measures the temporal distance between the reference and the frame under test. If the temporal distance is less than a threshold period of time, then it disregards this template as being part of the same excitations of the sound generator, and continues with the frame with the next greater normalized minimum error, and so on. If the temporal distance is greater than a threshold period of time, then the algorithm assumes that the frame under test belongs to another excitation of the same sound generator, it

considers the outcome as a success, and it continues the search with the next frame with the next greater normalized minimum error and so on.

The threshold period of time was set experimentally. During the experiments we observed that a freely vibrating sound generator keeps its characteristic frequencies for a very short period of time. For one or two frames at most, after each excitation, the same frequency components appear in the PSD. After that the PSD becomes flatter with each frame, and after, at most four frames, the sound is degenerated and no characteristic frequencies were observed. After these observations, we set the threshold period of time equal to 80 ms (4 frames distance).

The computer time required for the implementation of the adaptive template matching algorithm is very small compared to the other steps. The memory required is of variable length and depends on the length of the signal. If we assume that the sound goes on for 2 s, then the memory required is approximately 500 memory locations.

3.6 Conclusions

In this chapter we examined all the necessary steps taken for the signal processing. We separated the overall signal processing in three main stages; the pre-, the main-, and the post-processing depending on the domain of the variables the algorithm operates upon. Deterministic values were given for the two variables left undefined in the model building stage; namely the order of parameters and the frame size. The back end of the algorithm uses as main 'features' a vector of peak locations for each frame. Supportive components to this vector are the power distribution and the normalized minimum error (framewise). Finally, for each step taken the CPU time and the memory required were estimated.

CHAPTER 4

RESULTS - DISCUSSION

4.1 Results

The algorithm described in the previous chapter was implemented on a PDP 11/45 mini-ocomputer supported by an ADC, a user's control device (Figure 4.18), and a microphone (Figure 4.1). The software used can be found in Appendix B of the present thesis. Table 4.1 shows the results of the implemented algorithm scheme. These results were taken with 6.25 KHz sampling rate, single-precision FP arithmetic, SNR $\geq$ 0 dB, and resolution for the PSD equal to 256 points/frame.

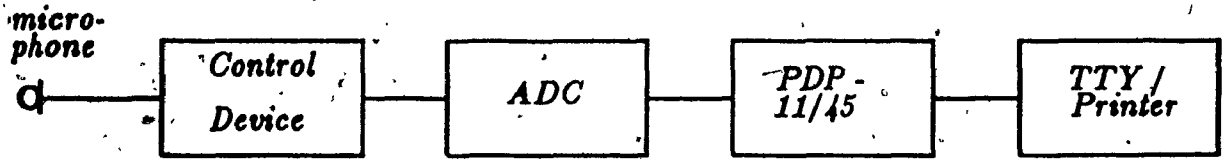


Figure 4.1

Block Diagram of the Hardware Set-up for Testing the Algorithm

We arrived at the following conclusions from the experimental part:

a) No significant improvement in the results was observed when the sampling frequency was increased from 6.25 KHz to 10 KHz.

b) The stability of the system of equations 2.20 was not affected when we used double-precision FP arithmetic instead of single-precision.

c) For SNR $\geq$ 0 dB the results taken are as shown in Table 4.1. For SNR $<$ 0 dB the success rate of the algorithm dropped almost exponentially.

d) For PSD resolution of 256 points/frame, some peaks were not found by the algorithm (see Tables 4.2, 4.3, and 4.4). This was due to either because the peaks were very close to each other or the PSD was very flat close to the particular peak. By increasing the PSD resolution to 1024 points/frame, almost all the peaks were found by the algorithm.

The noise in the room, where we performed the experiments, was measured over a 24 hours period in half an hour intervals. From the data taken, the average intensity (per frame) of the noise in the room was approximately 0.00006 V^2. This is the average value for R(0)'s, and it corresponds to approximately 30 dBs in sound intensity (assuming that the amplification factor of the amplifier used was equal to 425).

4.2 Analysis of the Results

4.2.1 Time Invariant Signals - Fixed Template Matching

As we see from Table 4.1 the algorithm was successful 100% of the times in recognizing time-invariant signals. These signals consisted entirely of telephone rings from various distances from the recording microphone.

The part of the algorithm dealing with time-invariant signals is the fixed template matching, and it is described in detail in section 3.5.1. It matches reference templates (which have been stored in the machine) with signal templates.

An example of a time-invariant signal (telephone ring) is shown in Figure 4.2. The signal to-noise ratio for this signal is approximately 15 dB in all frames. 20 successive frames of this signal and their respective PSD (both normalized) are shown in Figures 4.3 to 4.6. As we see from these figures, the characteristic frequency of the signal of approximately 1262 Hz appears in all frames. The peak frequencies of the templates stored were 1275, 1667, and 2990 Hz.

4.2.2 Time-varying Signals - Adaptive Template Matching

The adaptive template matching algorithm (section 3.5.2) was used for the recognition of repetitive sound signals. These signals were produced by repetitively knocking on various surfaces. We experimented on knocks on wood, on plastic, and on metal. An example of a repetitive sound signal is shown in Figure 4.7. There are not any reference templates for the signals to be recognized. The algorithm constructs a reference template from the frame with the least minimum error. From this frame, the frequency of the three most significant peaks of the PSD are extracted and used in the template. The reference template is used then to match the remaining frames accordingly to the algorithm as explained in section 3.5.2.

As an example the algorithm was successful in recognizing the time-varying signal shown in Figure 4.7. Frame 11 was used as the reference template in the adaptive template algorithm. The characteristic frequencies of the signal frames 2 and 20 were successfully matched against the characteristic frequencies of the reference template (502 and 1127 Hz). The signal-to-noise ratio in this signal has a maximum value of ~ 30 dB (in frame 11), and a minimum value of ~ 16 dB (in frame 1).

In Figure 4.12 another time-varying signal is shown. The algorithm failed to recognize this signal because the smallest value for the minimum normalized error E_m in the 20 frames was greater than the threshold value of 0.55. The smallest value for E_m among the 20 frames is in frame no. 15 and it is equal to 0.69. As we observe from Figures 4.13 to 4.16 the PSD of all the frames of this signal is very flat in comparison to the PSD of the two previous signals. This is in accordance with the curves of Figure 2.3, where in there it is clearly shown that noisy (flat) signals have a large E_m. The maximum value of the SNR in this signal is ~ 26 dB (frame 19), and the minimum value is ~ 15 dB (frame 4).

Signal	# of signals tested	# of successes	# of failures	% success
Time - invariant	20	20	0	100
Time - varying	284	187	97	66
Total	304	197	97	

Table 4.1
Results of Recognition

4.2.3 Model Verification

The validity of the modeling was verified using a digital waveform analyzer (Precision Data 6000, Universal Waveform Analyzer). This analyzer employs Discrete Fast Fourier Transform (DFFT) for the frequency analysis of the signals, and the window function used is a Hamming window. Short-time analysis (zero-

crossings) was also used to verify the modeling part of the algorithm.

4.2.4 Further Characterization of the Signals by Using Image Processing Techniques

Using image processing techniques (emulation of a vocoder), we arrived in the following inconclusive result. The time-varying signals (in the test case these signals were repetitive knocks on various surfaces) showed a characteristic pattern related to the material the sound generator is composed of. This characteristic pattern of the signals is that when the sound settles (trailing edge) then consistent patterns of frequencies were observed. When these patterns were observed the surface was vibrating freely. These patterns were different than the patterns generated when the surface was forced vibrating.

These distinct patterns are shown in Figures 4.17a and 4.17b. In each of these Figures, which one consists of five independent repetitive sound signals, the evolution of frequencies in time is shown. The vertical axis spans the frequency range from 0 to 3.125 KHz. In the horizontal axis, for each signal, 80 successive frames are plotted. The duration of each frame (in time domain) is approximately 10 ms. The different colors in these Figures signify the intensity (amplitude) of the frequencies. The black color indicates the smaller frequency amplitudes, and the white color indicates the higher amplitudes. The intermediate frequency intensities are shown in the scale of colors :

(black) - red - green - yellow - blue - magenta - cyan - (white).

Table 4.5 shows the frame number in which the stimulus was applied to the sound generator (beginning of signal).

The above observation is clearly visible in the Figures 4.17a and 4.17b. From these Figures we see that we have two concentrations of frequency amplitudes evolving in time. The concentration in the lower frequencies corresponds to the forced vibrations, and the concentration in the higher frequencies corresponds to the free oscillations of the sound generator. We have used the former patterns for the recognition scheme of the algorithm. The latter patterns may be used to further identify the sound generator. It is visible that the lower frequencies occur earlier than the higher frequencies which are attributable to bulk oscillations of the material (e.g. wood or metal or plastic). Figure 4.17a shows five independent knocks on wood, and Figure 4.17b shows five independent knocks on metal. From these Figures we see that the average frequencies of the knocks on metal are marginally higher than the average frequencies of the knocks on wood. This is true for both the forced and the free oscillations.

The inconclusiveness of this result is that the minimum normalized error of the frames containing the trailing edge of the sound is quite high (above 0.5). The reason for this is that the signal-to-noise ratio in these frames is, in comparison to the other frames, very small.

Nevertheless, the upwards shifting of frequencies when the material was allowed to oscillate freely, was consistently observed for all the cases.

Further investigation of this subject (using more precise hardware) may shed more information about the signal and its generator, so that a sound pattern recognition scheme may be deviced giving superior results than the scheme we employed in our algorithm.

Frame	Peak 1	Peak 2	Peak 3	Peak 4	
1	-	1262	1667	2953	Hz
	-	10.75	1.56	4.61	
2	588	1275	1667	2941	Hz
	0.04	31.05	2.26	4.82	
3	527	1275	1642	-	Hz
	0.06	18.76	4.81	-	
4	416	1275	1654	-	Hz
	0.03	8.47	27.59	-	
5	135	1275	1703	-	Hz
	0.03	110.59	0.58	-	
6	502	1250	1672	-	Hz
	0.05	27.14	2.32	-	
7	490	1262	1642	2855	Hz
	0.02	141.53	1.35	0.88	
8	576	1275	1630	2206	Hz
	0.13	115.70	1.10	0.12	
9	466	1275	1642	2426	Hz
	0.06	15.28	7.01	0.26	
10	490	1275	1642	2145	Hz
	0.01	191.02	0.87	0.05	
11	502	1287	1630	-	Hz
	0.07	23.07	1.70	-	
12	404	1275	1691	-	Hz
	0.02	91.25	1.91	-	
13	306	1262	2010	3002	Hz
	0.06	12.20	0.46	6.17	
14	551	1275	1680	-	Hz
	0.13	10.51	6.98	-	
15	588	1275	1703	-	Hz
	0.07	66.39	0.17	-	
16	392	1262	1667	-	Hz
	0.13	5.63	3.65	-	
17	135	1275	1716	2610	Hz
	0.04	38.81	0.88	0.37	
18	270	1275	1654	2426	Hz
	0.04	75.69	1.22	0.17	
19	208	1275	1630	2941	Hz
	0.03	74.26	0.88	1.20	
20	319	1260	1642	3040	Hz
	0.10	22.08	2.67	4.24	

Table 4.2

Peak's Location and Absolute Magnitude
for the Time-Invariant Signal of Fig. 4.3

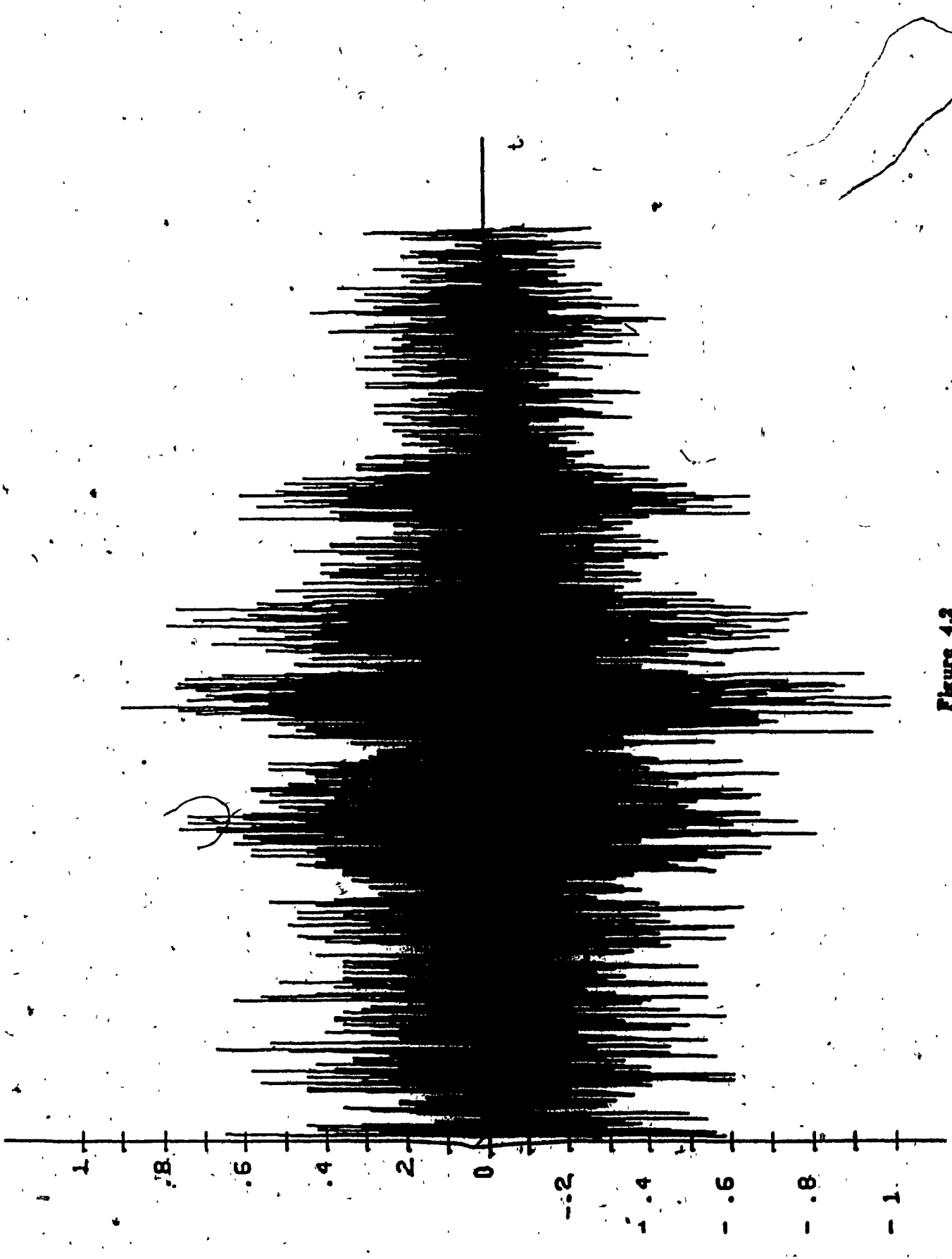


Figure 4.2

Sampled Time-invariant Signal (20 Frames; Total duration = 0.82 s).

1	2	3	4	5
$E_m = 0.28174$	$E_m = 0.14389$	$E_m = 0.20653$	$E_m = 0.16643$	$E_m = 0.10816$
$R = 0.002295071$	$R = 0.005584199$	$R = 0.003983119$	$R = 0.003286266$	$R = 0.004726660$

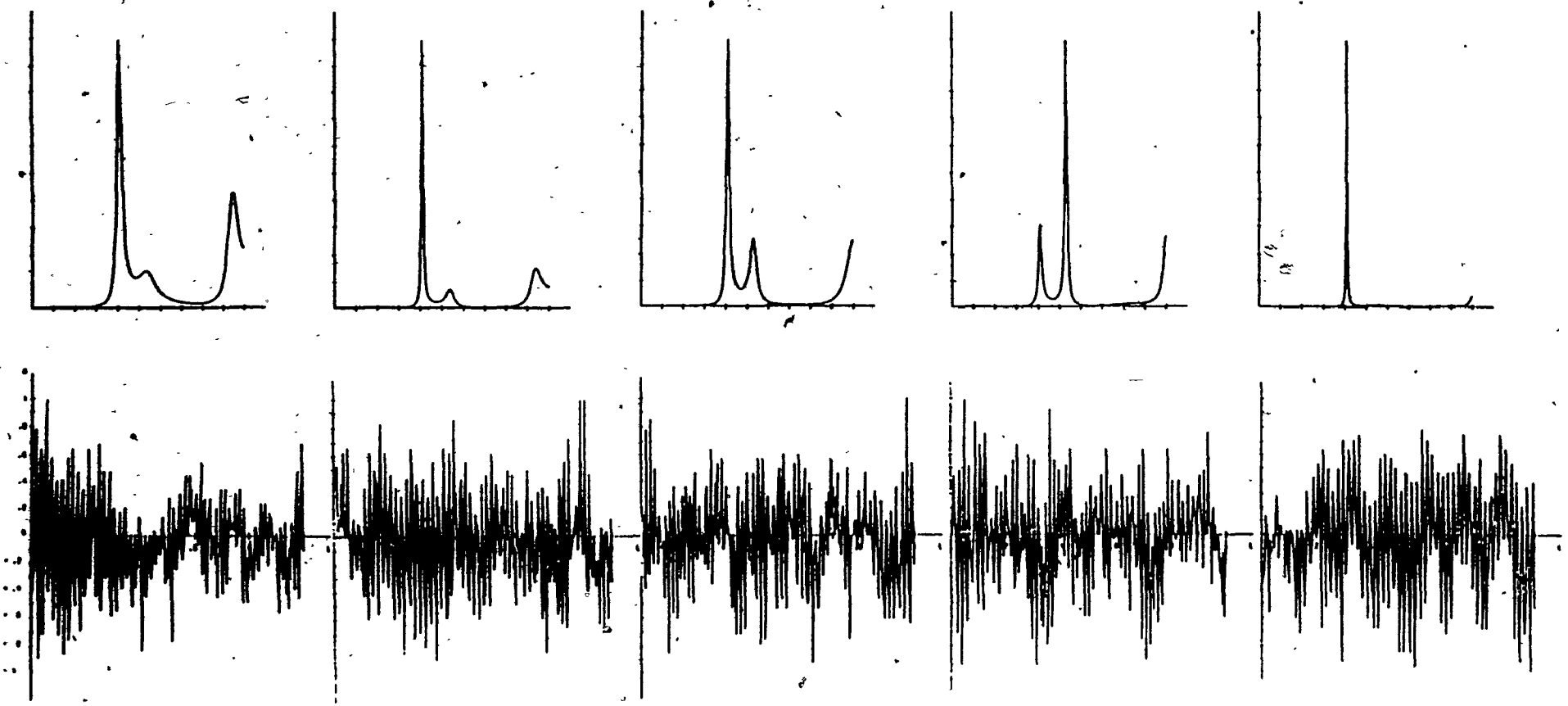


Figure 4.3

Frames 1-5

Estimated PSD (top), Sampled Signal (bottom)

6	7	8	9	10
$E_m = 0.20608$	$E_m = 0.08936$	$E_m = 0.10362$	$E_m = 0.23589$	$E_m = 0.04826$
$R = 0.002559729$	$R = 0.008200518$	$R = 0.007062208$	$R = 0.003080416$	$R = 0.016954741$

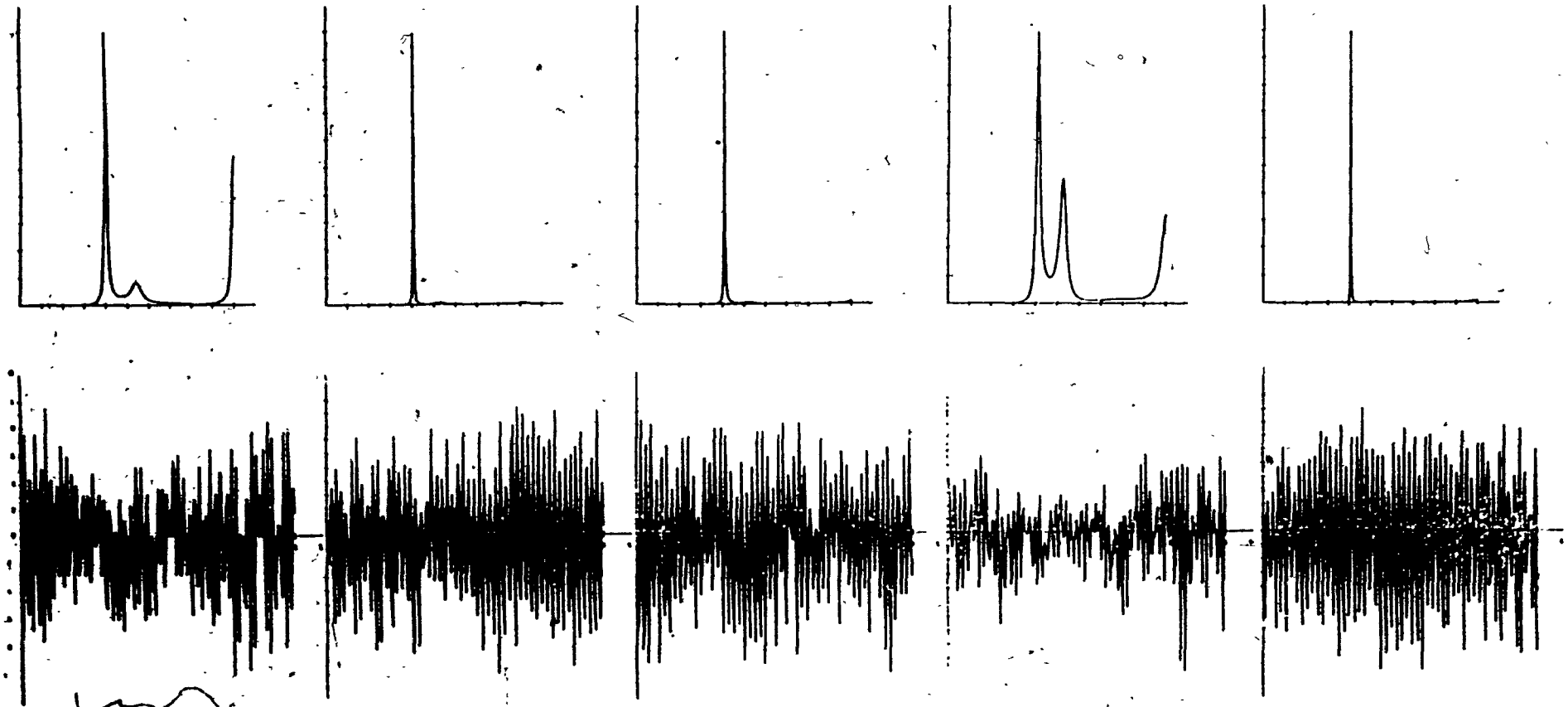


Figure 4.4

Frames 6-10

Estimated PSD (top), Sampled Signal (bottom)

11	12	13	14	15
$E_m = 0.20359$	$E_m = 0.10213$	$E_m = 0.28843$	$E_m = 0.19013$	$E_m = 0.09979$
$R = 0.004703220$	$R = 0.009339093$	$R = 0.002736282$	$R = 0.003956199$	$R = 0.004500189$

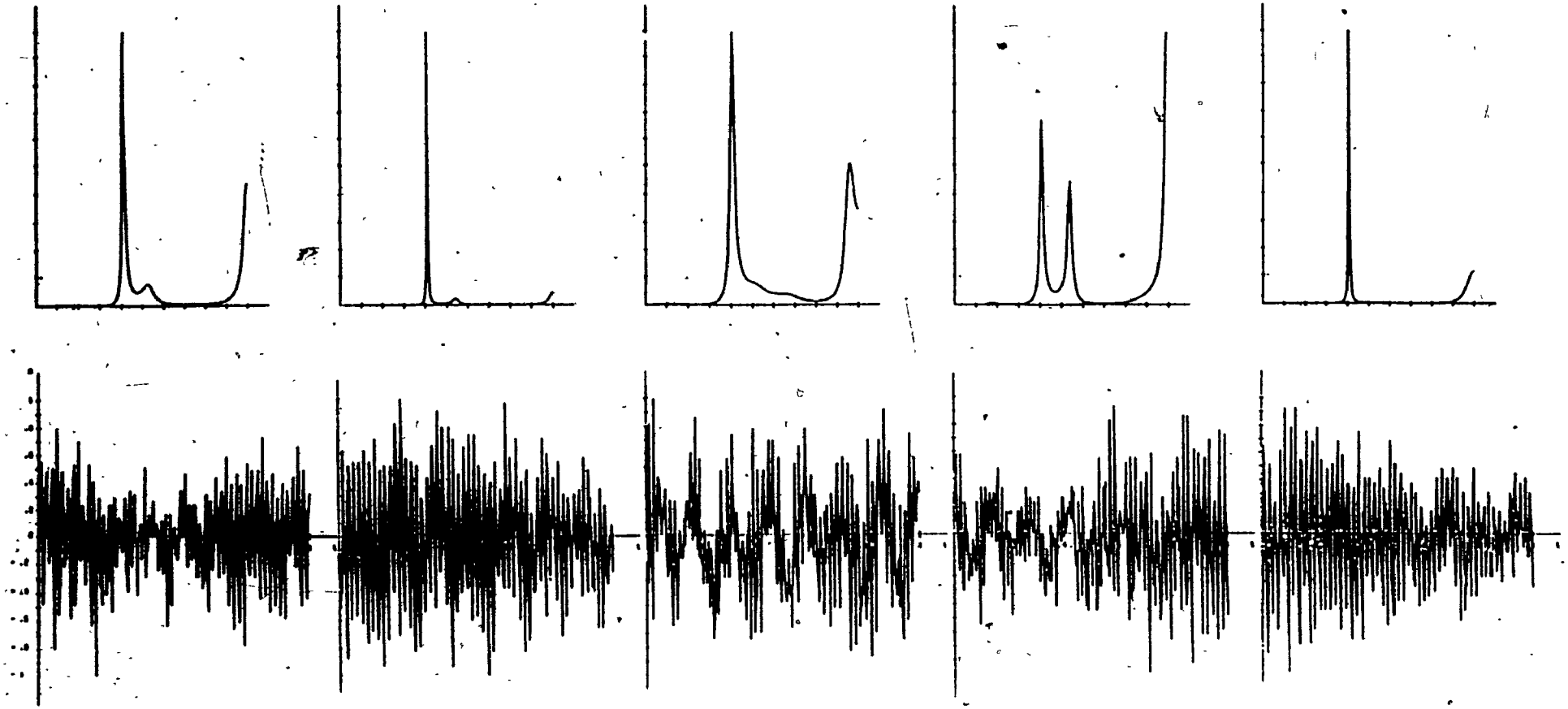


Figure 4.5

Frames 11-15

Estimated PSD (top), Sampled Signal (bottom)

16	17	18	19	20
$E_m = 0.42640$	$E_m = 0.17857$	$E_m = 0.10915$	$E_m = 0.11472$	$E_m = 0.25900$
$R = 0.000815337$	$R = 0.001193610$	$R = 0.001825614$	$R = 0.001595298$	$R = 0.000705244$

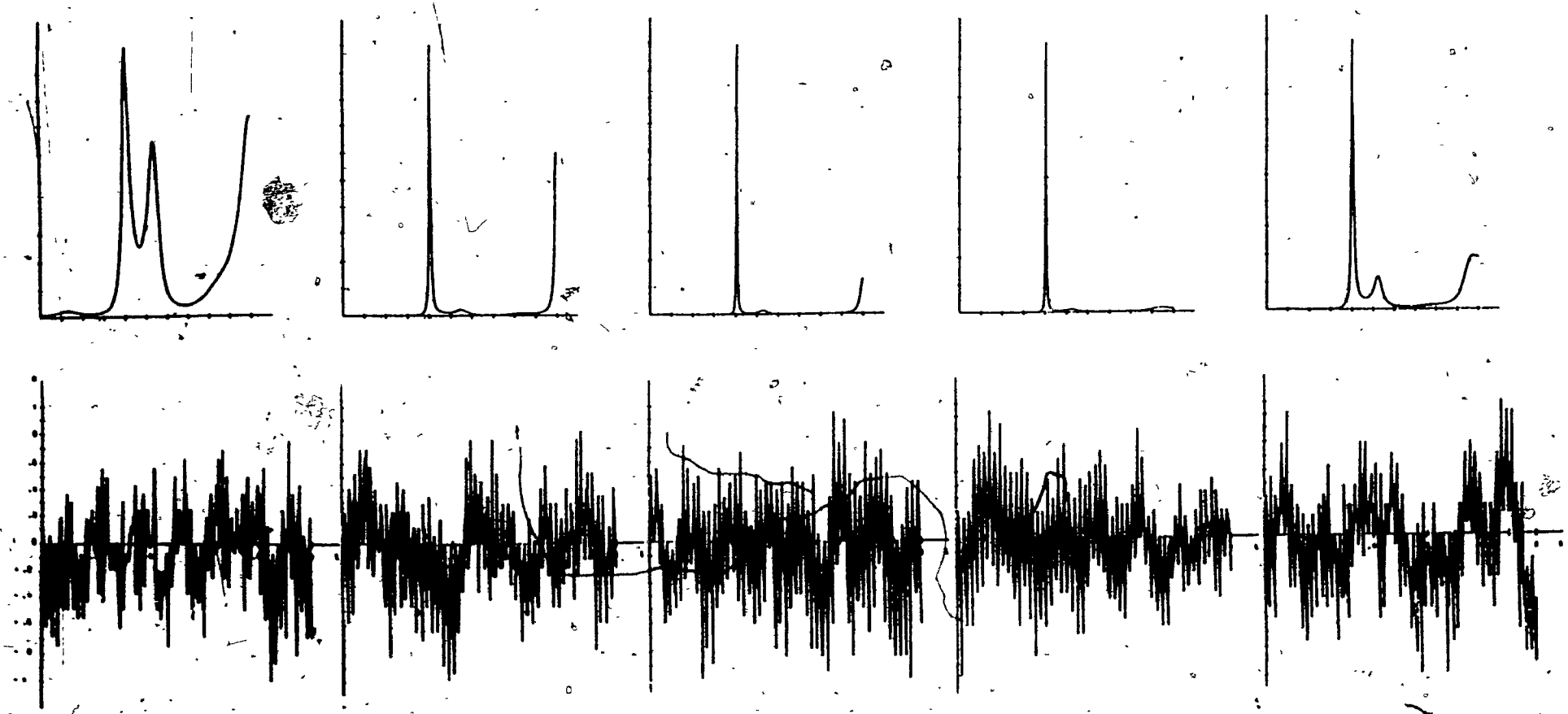


Figure 4.6

Frames 16-20

Estimated PSD (top), Sampled Signal (bottom)

Frame	Peak 1	2	3	4	
1	245 0.57	1042 0.91	1740 0.64	2390 3.74	Hz
2	502 4.17	1091 0.43	1740 2.63	2083 4.02	Hz
3	- -	870 2.92	1618 2.02	2120 1.38	Hz
4	453 3.42	944 1.22	1605 1.64	2059 1.96	Hz
5	748 1.70	1507 2.61	2034 2.01	2684 0.55	Hz
6	478 0.97	980 1.84	1642 1.52	2022 2.16	Hz
7	698 1.45	- -	1887 2.23	2745 2.36	Hz
8	453 1.23	1348 1.22	1826 1.40	2402 1.29	Hz
9	784 0.97	- -	1973 1.49	2561 2.09	Hz
10	- -	968 0.45	2194 2.08	2720 1.86	Hz
11	502 1.26	1127 9.99	- -	- -	Hz
12	502 2.21	1115 6.84	2292 0.23	- -	Hz
13	600 1.87	1103 3.23	1593 1.93	2377 0.31	Hz
14	- -	1054 3.60	- -	2586 0.52	Hz
15	588 0.76	1140 9.65	1687 0.48	2586 0.74	Hz
16	- -	1091 3.79	- -	2561 2.87	Hz
17	- -	821 2.14	1740 0.78	2672 3.05	Hz
18	184 0.61	1164 3.42	- -	2635 2.08	Hz
19	343 0.48	993 1.79	1740 0.93	2733 2.64	Hz
20	490 4.75	1078 6.51	1458 1.60	2304 0.39	Hz

Table 4.3

Peak's Location and Absolute Magnitude

for the Time-Varying Signal of Fig. 4.8

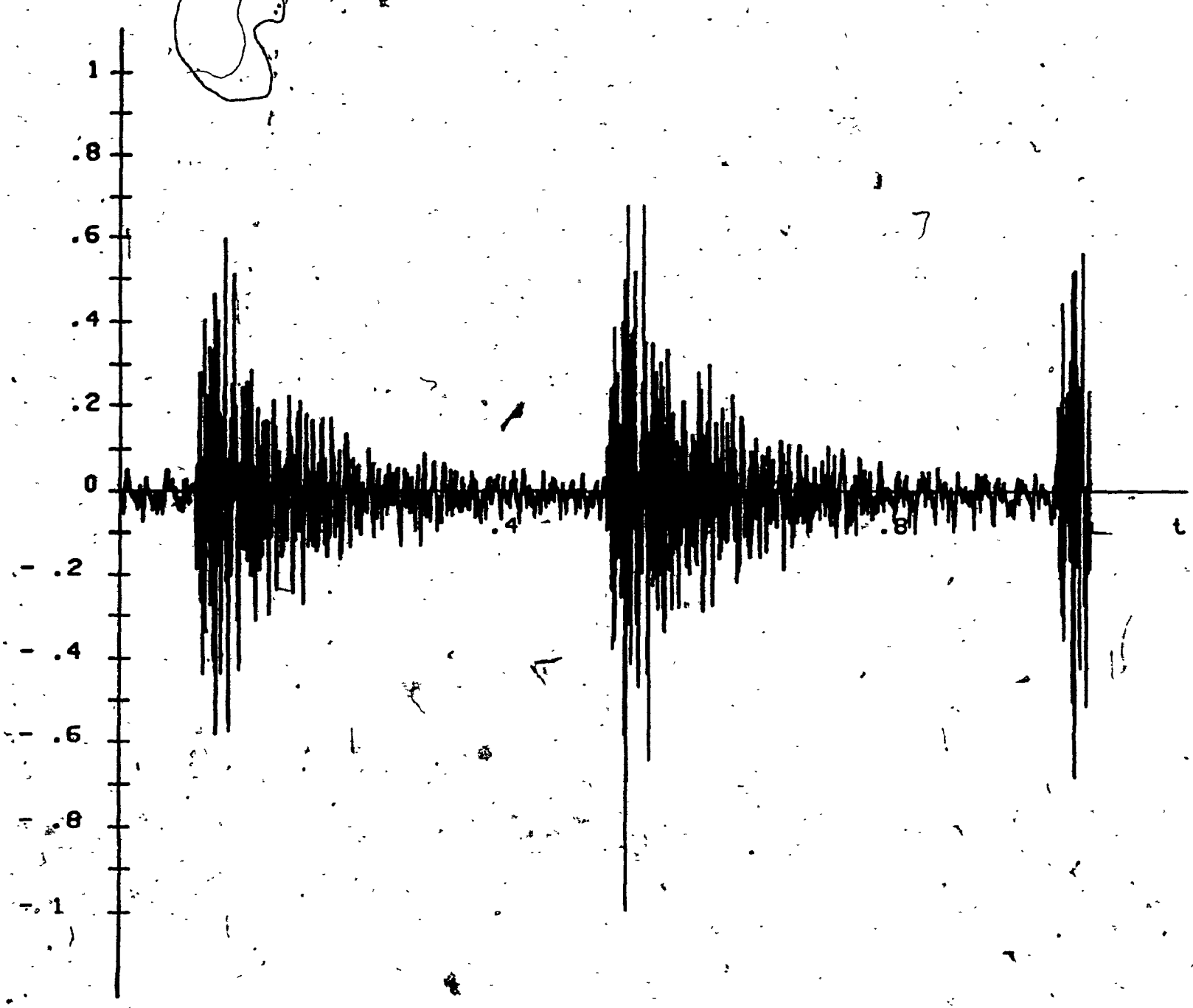


Figure 4.7

Sampled Time-varying Signal (success) (20 Frames, Total duration = 0.82 s)

1	2	3	4	5
$E_m = 0.70988$	$E_m = 0.45903$	$E_m = 0.57926$	$E_m = 0.75167$	$E_m = 0.79864$
$R = 0.000223927$	$R = 0.007839720$	$R = 0.012323762$	$R = 0.003948302$	$R = 0.001937184$

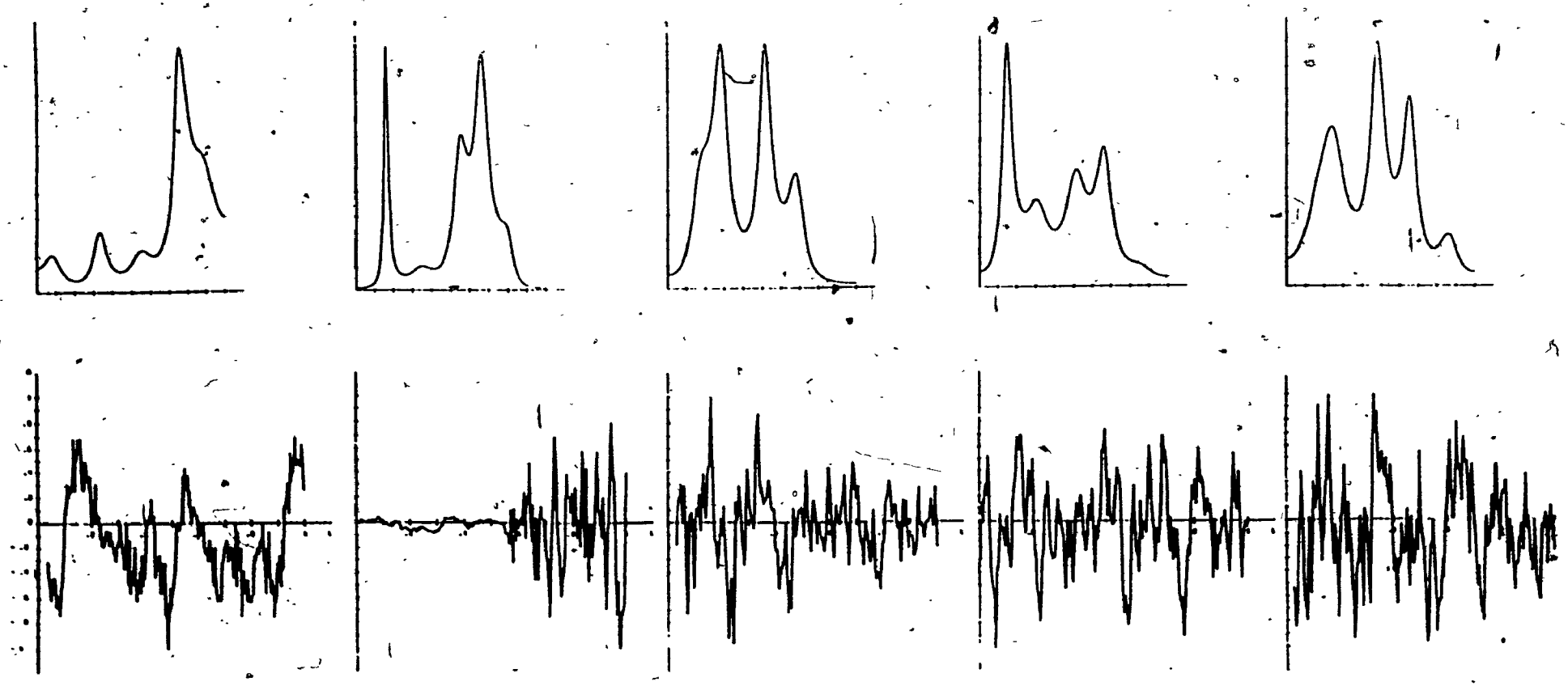


Figure 4.8

Frames 1-5

Estimated PSD (top), Sampled Signal (bottom)

$E_m = 0.85667$	$E_m = 0.89424$	$E_m = 0.89938$	$E_m = 0.85552$	$E_m = 0.78319$
$R = 0.000808079$	$R = 0.000596707$	$R = 0.000822100$	$R = 0.000995986$	$R_1 = 0.000264143$
6	7	8	9	10

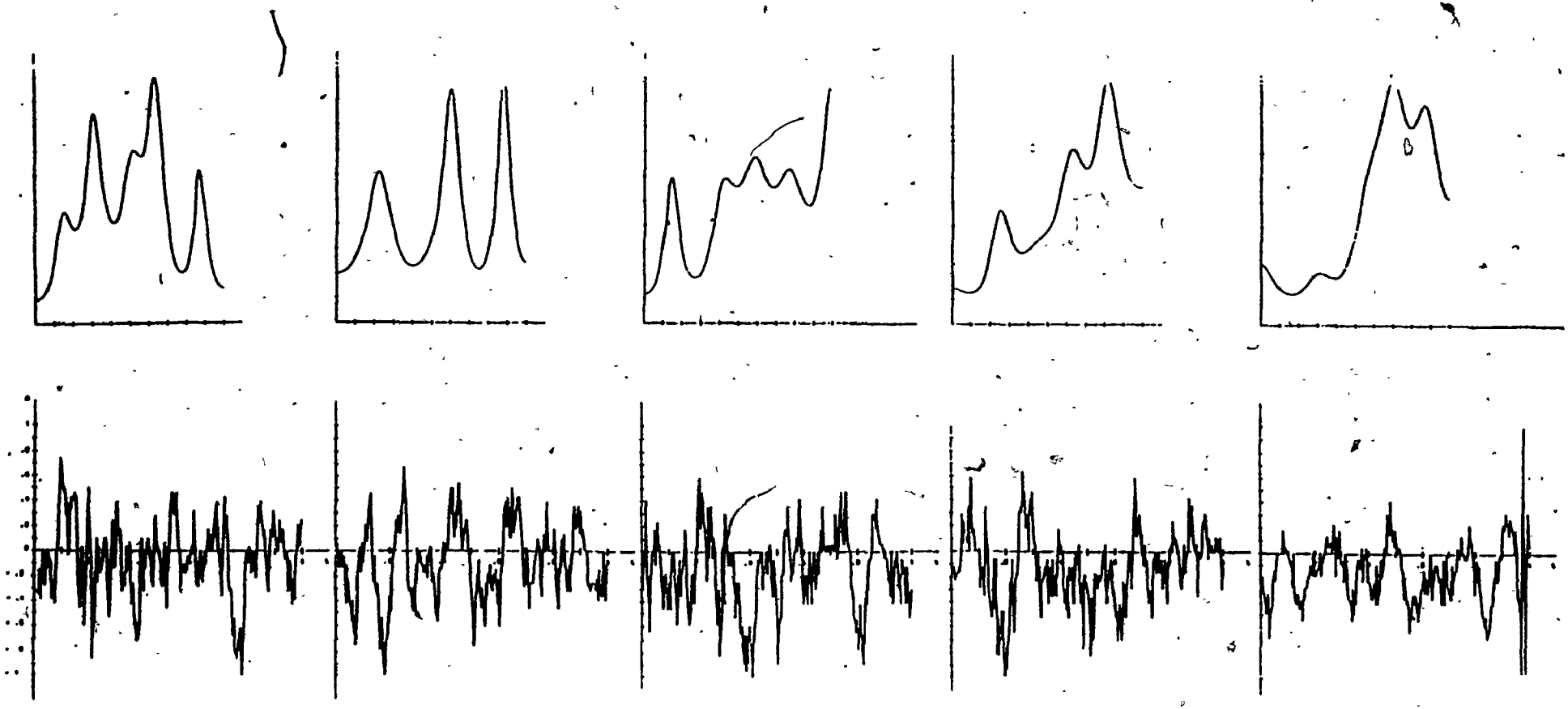


Figure 4.9

Frames 6-10

Estimated PSD (top), Sampled Signal (bottom)

$E_m = 0.28239$	$E_m = 0.43278$	$E_m = 0.39619$	$E_m = 0.63448$	$E_m = 0.62976$
$R = 0.055559485$	$R = 0.007836427$	$R = 0.009919984$	$R = 0.001587051$	$R = 0.001080808$
11	**12**	**13**	**14**	**15**

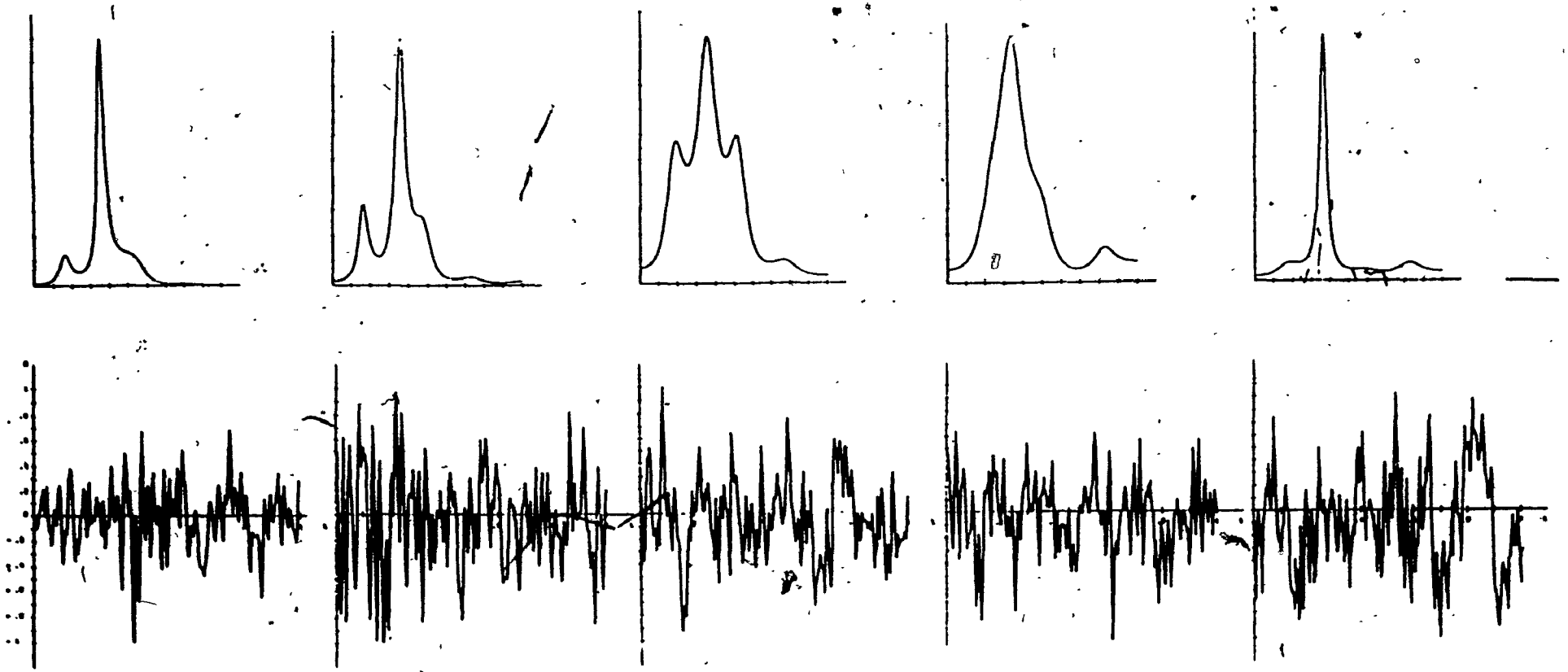


Figure 4.10

Frames 11-15

Estimated PSD (top), Sampled Signal (bottom)

$E_m = 0.76746$	$E_m = 0.79153$	$E_m = 0.81512$	$E_m = 0.80290$	$E_m = 0.42217$
$R = 0.000546671$	$R = 0.000386783$	$R = 0.000257695$	$R = 0.000224650$	$R = 0.026348110$
16	**17**	**18**	**19**	**20**

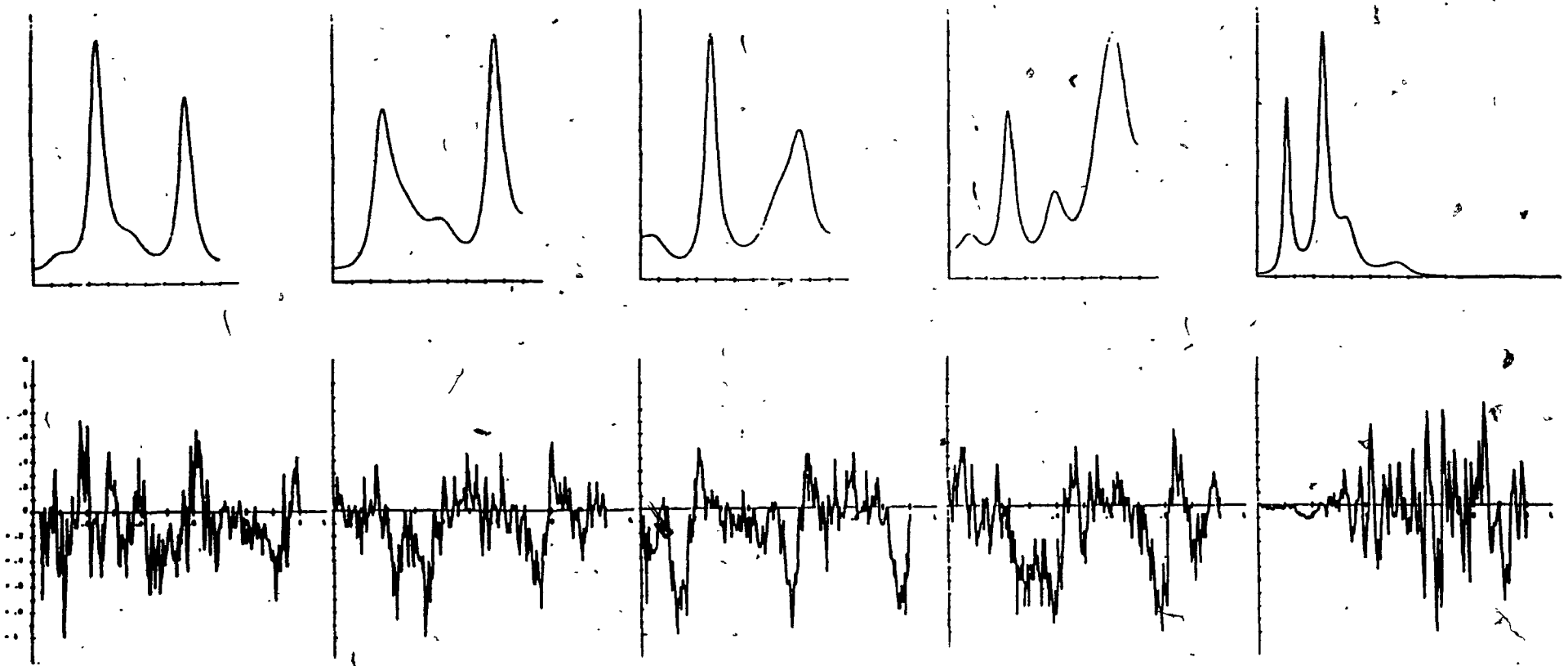


Figure 4.11

Frames 16-20

Estimated PSD (top), Sampled Signal (bottom)

Frame	Peak 1	Peak 2	Peak 3	Peak 4	
1	343 2.77	1127 1.19	1507 1.89	2132 2.30	Hz
2	355 4.76	1152 0.94	1679 13.83	2243 1.12	Hz
3	380 2.69	1336 1.19	1900 1.44	- -	Hz
4	- -	- -	1716 1.30	2635 3,07	Hz
5	392 2.98	1140 4.73	1740 0.72	2757 2.92	Hz
6	168 14.35	993 1.20	1605 1.05	2255 0.93	Hz
7	355 10.17	1140 1.04	1691 1.60	2721 1.58	Hz
8	343 2 04	1091 1.32	1728 1.11	2525 2.14	Hz
9	282 4 64	1078 0.77	1691 1.36	2721 1.32	Hz
10	380 3.90	1238 0.72	1728 0.83	2647 2.83	Hz
11	257 2.20	1029 1 88	2059 1.34	2672 1.98	Hz
12	- -	1164 4.36	1887 2.79	2512 0.81	Hz
13	404 1.22	1029 1.46	1630 3.43	2304 0.97	Hz
14	368 1.05	1042 1.24	1581 1.37	2255 2.60	Hz
15	233 8.95	1091 2.35	1740 1.39	2292 0.68	Hz
16	245 4.72	1176 1.35	1654 1.82	2426 0.99	Hz
17	233 1.25	1042 1.37	1789 1.14	2537 3.13	Hz
18	270 1.61	1103 1.19	1691 1.27	2586 3.24	Hz
19	368 0.69	1115 3.50	2059 2.51	- -	Hz
20	355 2.44	1127 2.58	2145 1.19	2806 0.64	Hz

Table 4.4

Peak's Location and Absolute Magnitude

for the Time-varying Signal of Fig. 4.13

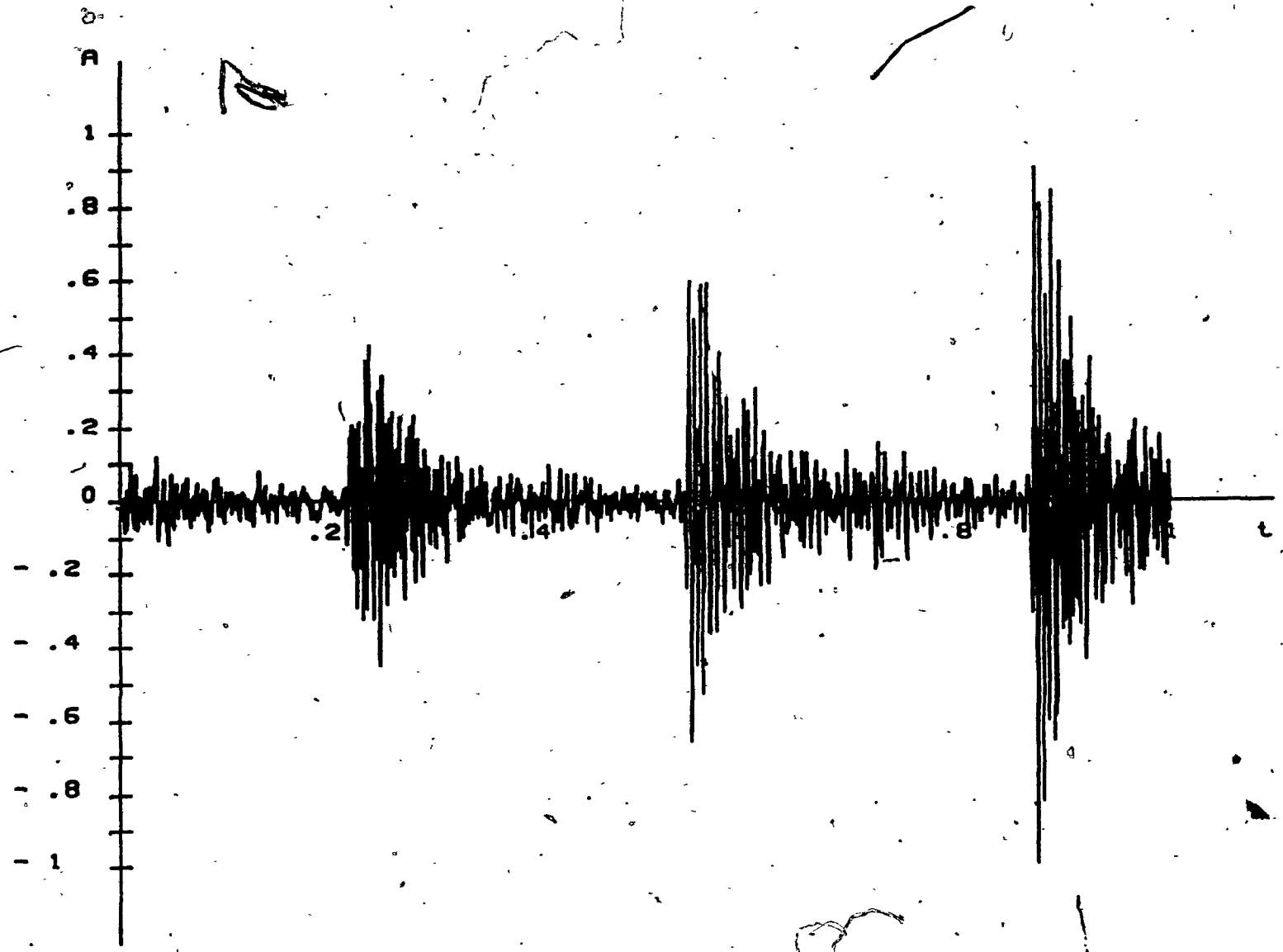


Figure 4.12

Sampled Time-varying Signal (failure) (20 Frames, Total duration = 0.82 s)

1	2	3	4	5
$E_m = 0.86472$	$E_m = 0.85247$	$E_m = 0.85674$	$E_m = 0.81342$	$E_m = 0.72554$
$R = 0.000831699$	$R = 0.000426378$	$R = 0.000346379$	$R = 0.000256564$	$R = 0.018668298$

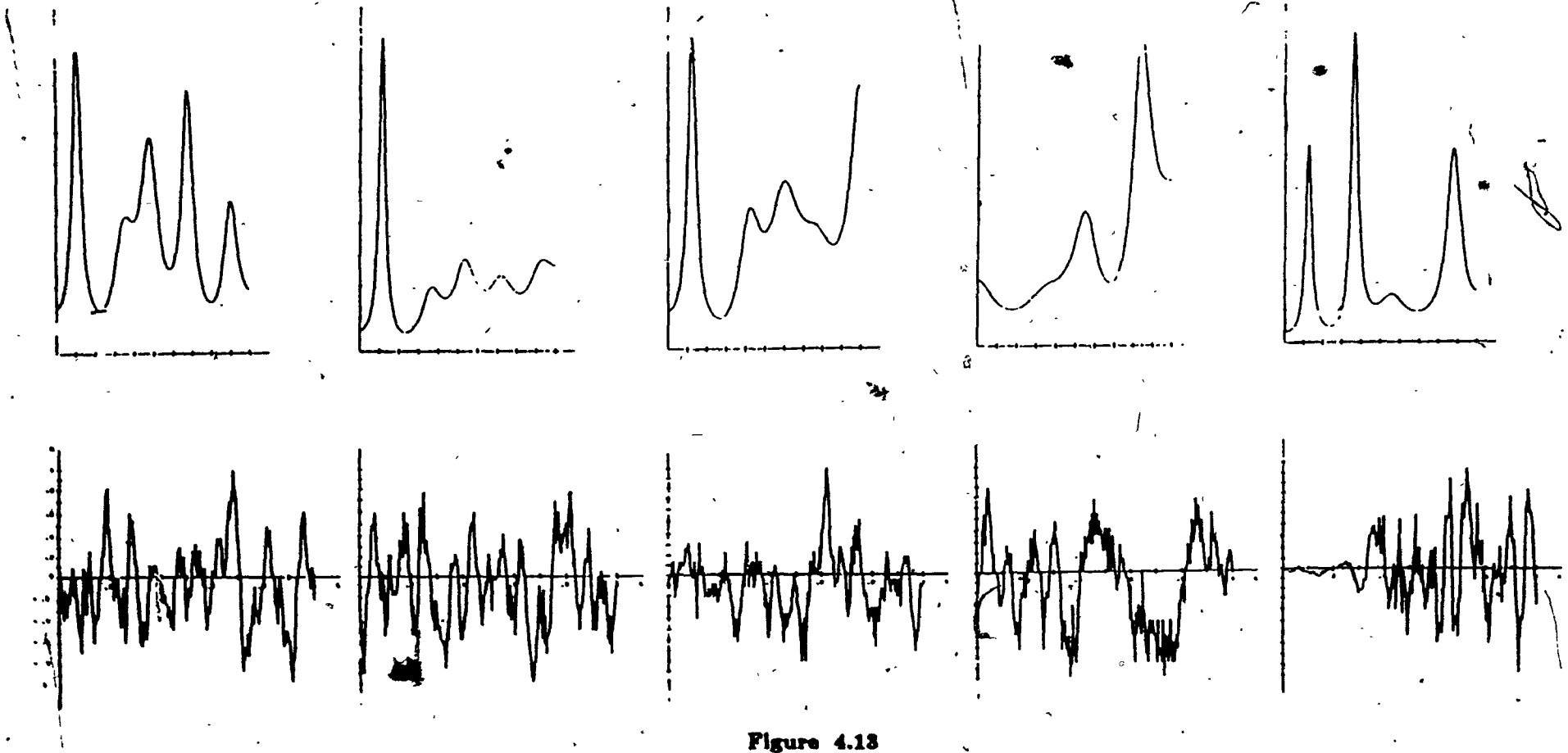


Figure 4.13

Frames 1-5

Estimated PSD (top), Sampled Signal (bottom)

$E_m = 0.69762$	$E_m = 0.74328$	$E_m = 0.90683$	$E_m = 0.85494$	$E_m = 0.79021$
$R = 0.005084246$	$R = 0.001018366$	$R = 0.000574152$	$R = 0.000494942$	$R = 0.000261732$
6	7	8	9	10

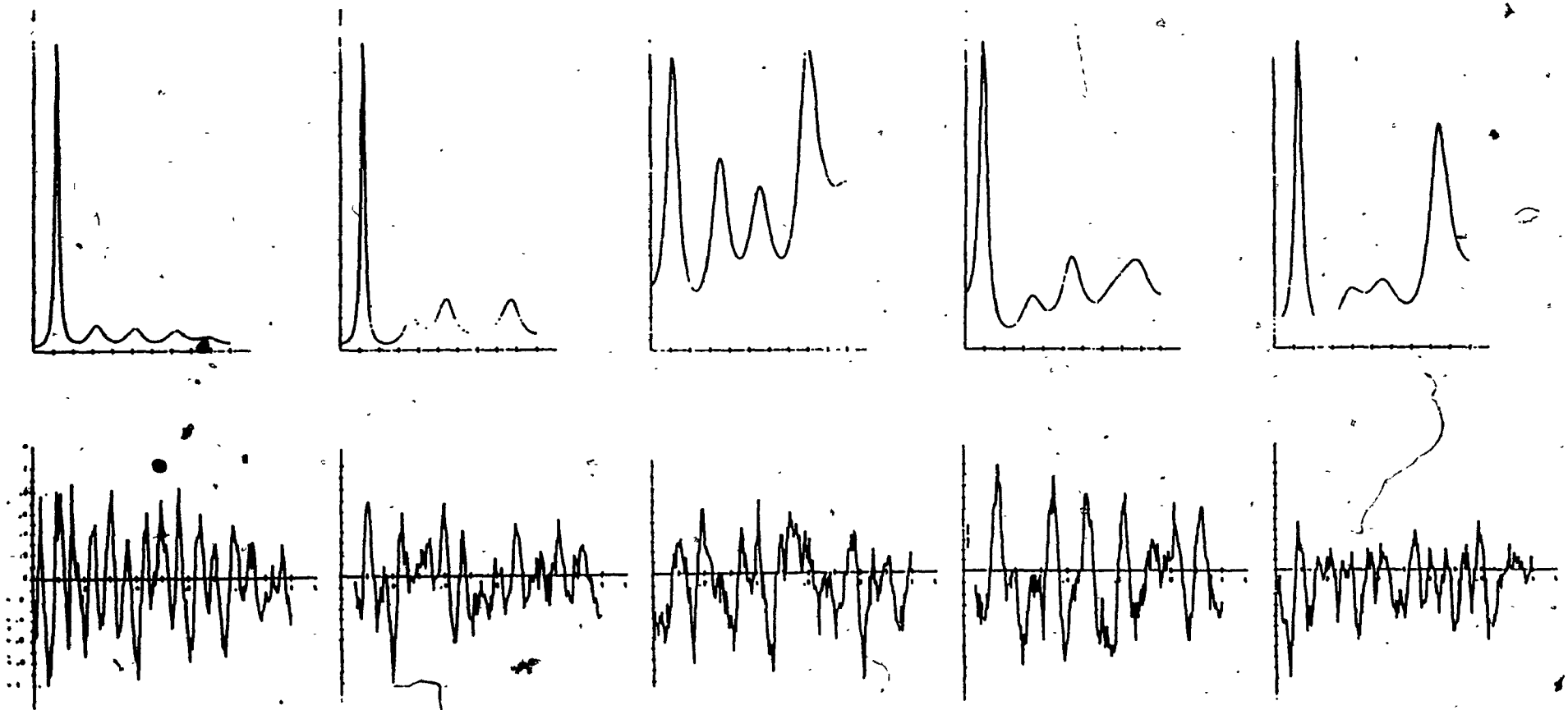


Figure 4.14

Frames 6-10

Estimated PSD (top), Sampled Signal (bottom)

11	12	13	14	15
$E_m = 0.87164$	$E_m = 0.79977$	$E_m = 0.87739$	$E_m = 0.92230$	$E_m = 0.69395$
$R = 0.002312742$	$R = 0.023327025$	$R = 0.003464268$	$R = 0.001312961$	$R = 0.001106244$

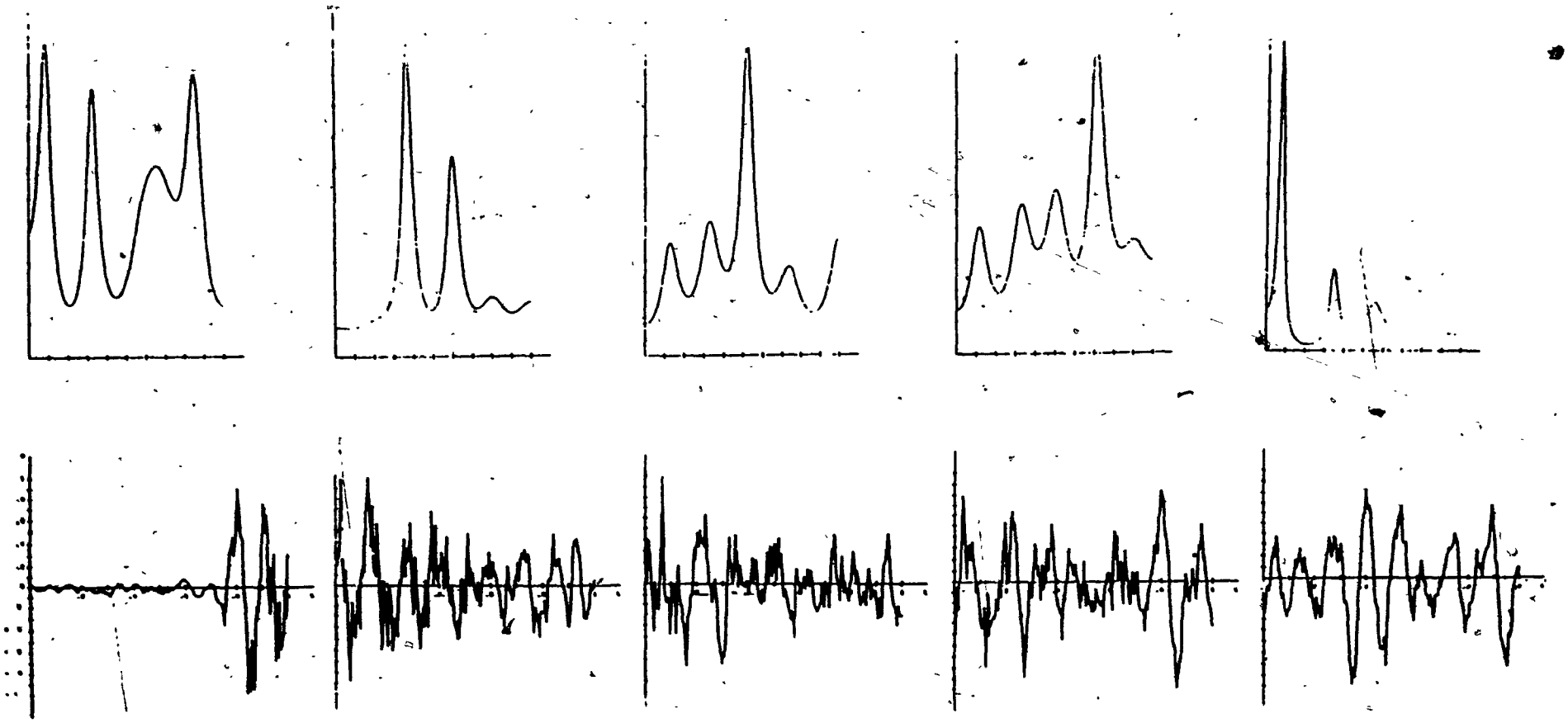


Figure 4.15

Frames 11-15

Estimated PSD (top), Sampled Signal (bottom)

16

$E_m = 0.78517$

$R = 0.000531929$

17

$E_m = 0.81708$

$R = 0.000347719$

18

$E_m = 0.86914$

$R = 0.086348712$

19

$E_m = 0.78614$

$R = 0.024840705$

20

$E_m = 0.87607$

$R = 0.005345013$

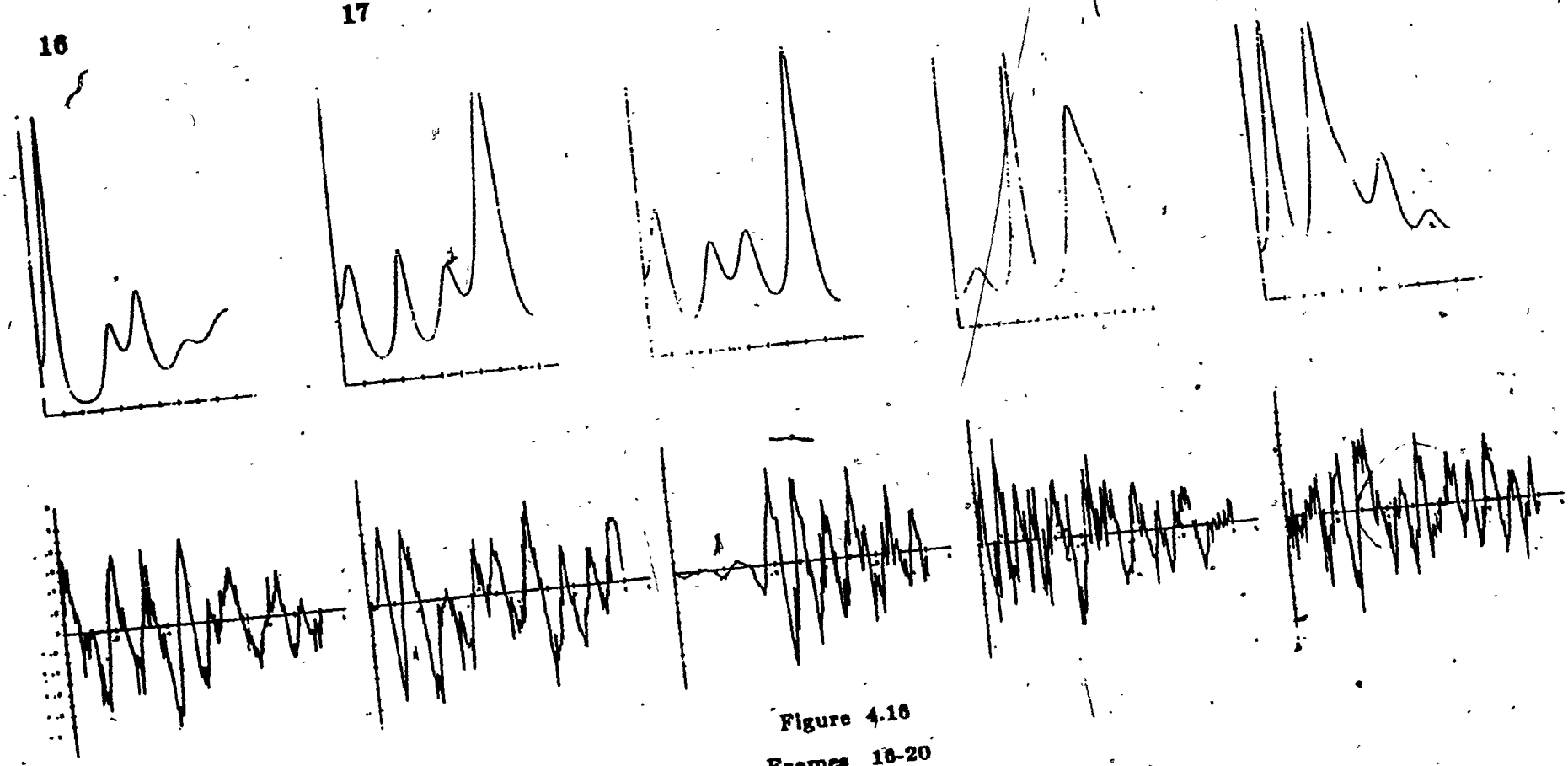


Figure 4.16

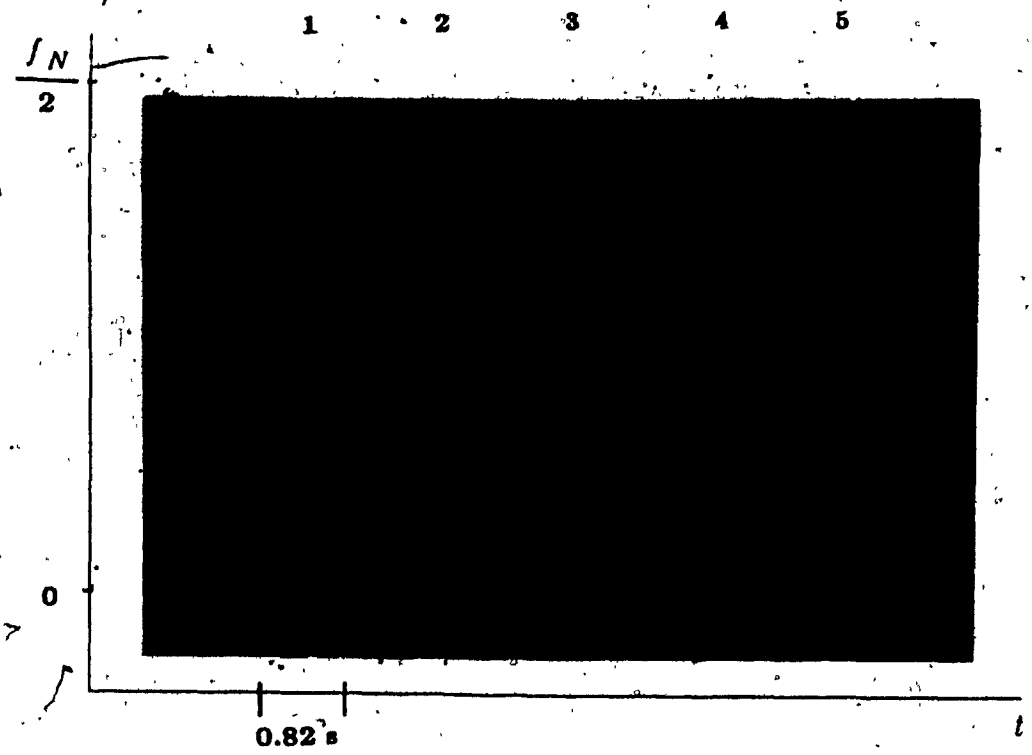
Frames 16-20

Estimated PSD (top), Sampled Signal (bottom)

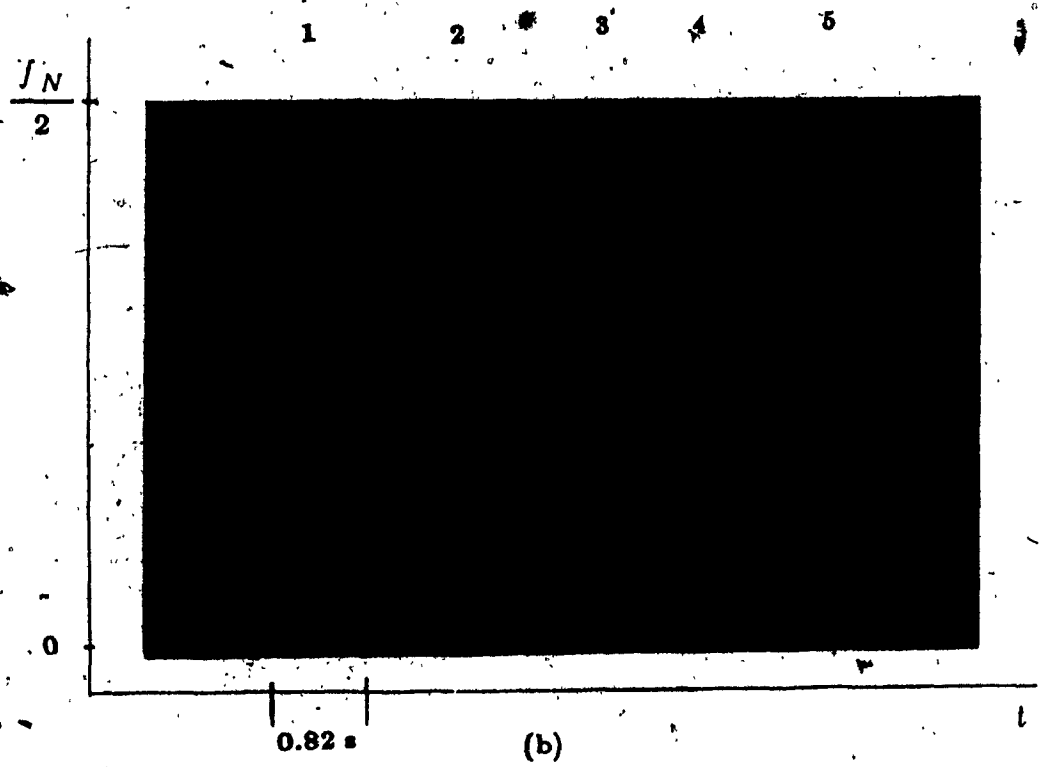
Figure no	Signal	Beginning of the Knocks (Frame no.)		
		knock 1	knock 2	knock 3
4.17a	1	21	45	71
4.17a	2	10	39	72
4.17a	3	25	57	77
4.17a	4	1	20	43
4.17a	5	2	38	76
4.17b	1	4	32	60
4.17b	2	11	48	-
4.17b	3	10	47	-
4.17b	4	1	27	66
4.17b	5	1	21	59

Table 4.5

Frame no. of the Beginning of the Excitation,
for the Sound Signals of Figures 4.17a and 4.17b



(a)



(b)

Figure 4.17

Emulation of a Vocoder - Five Knocks, (a) Wood, (b) Metal

($f_N = 6.25KHz$)

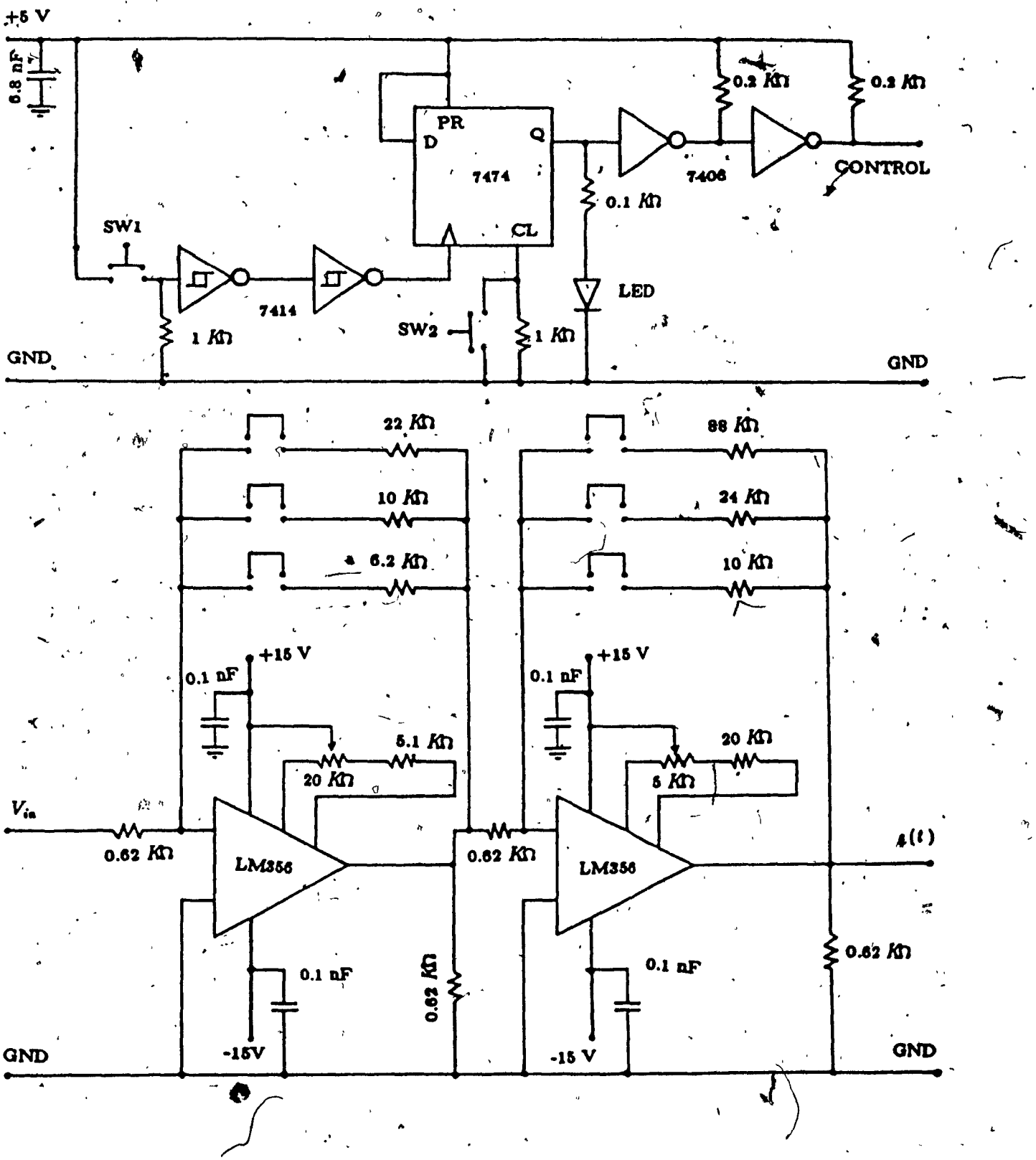


Figure 4.18

User's Control Device - Preamplifier

4.3 Discussion

Assuming that the performance of the algorithm is acceptable, the last question to be answered is: Is there, available in the market, computer hardware needed for the realization of the algorithm? Memory and computer speed requirements are of the main concern. We will examine these two requirements in detail.

Summing up the memory requirements as outlined in Chapter 2 and Chapter 3 we see that the memory required is of the order of $O(10^3)$. This includes the memory required for storage of the opcode of the algorithm. This requirement can be easily realizable with existing memory IC's. In addition to the storage data capabilities of the commercially available memory chips, their access time may be as low as 120 ns.

As we have outlined in Chapter 2 and in Chapter 3 (see also Table 3.1) the required computer operations to process a frame is in the order of $O(10^3)$. This means that, for real time response, we should employ a computer architecture capable of performing in the order of $O(10^5)$ arithmetic instructions per second (approximately 3×10^5 instructions).

Generally, the completion of the execution of any arithmetic operation by a computer architecture which employs a Floating Point Processor (FPP) involves three principal activities. The execution time of each activity is an additive component of the overall execution time of the instruction.

These activities are:

1. CPU overhead involved in handling the instruction opcode and setting up the FFP (which includes the transfer in a register of the FPP of the opcode of the numeric instruction),

2. The execution by the FPP of the instruction, and

3. The transfer of operands between the FPP and the memory or a CPU register.

Assuming for the first activity that the CPU and the FPP work in parallel and with the same clock frequency, and for the third activity that no wait states are involved (which presumes fast memory and no contention for the bus from the various elements of the computer architecture), then the overall numeric instruction execution time is approximately equal to the instruction execution time of the FPP.

A representative of a commercially available FPP is the INTEL 80827 16-bit FPP. The general purpose FPPs available in the market are of compatible speed to the INTEL 80287 (the MOTOROLA 68881 FPP is an example). It takes the 80827 ~ 150 clock cycles to execute a floating point multiplication. Assuming the clock frequency to be 5 MHz, then it can execute ~ 30,000 FLoating (point) Operations Per Second (FLOPS), which is one order of magnitude slower than the requirements. Approximately the same clock cycles are needed for the execution of a FP addition, and substantially more for the execution of a FP division. From the above we see that there is not any general purpose FPP available in the market, which we could use for real time response.

A possible alternative involves a parallel computer architecture. Taking into consideration that the overall performance of a parallel computer architecture is not proportional to the processing elements involved (because of the bus contention and of the communication among the the various processors), we conclude that a substantial number of processors is needed for the implementation of our algorithm in such an architecture. Because of that, this alternative does not seem an attractive alternative.

The last alternative in realizing the algorithm for real time results is by using specialized hardware. Observing equation 2.27 we see that for the computation of the PSD we perform autocorrelation on the linear prediction autocorrelation coefficients. From Table 3.1 we see that the load added to the system for the computation of the autocorrelation coefficients and for the computation of the PSD add up to approximately 83% of the overall load. The remaining 17% of the overall load can be processed in real time using one of the FPPs mentioned earlier. Thus a specialized processor capable of processing 83% of the overall load (or equivalently, processing one quarter of a million numeric instructions per second) is necessary.

As we have seen in section 3.3.2, preemphasis of the signal results in the increase of the stability of the system of equations 2.20. Because of that, we have concluded in the same section that the number of bits used for the representation of our data may be reduced without affecting the stability of the system of equations 2.20, and hence of our results. This conclusion was indirectly verified, in the respect that, during the experimentation, we did not have any improvement in our results when we used double-precision FP arithmetic instead of single-precision FP arithmetic.

Reasoning across the same line of thinking, we can speculate [21,23] that using integer arithmetic, in a 16-bits word length machine, taking the results in 32-bits, and rounding the results after the operations will not affect our overall results. This means a reduction of 13 bits, from the necessary 45 bits (section 3.3.1) (without taking into consideration preemphasis) down to 32 bits.

In the general purpose mini-computer used for the experimentation (PDP-11/45), we measured the time it takes to perform various arithmetic operations, using different operands. It takes approximately the same time for the execution

of a addition/subtraction of FP numbers (13.6 μs) and of integers (16-bits or 32-bits long operands) (18 μs). The same stands true for FP and for single-precision integer arithmetic (20 μs *vs* 16 μs). But it takes ~ 192 μs for the execution of one double-precision integer multiplication, which is an order of magnitude slower than the time needed for one single-precision integer or one FP multiplication. The last statement is also true for the FPP mentioned previously (INTEL 80287). Assuming that the same stands true for compatible to INTEL 80287 FPPs, we see that a specialize processor of 16-bits word length (results in 32-bits), and fast so as to process the 83% of the overall load, is needed.

Such a processor available in the market is the Texas Instrument TMS32010. This processor is capable of 16x16 bits integer multiplication (result in 32 bits) in one machine cycle. For a clock frequency of 5 MHz it takes 200 ns for the execution of a numeric instruction. Using memory whose access time is of the same order as the processor's machine cycle, this processor can process the signal in real time.

From the above we see that we can implement the algorithm in real time by using fast memory and specialized hardware (both available in the market).

CHAPTER 5

SUMMARY

In this thesis we have introduced the computer requirements for the construction of a digital hearing aid device. This device should be able to make aware the hearing impaired persons of hazardous environmental conditions by recognizing sounds such as a fire alarm, a siren, etc., and also to improve their communication by recognizing sounds such as a telephone ring, a knock at a door, etc.

For the completion of the present thesis, in addition to the theoretical investigation of the overall problem, the following tasks were completed:

1. Design of the software (algorithm) to be incorporated in the device,
2. Design and building of the hardware needed for testing the software in a general purpose mini-computer,
3. Testing of the algorithm,
4. Analysis of the results, and
5. Outline of the computer memory and speed required for the realization of the device.

The algorithm which we developed and we tested was successful in recognizing time-invariant signals, and it was partly successful in recognizing time-varying signals. The overall success rate is in the range of 80% for all signals tested.

Using image processing techniques the evolution in time of the frequency components of the signals was observed. The free and the forced oscillations of

the sound generators give two well defined and distinguishable ranges of frequencies. These ranges depend on the material the sound generator is composed of, and they may be used to identify the sound generator.

We plan to use the results taken by using image processing techniques in the future, when the building stage of the digital hearing aid device will take place, so that to further improve the success rate of the algorithm.

BIBLIOGRAPHY

1. Akaike, H., "A New Look at the Statistical Model Identification," *IEEE Trans. Automat. Control*, vol. AC-19, pp. 716-723, Dec. 1974.
2. Aldefeld, B., Rabiner L.R., Rosenberg A.E., and Wilpon J.G., "Automated Directory Listing Retrieval Sustem Based on Isolated Word Recognition," *Proc. IEEE*, vol. 68, no. 11, pp. 1364-1379, Nov. 1980.
3. Antoniou, A., *Digital Filters: Analysis and Design*, McGraw-Hill Book Company, 1979.
4. Atal, B.S. and Hanauer S.L., "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *J. Acoust. Soc. Am.*, vol. 50, pp. 637-655, Aug. 1971.
5. Bartlett, M.S., *An Introduction to Stochastic Processes, with Special Reference to Methods and Applications*, Cambridge at the University Press, 1962.
6. Benjamin, J.R. and Cornell C.A., *Probability, Statistics and Decision for Civil Engineers*, McGraw-Hill Book Company, 1970.
7. Bode, H.W. and Shannon C.E., "A Simplified Derivation of Linear Least Square Smoothing and Prediction Theory," *Proceedings of the I.R.E.*, vol. 38, pp. 417-425, 1950.
8. Brantingham, G.L., "Impact of Integrated Circuits Technology on Speech Processing Methodologies," in *Computer Analysis and Perception*, ed. C.Y. Suen, and R. DeMori, vol. II, Auditory Signals, pp. 143-162, CRC Press, Inc., 1982.
9. Broad, D.J. and Shoup J.E., "Concepts for Acoustic Phonetic Recognition," in *Speech Recognition, Invited Papers Presented at the 1974 IEEE Symposium*, ed. D.R. Reddy, pp. 243-274, Academic Press, 1975.
10. Carterette, E.C., "Some Historical Notes on Research in Hearing," in *Handbook of Perception*, ed. E.C. Carterette and M.P. Friedman, vol. IV, Hearing , pp. 3-39, Academic Press, 1978.
11. Christiansen, R.W. and Rushforth C.K., "Detecting and Locating Key Words in Continuous Speech Using Linear Predictive Coding," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-25, pp. 361-367, Oct. 1977.
12. Coker, M.J. and Boll S.F., "An Improved Isolation Word Recognition System Based upon the Linear Prediction Residual," *IEEE, ICASSP*, pp. 206-209, 1976.
13. Dautrich, B.A., Rabiner L.R., and Martin T.B., "The Effects of Selected Signal Processing Techniques on the Performance of a Filter-Bank-Based Isolated Word Recognizer," *The Bell System Technical Journal*, vol. 62, No. 5, May-June 1983.
14. Del'Aune, W., Lewis C., Dolan M., Grimmelsman T., and Needham W., "Two Sensory Aids Having Profound Effects on the Blind," *IEEE, ICASSP*, pp. 606-610, 1976.
15. Drucker, H., Woodworth C., and Lawrence C., "Microprocessor-Based Signal Processing for the Perceptually Deaf," *IEEE, ICASSP*, pp. 255-256, 1977.

16. Duker, S., *Time-compressed Speech: An Anthology and Bibliography in Three Volumes*, vol. II, The Scarecrow Press, Inc., 1974.

17. Durbin, J., "The Fitting of Time-Series Models," *Inst. Int. De Stat. Revue de*, vol. 28, no. 3, pp. 233-243, 1960.

18. Fu, K.S., "Special Computer Architecture for Pattern Recognition and Image Processing," *AFIPS, Proceedings of the NCC*, vol. 47, pp. 1003-1013, 1978.

19. Gayford, M.L., *STM Monograph, Electroacoustics, Microphones, Earphones and Loudspeakers*, American Elsevier Publishing Company, Inc, 1971.

20. Georgiou, V.J., *A Parallel Pipeline Computer Architecture for Speech Processing*, UMI Research Press, 1984.

21. Gray, A.H. and Markel J.D., "A Spectral-Flatness Measure for Studying the Autocorrelation Method of Linear Prediction of Speech Analysis," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-22, pp. 207-217, June 1974.

22. Gray, A.H. and Markel J.D., *Linear Prediction of Speech*, 1976.

23. Gray, A.H. and Markel J.D., "Quantization and Bit Allocation in Speech Processing," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 459-473, Dec. 1976.

24. Itakura, F. and Saito S., "On the Optimum Quantization of Feature Parameters in the PARCOR Speech Synthesizer," *IEEE, 1972 Conf. on Speech Commun. and Process.*, pp. 434-437, 1972.

25. Itakura, F., "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-23, pp. 67-72, Feb. 1975.

26. Jenkins, G.M. and Watts D.G., *Spectral Analysis and its Applications*, Holden-Day, 1968.

27. Kolmogorov, A.N. and Fomin S.V., *Measure, Lebesgue Integrals, and Hilbert Space*, Academic Press, 1961.

28. Lesser, V.R., "Parallel Processing in Speech Understanding Systems: A Survey of Design Problems," in *Speech Recognition, Invited Papers Presented at the 1974 IEEE Symposium*, ed. D.R. Reddy, pp. 481-499, Academic Press, 1975.

29. Makhoul, J., "Linear Prediction in Automatic Speech Recognition," in *Speech Recognition, Invited Papers Presented at the 1974 IEEE Symposium*, ed. D.R. Reddy, pp. 183-220, Academic Press, 1975.

30. Makhoul, J., "Linear Prediction: A Tutorial Review," *Proc. IEEE*, vol. 63, pp. 561-580, Apr. 1975.

31. Markel, J.D., "The SIFT Algorithm for Fundamental Frequency Estimation," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 367-377, Dec. 1972.

32. Markel, J.D., "Digital Inverse Filtering - A New Tool for Formant Trajectory Estimation," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 129-137, June 1972.

33. McCandless, S.S., "An Algorithm for Automatic Formant Extraction Using Linear Prediction Spectra," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-22, pp. 135-141, April 1974.

34. Mermelstein, P., "Computer Recognition of Continuous Speech," in *Computer Analysis and Perception*, ed. C.Y. Suen, and R. DeMori, vol. II, Auditory Signals, pp. 81-100, CRC Press, Inc., 1982.

35. Noll, A.J., "Cepstrum Pitch Determination," *J. Acoust. Soc. Am.*, vol. 41, pp. 293-309, Feb. 1967.

36. Nudd, G.R., "Image Understanding Architectures," *AFIPS, Proceedings of the NCC*, vol. 49, pp. 377-390, 1980.

37. Papoulis, A., *Signal Analysis*, McGraw-Hill Book Company, 1977.

38. Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Book Company, 1984.

39. Parzen, E., *Stochastic Processes*, Holden-Day, Inc., 1964.

40. Pierce, A.D., *Acoustics, An Introduction to its Physical Principles and Applications*, McGraw-Hill Book Company, 1981.

41. Pierce, J.R., "Whither Speech Recognition," *J. Acoust. Soc. Am.*, vol. 46, pp. 1049-1059, 1969.

42. Plomp, R., *Aspects of Tone Sensation: a Psychophysical Study*, Academic Press, 1976.

43. Rabiner, L.R. and Gold B., *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., 1975.

44. Rabiner, L.R., Cheng M.J., Rosenberg A.E., and McGonegal C.A., "A Comparative Performance Study of Several Pitch Detection Algorithms," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 399-418, Oct. 1976.

45. Rabiner, R.L., Rosenberg A.E., and Levinson S.E., "Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-26, pp. 575-582, Dec. 1978.

46. Ruske, G., "Auditory Perception and its Application to Computer Analysis of Speech," in *Computer Analysis and Perception*, ed. C.Y. Suen, and R. DeMori, vol. II, Auditory Signals, pp. 1-42, CRC Press, Inc., 1982.

47. Sakoe, H. and Chiba S., "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-26, pp. 43-49, Feb. 1978.

48. Sambur, M.R., "Speaker Recognition Using Orthogonal Linear Prediction," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 283-289, Aug. 1976.

49. Schafer, R.W. and Rabiner L.R., "Parametric Representation of Speech," in *Speech Recognition, Invited Papers at the 1974 IEEE Symposium*, ed. D.R. Reddy, pp. 99-150 , Academic Press, 1975.

50. Schafer, R.W. and Rabiner L.R., "Digital Representations of Speech Signals," *Proc. IEEE*, vol. 63, pp. 662-677, Apr. 1975.

51. Schubert, E.D., "History of Research on Hearing," in *Handbook of Perception*, ed. E.C. Carterette and M.P. Friedman, vol. IV, Hearing, pp. 42-80, Academic Press, 1978.

52. Stremler, F.G., *Introduction to Communication Systems*, Addison-Wesley Publishing Co., 1977.

53. Viswanathan, R. and Makhoul J., "Quantization Properties of Transmission Parameters in Linear Predictive Systems," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-23, pp. 309-321, June 1975.

54. White, G.M., "Speech Recognition : A Tutorial Overview," *Computer*, vol. 9, pp. 40-53, May 1976.

55. Whittle, P., *Prediction and Regulation by Linear Least-Square Methods*, The English Universities Press, Ltd, 1963.

56. Wiener, N., *Cybernetics*, The MIT Press, 1962.

57. Willsky, A.S., *DSP and Control and Estimation Theory*, The MIT Press, 1979.

58. Wohlford, R.E., Wrench E.H., and Landell B.P., "A Comparison of Four Techniques for Automatic Speaker Recognition," *IEEE, ICASSP Proceedings*, pp. 908-911, 1980.

APPENDICES

APPENDIX A.1

Recursive Solution of the System $\mathbf{R}_{nn}\,\mathbf{a}_n = \mathbf{r}_n$

[Durbin's Algorithm]

In here we present a detailed solution of the system of equations $\mathbf{R}_{nn}\,\mathbf{a}_n = \mathbf{r}_n$ and we demonstrate the first steps of the recursion as they were implemented in our algorithm.

In solving the given system of n-equations for a_i $(i = 1,2,...,n)$ unknowns, we assume that the input data s(j) $(j = 1,2,...,N)$ are linearly independent. In this case the correlation matrix matrix $\mathbf{R}_{nn}$ is positive definite. For clarity we have change the notation of the elements of this matrix from R(x) (which appears in the text) to R_x.

The Toeplitz matrix $\mathbf{R}_{nn}$ and the column vectors $\mathbf{a}_n$ and $\mathbf{r}_n$, are defined as follows:

$$\mathbf{R}_{nn} = \begin{bmatrix} R_0 & R_1 & \cdots & R_{n-2} & R_{n-1} \\ R_1 & R_0 & \cdots & R_{n-3} & R_{n-2} \\ R_2 & R_1 & \cdots & R_{n-4} & R_{n-3} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ R_{n-1} & R_{n-2} & \cdots & R_1 & R_0 \end{bmatrix} \tag{A.1.1}$$

$$\mathbf{a}_n = [a_1 \ldots a_n]^t \tag{A.1.2}$$

$$\mathbf{r}_n = [r_1 \ldots r_n]^t \tag{A.1.3}$$

We also define the column vectors $\mathbf{a}_n^m$, $\mathbf{b}_n^m$, and $\mathbf{v}_n$ as:

$$\mathbf{a}_n^m = [\, a_1^m \ldots a_n^m \,]^t \tag{A.1.4}$$

$$\mathbf{b}_n^m = [\, a_n^m \ldots a_1^m \,]^t \tag{A.1.5}$$

$$\mathbf{v}_n = [\, R_n \cdots R_1 \,]^t \qquad \text{(A.1.6)}$$

Superscripts define the order of the system (i.e. a_i^j means the i^{th} autocorrelation coefficient of a system of j equations, $i \leq j$). Clearly the solutions to the system are given by the a_i^n, $(1 \leq i \leq n)$.

Using equation (A.1.6) we can rewrite equation (A.1.1) as follow:

$$\mathbf{R}_{nn} = \begin{bmatrix} \mathbf{R}_{(n-1)(n-1)} & \mathbf{v}_{n-1} \\ \mathbf{v}_{n-1}^t & R_0 \end{bmatrix}$$

Using the above equation in our homogeneous system and by partitioning we take :

$$\left[\begin{array}{c|c} \mathbf{R}_{(n-1)(n-1)} & \mathbf{v}_{n-1} \\ \hline \mathbf{v}_{n-1}^t & R_0 \end{array}\right] \left[\begin{array}{c} a_1^n \\ a_{n-1}^n \\ \hline a_n^n \end{array}\right] = \left[\begin{array}{c} R_1 \\ R_{n-1} \\ \hline R_n \end{array}\right]$$

Carrying the multiplication on the partitioned matrices we take the following simultaneous equations :

$$\mathbf{R}_{(n-1)(n-1)}\, \mathbf{a}_{n-1}^n + \mathbf{v}_{n-1}\, a_n^n = \mathbf{r}_{n-1} \qquad \text{(A.1.7a)}$$

$$\mathbf{v}_{n-1}^t\, \mathbf{a}_{n-1}^n + R_0\, a_n^n = R_n \qquad \text{(A.1.7b)}$$

The relation $\mathbf{R}_{nn}\, \mathbf{a}_n^n = \mathbf{r}_n$ holds true for any n $(n > 0)$, and thus it also holds true for $n = n-1$. Assuming the order of the system to be at least one, we have :

$$\mathbf{R}_{(n-1)(n-1)}\, \mathbf{a}_{n-1}^{n-1} = \mathbf{r}_{n-1} \qquad \text{(A.1.8)}$$

or

$$\mathbf{a}_{n-1}^{n-1} = \mathbf{R}_{(n-1)(n-1)}^{-1}\, \mathbf{r}_{n-1} \qquad \text{(A.1.9)}$$

Expanding equation (A.1.8) we take :

$$\begin{bmatrix} R_0 & R_1 & \dots & R_{n-2} \\ R_1 & R_0 & \dots & R_{n-3} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ R_{n-2} & R_{n-3} & \cdot & R_0 \end{bmatrix} \begin{bmatrix} a_1^{n-1} \\ \cdot \\ \cdot \\ \cdot \\ a_{n-1}^{n-1} \end{bmatrix} = \begin{bmatrix} R_1 \\ \cdot \\ \cdot \\ \cdot \\ R_{n-1} \end{bmatrix}$$

which is equivalent (by rearranging rows and columns) to :

$$\begin{bmatrix} R_0 & R_1 & \dots & R_{n-2} \\ R_1 & R_0 & \dots & R_{n-3} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ R_{n-2} & R_{n-3} & \cdot & R_0 \end{bmatrix} \begin{bmatrix} a_1^{n-1} \\ \cdot \\ \cdot \\ \cdot \\ a_{n-1}^{n-1} \end{bmatrix} = \begin{bmatrix} R_{n-1} \\ \cdot \\ \cdot \\ \cdot \\ R_1 \end{bmatrix}$$

or in matrix notation

$$\mathbf{R}_{(n-1)(n-1)}\, \mathbf{b}_{n-1}^{n-1} = \mathbf{v}_{n-1}$$

or

$$\mathbf{b}_{n-1}^{n-1} = \mathbf{R}_{(n-1)(n-1)}^{-1}\, \mathbf{v}_{n-1} \qquad \text{(A.1.10)}$$

Multiplying equation (A.1.7a) by $\mathbf{R}_{(n-1)(n-1)}^{-1}$, using equations (A.1.9) and (A.1.10), and rearranging we have from equations (A.1.7a) and (A.1.7b) :

$$\mathbf{a}_{n-1}^{n} = \mathbf{a}_{n-1}^{n-1} - a_n^n\, \mathbf{b}_{n-1}^{n-1} \qquad \text{(A.1.11a)}$$

$$\mathbf{v}_{n-1}^{t}\, \mathbf{a}_{n-1}^{n} + R_0 a_n^n = R_n \qquad \text{(A.1.11b)}$$

Multiplying equation (A.1.11a) by $\mathbf{v}_{n-1}^{t}$ and inserting equation (A.1.11b) we take:

$$\mathbf{a}_{n-1}^{n} = \mathbf{a}_{n-1}^{n-1} - a_n^n\, \mathbf{b}_{n-1}^{n-1} \qquad \text{(A.1.12a)}$$

$$a_n^n \left[R_0 - \mathbf{v}_{n-1}^t \mathbf{b}_{n-1}^{n-1} \right] = R_n - \mathbf{v}_{n-1}^t \mathbf{a}_{n-1}^{n-1} \qquad \text{(A.1.12b)}$$

Observe that :

$$\mathbf{v}_{n-1}^t \mathbf{b}_{n-1}^{n-1} = \sum_{k=n-1}^{1} a_k^{n-1} R_k = \sum_{k=1}^{n-1} R_k \, a_k^{n-1}$$

and

$$\mathbf{v}_{n-1}^t \mathbf{a}_{n-1}^{n-1} = \sum_{k=n-1}^{1} R_k \, a_{n-k}^{n-1} = \sum_{k=1}^{n-1} R_{n-k} \, a_k^{n-1}$$

Using above equations in the system of equations (A.1.12a) and (A.1.12b) we take :

$$\mathbf{a}_{n-1}^n = \mathbf{a}_{n-1}^{n-1} - a_n^n \, \mathbf{b}_{n-1}^{n-1} \qquad \text{(A.1.13a)}$$

$$a_n^n = \frac{R_n - \sum_{k=1}^{n-1} R_{n-k} \, a_k^{n-1}}{R_0 - \sum_{k=1}^{n-1} R_k \, a_k^{n-1}} \qquad \text{(A.1.13b)}$$

[Solutions of the last system of equations are given by

$$a_k^n = a_k^{n-1} - a_n^n \, a_{n-k}^{n-1} \quad (k = 1,2,\ldots,n-1) \quad]$$

The last two equations provide a complete recursive solution to the system of equations

$$\mathbf{R}_{nn} \, \mathbf{a}_n = \mathbf{r}_n$$

As en example we demonstrate the first steps implemented in our algorithm. We start by setting $n = 1$ in equation (A.1.13b) to take :

$$a_1^1 = \frac{R_1}{R_0}$$

For $n = 2$ we take from equation (A.1.13b)

$$a_2^2 = R_2 - R_1 a_1^1$$

and from equation (A.1.13a)

$$a_1^2 = a_1^1 - a_2^2 a_1^1$$

which are both known quantities from the previous step(s). We repeat above steps until $n = p$. The calculation of the total square error E_m can be incorporated in the above recursion by setting

$$E_m^0 = R_0 \quad \text{and using}$$

$$E_m^n = (1 - a_n^n) E_m^{n-1}$$

for $(1 \leq n \leq p)$.

APPENDIX A.2

Demonstration Proof of Equation 2.27

We will demonstrate that the following equality is true.

$$|A(e^{-j\omega})|^2 = \rho(0) + 2\sum_{k=1}^{p} \rho(k)\cos(k\omega) \tag{A.2.1}$$

where:

$$\rho(k) = \sum_{l=0}^{p-k} a_l\, a_{l+k} \quad ; \quad a_0 = 1$$

Observe that

$$\sum_{k=0}^{l} a_{l-k} = a_l + a_{l-1} + \cdots + a_0 = \sum_{k=0}^{l} a_k \tag{A.2.2}$$

First, as an intermediate step we shall prove the following relation:

$$\sum_{\substack{i=0 \\ i\neq j}}^{l}\sum_{j=0}^{l} a_i\, a_j = 2\sum_{i=1}^{l}\sum_{j=0}^{l-i} a_j\, a_{j+i} \tag{A.2.3}$$

Relation (A.2.3) is obviously true for $l = 1$. We assume that it is true for $l = l$, and we will prove that it is true for $l = l + 1$: The l.h.s of the equation A.2.3, for $l = l + 1$, can be successively equated as follows:

$$\sum_{\substack{i=0 \\ i\neq j}}^{l+1}\sum_{j=0}^{l+1} a_i\, a_j = \sum_{\substack{i=0 \\ i\neq j}}^{l}\sum_{j=0}^{l+1} a_i\, a_j + \sum_{j=0}^{l} a_{l+1}\, a_j$$

$$= \sum_{\substack{i=0 \\ i \neq j}}^{l} \sum_{j=0}^{l} a_i \; a_j + \sum_{i=0}^{l} a_i \; a_{l+1} + \sum_{j=0}^{l} a_{l+1} \; a_j$$

$$= \sum_{\substack{i=0 \\ i \neq j}}^{l} \sum_{j=0}^{l} a_i \; a_j + 2 \sum_{k=0}^{l} a_k \; a_{l+1} \qquad \text{(A.2.4)}$$

The r.h.s of the equation A.2.3, for $l = l + 1$, is successively equated as follows:

$$2 \sum_{i=1}^{l+1} \sum_{j=0}^{l-i+1} a_j \; a_{j+i} = 2 \sum_{i=1}^{l} \sum_{j=0}^{l-i+1} a_j \; a_{j+i} + 2 \sum_{j=0}^{(l+1)-(l+1)} a_j \; a_{j+l+1}$$

$$= 2 \sum_{i=1}^{l} \sum_{j=0}^{l-i+1} a_j \; a_{j+i} + 2 \; a_0 \; a_{l+1}$$

$$= 2 \sum_{i=1}^{l} \sum_{j=0}^{l-i} a_j \; a_{j+i} + 2 \sum_{i=1}^{l} a_{l+1-i} \; a_{l+i} + 2 \; a_0 \; a_{l+1}$$

[using equation A.2.2]

$$= 2 \sum_{i=1}^{l} \sum_{j=0}^{l-i} a_j \; a_{j+i} + 2 \sum_{i=1}^{l} a_{l+1} \; a_i + 2 \; a_0 \; a_{l+1}$$

$$= 2 \sum_{i=1}^{l} \sum_{j=0}^{l-i} a_j \; a_{j+i} + 2 \sum_{k=0}^{l} a_k \; a_{l+1} \qquad \text{(A.2.5)}$$

From equations A.2.4 and A.2.5 we see that the equation A.2.3 is true (for $l = l + 1$).

[Rem: We have assumed that the equation A.2.3 is true for $l = l$]

Using equation A.2.3 we will prove the relation A.2.1. For that we successivelly equate:

$$| A (e^{-j\omega}) |^2 = \left| \sum_{k=0}^{p} a_k \; z^{-k} \right|^2_{/z = e^{j\omega}} = \left| \sum_{k=0}^{p} a_k \; e^{-jk\omega} \right|^2$$

$$= \left| \sum_{k=0}^{p} a_k \cos(k\omega) - j \sum_{k=0}^{p} a_k \sin(k\omega) \right|^2$$

$$= \left[\sum_{k=0}^{p} a_k \cos(k\omega) \right]^2 + \left[\sum_{k=0}^{p} a_k \sin(k\omega) \right]^2$$

$$= \sum_{k=0}^{p} a_k^2 \cos^2(k\omega) + \sum_{\substack{k=0 \\ k\neq l}}^{p} \sum_{l=0}^{p} a_k a_l \cos(k\omega) \cos(l\omega) +$$

$$+ \sum_{k=0}^{p} a_k^2 \sin^2(k\omega) + \sum_{\substack{k=0 \\ k\neq l}}^{p} \sum_{l=0}^{p} a_k a_l \sin(k\omega) \sin(l\omega)$$

$$= \sum_{k=0}^{p} a_k^2 \left[\cos^2(k\omega) + \sin^2(k\omega)\right] +$$

$$+ \sum_{\substack{k=0 \\ k\neq l}}^{p} \sum_{l=0}^{p} a_k a_l \left[\cos(k\omega) \cos(l\omega) + \sin(k\omega) \sin(l\omega)\right]$$

$$= \sum_{k=0}^{p} a_k^2 + \sum_{\substack{k=0 \\ k\neq l}}^{p} \sum_{l=0}^{p} a_k a_l \cos[(k-l)\omega]$$

[using equation A.2.3]

$$= \sum_{k=0}^{p} a_k^2 + 2 \sum_{k=1}^{p} \sum_{l=0}^{p-k} a_l a_{k+l} \cos(k\omega)$$

The last equation is equation A.2.1, for

$$\sum_{k=0}^{p} a_k^2 = \rho(0)$$

and

$$\sum_{l=0}^{p-k} a_l a_{l+k} = \rho(k)$$

APPENDIX B

Software

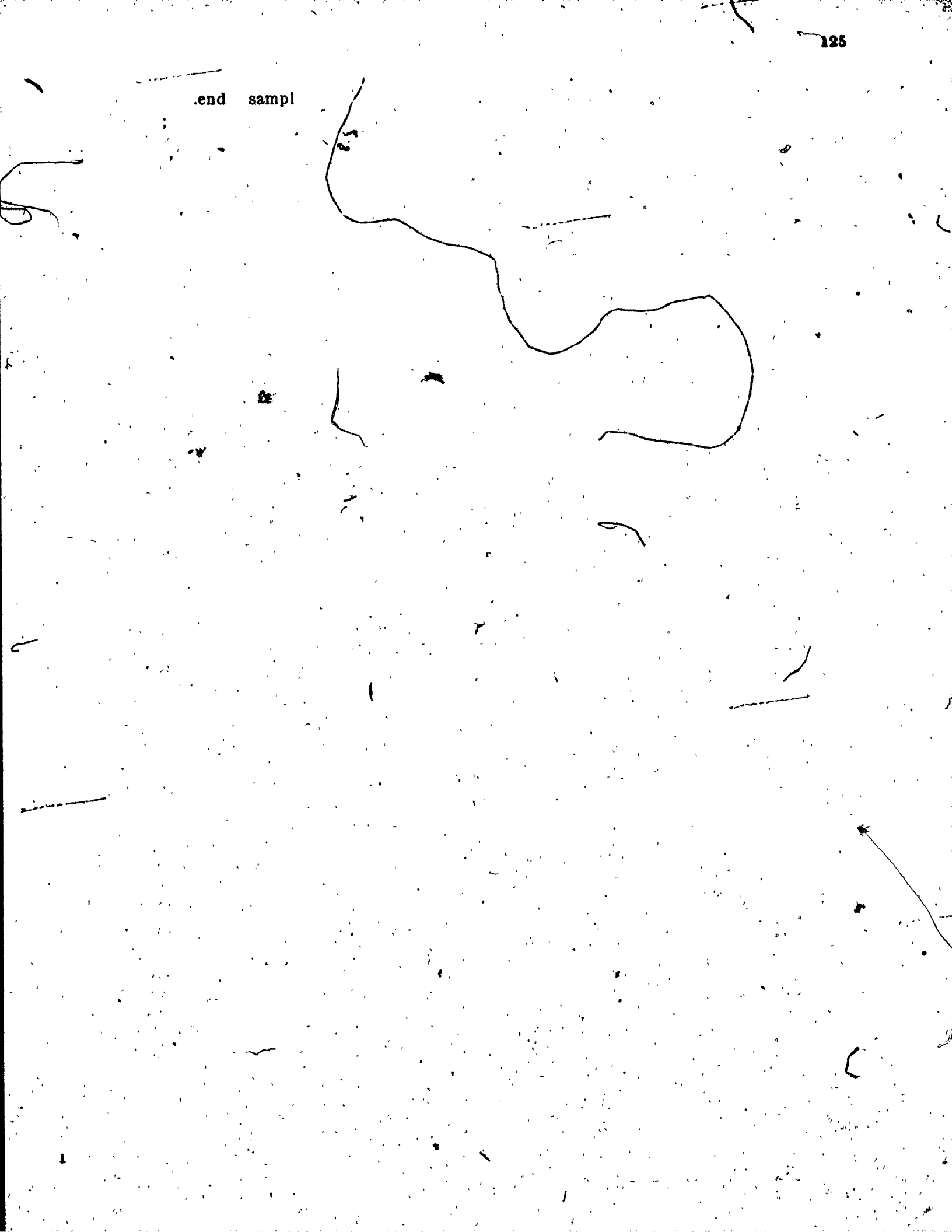
Files - Programs

```
;+
;
;  File adc.mac
;
;   This program is written to acquire data and
;  to store them in the disc file adc.dat.
;  For that:
;
;  1. Initializes register address for DRII-C
;     (// port) and KWII-K (clock real time).
;  2. Calls needed directives from the system
;     Macro Library and issues them.
;     [
;       $c          = to be used only once
;       $s          = to reenter
;       $a  = assembly-time FDB Macro call
;       $r  = run-time      #
;       $w  = write
;       FDB = File Descriptor Block
;       FSR = File Storage Region
;
;     a. fdbdf = allocate fdb - REQUIRED
;     b. fdrc  = init. Record-Access section of fdb
;     c. fdop  =  #    File-open        #
;     d. fdbk  =  #    Block-access     #
;     e. fsrsz =  #    FSR at assembly time
;     f. wait  = suspends program execution until a
;                recquested block I/O operation is
;                completed
;     g. write = virtual data blocks to a file
;     h. open  = s and prepares file for proccessing
;     i. close = s, orderly, file proccessing
;     j. alun  = instructs the system to assign a phy-
;                sical device unit to a Logical Unit
;                Number (LUN)
;                     ]
;  3. Opens datafile
;  4. Goes to a loop waiting for hi request B
;     (DRII-C, csr, bite 15)
;  5. Init. clocks A and B
;     Mode: Feed B to A (csrb, bite 5 = 1)
;     Rate: 5 KHz
;  6. Sets counters. Registers are used as:
;     %0 = addresses next empty spot in buff.
;     %1 = counts # of places left in buffer
;     %6 = pc
```

```
;       %7 = sp
;       7. Samples until buffer is full
;       8. Flushes buf. to disk
;       9. Over
;
;
;       (1)
;
        dr$csr  =  177540
        dr$out  =  177542
        dr$in   =  177544
;
        c$csra  =  170404
          c$bufa  =  170406
        c$cnta  =  170430
        c$csrb  =  170432
        c$bufb  =  170434
;
;       (2)
;
        .mcall  fdbdf$,fdrc$a,fdop$a,fdbk$a,fsrsz$,qiow$
        .mcall  wait$,write$,open$w,close$,alun$s,dir$,exit$s
;
;       (3)
;
sampl:          open$w  #fdb1,,,,,,err1
;
;       (4)
;
        tst             @#dr$csr;loop,external start
        bpl             .-4         ;branch on plus
;
;       (5)
;
        clr             @#c$bufa;clear A buffer
        mov             #401,@#c$csra;set A in B overflow rate
;                               ;enable A, mode 01
        mov     #-25.,@#c$bufb;freq.of B/25
        mov             #45,@#c$csrb;B: enable,freq=100 KHz,
;                               ;feed B to A
;
;       (6)
;
        mov             #5120.,r1;buffer size
        mov             #buff,r0;address of buf. in r0
;
```

```
;   (7)
;
loop:           tstb    @#c$csrb;test B overflow flag
    bpl         .-4     ;wait bit 07 to go hi
    bic         #200,@#c$csrb;togle bit 07
;
    mov         #0.,@#dr$out;conversion in channel 0
    tstb        @#dr$csr;wait EOC
    bmi         .-4     ;EOC is low on request Á
    mov         @#dr$in,(r0)+;data in buffer
    sob         r1,loop  ;if more room in buf.
;
;   (8)
;
    bic         #1,@#c$csra;stop A
    bic         #1,@#c$csrb;stop B
;
    write$  #fdb1,,,,,,,err2;flush in disk
    wait$       #fdb1,,,err3;wait end of write
;
;   (9)
;
    close$  #fdb1
err1:
err2:
err3:           exit$s
;
;   local data
;
    .even
    .nlist  bex
iost:           .blkw   2
    .even
    fsrsz$  0
fdb1:           fdbdf$
    fdbk$a  buff,10240.,,1,iost
    fdrc$a  fd.rwm
    fdop$a  3,filnam
filnam:         .word   4;dev,11,uic,11,nam
dev:.ascii  /dp2:/
uic: .ascii  /[004,004]/
nam:            .ascii  /adc.dat;*/
    .even
;
buff:           .blkw   5120.
;
```

```
.end   sampl
```



```
c
c
c               File tms.ftn.
c
c                Written to control to find LP coefficients.
c               Task build with tprep,tmain,tpost,toutp,tplotm,libnew.
c
      dimension   a(20,0:10),e(20),pnk(20,5,2),p(0:255)
      dimension   r(20,0:10)
      data                k2,L,n0,nfr,npo/256,256,0,20,10/
      common              /bl1/k2,L,len,n0,nfr,npo,ps1
c
      call        assign  (4,'lp:')
      call        assign  (5,'tt3:')
      call        assign  (6,'ti:')
      call        assign  (7,'tt11:')
c
c     k         = no. of points per frame  (time-domain).
c     L         = resolution of the PSD  (over upper-half circle).
c     len       = no. of non-noisy frames.
c     mf        = frame no. in process.
c     nfr       = no. of frames to be processed.
c     npo       = no. of autocorrelation coeff.-1 (poles).
c     ps1       = saved last value of last frame  (used in preemphasis).
c
      call        twind1  (p)
      call        ttrfn1  (a,e,pnk,p)
c
  200             write   (6,3000)
  3000            format  (2x,'How many frames ?(20. max.): ',$)
      read        (6,*)   nfr
      if          ((nfr.gt.20).or.(nfr.lt.1))goto 999
c
      open        (unit=1,name='adc.dat',readonly,type='old',
     @                    access='direct')
      mf          = 0
      len         = 0
      ntok        = 0
      ps1         = 0.0
c
  201             mf      = mf + 1
      if          (mf.gt.nfr)goto202
      call        tread   (p,mf)
      call        tprem   (p)
      call        twind2  (p)
      call        tauto   (p,r,mf)
```

```
      goto    201
c
 202  close   (unit=1)
      if      (len.eq.0)goto204
      mf      = nfr
      call    tcoef   (a,e,r)
      call    ttrfn2  (a,e,pnk,p)
      call    tpost   (e,nscore,ntok,pnk,r)
c
      call    tplotm  (p)
c
      write   (6,3005)
 3005 format  (2x,' Output to the line printer ? (y/n) : ',$)
      read    (6,3006) lpr
 3006 format  (a1)
      if      (lpr.ne.'y')goto205
      write   (4,3008)
 3008 format  ('1')
      call    tout1   (a,e,pnk,r,mf)
      call    tout9   (pnk,nscore)
      goto    205
 204  write   (6,3010)
 3010 format  (2x,' It is only noise in the 20 frames')
 205  goto    200
 999  stop
      end
```

```
c
c
c       File tprep.ftn.
c
c        Collection of subroutines. Controlled by tms.ftn.
c       For signal pre-processing.
c       Task build with tms,tmain,tpost,toupt,tplotm,libnew.
c
c       Subroutines
c
c         1. tread
c         2. tprem
c         3. twind1(entry: twind2)
c         4. tauto
c
c
c
c
      subroutine   tread(p,mf)
c
c          It reads data from adc.dat file, converts them into
c       equivalent, normalizes, and stores them into array p.
c
      dimension    n(0:255),p(0:255)
      common              /bl1/k2,L,len,n0,nfr,npo,ps1
c
      hbd          = - 4095. / 2047.
      read         (1'mf,end=999)(n(j), j=n0,k2-1)
      do 12        m1 = n0,k2-1
        p(m1)      = n(m1) /2047.
        if         (n(m1).le.2047)goto12
        p(m1)      = p(m1) + hbd
   12 continue
  999             return
      end
c
c
      subroutine   tprem(p)
c
c          Preemphasis:'Transfer function is  H(z) = 1 - a/z.
c       It is used to amplify the high frequences to aid in
c       recognition.
c
      dimension    p(0:255)
```

```
      common          /bl1/k2,L,len,n0,nfr,npo,ps1
      data            a/0.965/
c
      ps2       = p(k2-1)
      do 14     m1 = k2-1,1,-1
   14 p(m1)     = p(m1) - a * p(m1-1)
      p(n0)     = p(n0) - a * ps1
      ps1       = ps2
  999           return
      end

c
c
      subroutine  twind1(p)
c
c               Windowind of data frames using Hamming.
c
      dimension   wind(0:127),p(0:255)
      data              pi2/6.283185/
      common            /bl1/k2,L,len,n0,nfr,npo,ps1
c
      pi2k2     = pi2 / (k2-1)
      do 15     m1 = 0,(k2-1)/2
   15 wind(m1)  = 0.54 - 0.46 * cos(pi2k2 * m1)
  999           return
c
      entry       twind2(p)
c
      do 16     m1 = 0,(k2-1)/2
        p(m1)   =     wind(m1) * p(m1)
   16 p(k2-m1-1) =  wind(m1) * p(k2-1-m1)
 9999           return
      end

c
c
      subroutine  tauto(p,r,mf)
c
c               To find set of autocorelation coefficients.
c               r  = matrix of R's. (k1x(npo+1)).
c               npo= number of poles.
c               th = threshold of power (magn. of r(0)).
c
      dimension   r(20,0:10),p(0:255)
      common            /bl1/k2,L,len,n0,nfr,npo,ps1
      data              th/0.0001/
c
```

```
      r(mf,0)       = 0.0
      do 17         m2 = 0,k2-1
   17 r(mf,0)       = r(mf,0) + p(m2) ** 2
      if            (r(mf,0).lt.th)goto 999
      len           = len + 1
      r(len,n0)     = r(mf,n0)
      do 18         m2 = 1,npo
       r(len,m2)= 0.0
       do 18        m3 = 0,k2-1-m2
   18 r(len,m2)= r(len,m2) + p(m3) * p(m2+m3)
c
c     Normalize R's. Keep values of R(0)'s in r(x,0).
c
      do 19         m2 = 1,npo
   19 r(len,m2) = r(len,m2) / r(len,0)
  999               return
      end
```

```
c
c
c                File tmain.ftn.
c
c                 Collection of subroutines. Controlled by tms.ftn.
c                For signal main-processing.
c                Task build with tms,tprep,tpost,toupt,tplotm,libnew.
c
c                Subroutines:
c
c                 1. tcoef
c                 2. ttrfn1(entry:*ttrfn2)
c
c
c
c
c
      subroutine    tcoef(a,e,r)
c
c                    To find LP coefficients using Durbin's algorithm.
c                    Rem:   e(0) = r(0) in iteration
c
      dimension     aa(10,10),a(20,0:10),e(20),r(20,0:10),ee(10)
      data                  r0/1.000/
      common                /bl1/k2,L,len,n0,nfr,npo,ps1
c
      do 14         m1 =  1,len
       hk1          = -r(m1,1)/r0
       aa(1,1)  =   hk1
       ee(1)          =  (1. - hk1*hk1) * r0
       do 12        m2 =  2,npo
        hk1           =  0.0
        do 10  m3 = 1,m2-1
 10     hk1           =  hk1 + aa(m3,m2-1)*r(m1,m2-m3)
        hk1           = -(r(m1,m2) + hk1) / ee(m2-1)
        aa(m2,m2) = hk1
        a(m1,m2)  = hk1
        do 11  m4 = 1,m2-1
         aa(m4,m2) = aa(m4,m2-1) + hk1*aa(m2-m4,m2-1)
 11     a(m1,m4)  =   aa(m4,m2)
        ee(m2)    =  (1.-hk1*hk1) * ee(m2-1)
 12   continue
      e(m1) = ee(npo)
 14 continue
 999              return
```

```
      end

c
c
      subroutine    ttrfn1(a,e,pnk,p)
c
c                    Calculates, stores (in tf) values of the sine fn.
c                   It is called before processing.
c                   It calculates L values for the upper semi-circle.
c                   The value of L is set in tms.
c                   Sampling freq. = 39270  rad/s.
c
      dimension     a(20,0:10),e(20),pnk(20,5,2),p(0:255)
      dimension     rr(0:10),tf(0:255,10)
      common                /bl1/k2,L,len,n0,nfr,npo,ps1
      data                  pi,npk/3.14159,5/
c
c                    This to initialize the a(x,0)
c
      do 21         m1 = 1,nfr
   21 a(m1,0)       = 1.0
c
      rat           = pi / (L-1)
      do 22         m1 = 0,L-1
       h1           = rat * m1
       do 22        m2 = 1,npo
        h2          = h1 * m2
   22 tf(m1,m2)     = cos(h2)
  999               return
c
      entry         ttrfn2(a,e,pnk,p)
c
c                    To compute the PSD of the non-noisy frames.
c                   It also makes the peak picking.
c
      do 38         m1 = 1,len
       rr(0)        = 0.0
       do 31        m3 = 0,npo
   31  rr(0)        = rr(0) + a(m1,m3)**2
       do 33        m2 = 1,npo
        rr(m2) = 0.0
        do 32  m3 = 0,npo-m2
   32   rr(m2) = rr(m2) + a(m1,m3) * a(m1,m2+m3)
   33  rr(m2)  = rr(m2) * 2.
       do 35        m2 = 0,L-1
        denom  = rr(0)
```

```
      do 34 m3  = 1,npo
        denom   = denom + tf(m2,m3) * rr(m3)
34    continue
      p(m2) = e(m1) / denom
35  continue

c
c
c                 Peak picking.
c
      nk          = 1
      do 37       m2 = 1,L-2
        if        ((p(m2).gt.p(m2-1)).and.
     @                    (p(m2).gt.p(m2+1))) goto 36
        goto      37
36    pnk(m1,nk ,1)     = m2
      pnk(m1,npk,1)     = nk
      pnk(m1,nk ,2)     = p(m2)
      nk          = nk + 1
      if          (nk.eq.npk)goto38
37  continue
38  continue
9999              return
    end
```

```
c
c
c           File tpost.ftn.
c
c             It is written for the signal post-processing.
c           Task build with tms,tprep,tmain,tplotm,libnew,toutp.
c
c           Subroutines:
c
c             1. tpost
c             2. tsort
c             3. tring
c             4. tknock
c
c
c
c
c
      Subroutine  tpost(e,nscore,ntok,pnk,r)
c
c             Written to branch for various signals.
c
      dimension   e(20),pnk(20,5,2),r(20,0:10)
      dimension   dst(4),frq(4),mne(20)
      common            /bl1/k2,L,len,n0,nfr,npo,ps1
      data              dst,errth/3.,5.,7.,9.,0.55/
c
c             dst   = threshold distance for peaks.
c             errth = threshold for max. error.
c
      call        tsort   (e,mne)
      if          (e(mne(1)).lt.errth)goto41
      write       (4,3041)
      write       (6,3041)
      goto        999
   41 call        tring   (dst,frq,pnk,nscore,ntok)
      if          (ntok.eq.1)goto999
      call        tknock  (dst,e,frq,mne,nscore,ntok,pnk,r)
      if          (ntok.eq.2)goto999
      write       (4,3042)
      write       (6,3042)
 3041             format  (/,9x,'Min. error is greater than threshold')
 3042             format  (/,9x,'This is an unrecognized signal.',/)
  999             return
```

```
      end
c
c
      Subroutine tsort(e,mne)
e
c                 To short the error vector.
c
      dimension   e(20),mne(20)
      common            /bl1/k2,L,len,n0,nfr,npo,ps1
c
      do 51       m1 = 1,len
 51   mne(m1)     = m1
      do 53       m1 = 1,len
       do 53      m2 = 1,len-1
        if        (e(mne(m2)).gt.e(mne(m2+1)))goto52
        goto      53
 52     msave     = mne(m2)
        mne(m2)   = mne(m2+1)
        mne(m2+1)       = msave
 53   continue
 999              return
      end


c
c
      subroutine  tring(dst,frq,pnk,nscore,ntok)
c
c                 This to test if the pattern corresponds to telephone
c                 ring (i.e. checks if peaks close to 104, and 136/244 exist).
c                 It sets the token (ntok=1) if signal is known.
c
      dimension   pnk(20,5,2)
      dimension   dst(4),frq(4)
      common            /bl1/k2,L,len,n0,nfr,npo,ps1
      data              wpr1,wpr2,wpr/54.5,4.2,58.7/
      data              prthc,prthr/99.0,50.0/
c
c     dst         = Freq. tolerance from fixed peaks (experimental).
c     frq         = Array of template's frequencies.
c     frth        = Freq. threshold (= disregard freq. of 1rst half-quadr).
c     nsc1        = Score for freq. around 104.
c     nsc2        = Score for freq. around 136/244.
c     nscore      = nsc1 + nsc2.
c     prthc       = prob. threshold for certainty.
c     prthr       =     "     rejection.
c     wpr         = weight of probabilities (= average weight of
```

```
c                amplitudes (experimental)).
c
c                Initialize.
c
      frq(1)     = 104.
      frq(2)     = 136.
      frq(3)     = 244.
      frth       = L / 4.0
      nscore     =   0
      nsc1       =   0
      nsc2       =   0
c
c                Match templates + signal.
c                Decision based also on absolute magnitudes.
c
      do 62      m1 = 1,len
       m2        =  1
c                      Remove frq. bellow 64.
       if        (pnk(m1,m2,1).lt.frth)m2= m2 + 1
c                      Measure distance from 104.
       dist      = abs (pnk(m1,m2,1) - frq(1))
       if        (dist .le. dst(1))nsc1= nsc1 + 1
       m2        = m2 + 1
c                      Choose second largest magnitude.
      if         (pnk(m1,m2,2) .lt. pnk(m1,m2+1,2))goto61
c                      Measure distance from 136.
       dist      = abs (pnk(m1,m2,1) - frq(2))
       if        (dist .le. dst(2))nsc2= nsc2 + 1
       goto      62
c
 61   m2         = m2 + 1
c                      Measure distance from 244.
       dist      = abs (pnk(m1,m2,1) - frq(3))
       if        (dist .le. dst(3))nsc2= nsc2 + 1
 62 continue
c
c                 Calculate score and weighted probabilities.
c                Output decision.
c
      nscore     = nsc1 + nsc2
      prob       = 1[illegible] * (nsc1*wpr1 + nsc2*wpr2) / (len*wpr)
      if         [illegible]t.prthr)go to 999
      ntok       = 1
      if         (prob.gt.prthc)write(4,3061)
      if         (prob.gt.prthc)write(6,3061)
      if         (prob.le.prthc)write(4,3062)prob
```

```
      if          (prob.le.prthc)write(6,3062)prob
 3061             format (/,9x,'It is certainly a telephone ring. Ha, Ha !',/)
 3062             format (/,9x,'It is a telephone ring with weighted prob. ',
     @                    f7.3,' %.',/)
 999              return
      end

c
c
      subroutine  tknock(dst,e,frq,mne,nscore,ntok,pnk,r)
c
c                  To test for knocking.
c                 It sets ntok (=2) in success.
c
      dimension   e(20),pnk(20,5,2),r(20,0:10)
      dimension   dst(4),frq(4),mne(20),whi(4)
      common            /bl1/k2,L,len,n0,nfr,npo,psl
      data              n4/4/
c
c     lenpk             =  # of peaks in template.
c     mne               =  array of frame # of min. error in
c                          descending order.
c     prnorm            =  probability normalizer.
c     whi               =  array of template's weigth of intensities.
c
c                 Initialize.
c
      lenpk       =  ifix(pnk(mne(1),5,1))
      nscore      =  0
      nsc1        =  0
      nsc2        =  0
      prob        =  0.0
      prnorm      =  0.0
c
c
c                 Branching (synchronization with tring).
c
      if          (len.lt.8)goto999
      if          ((e(mne(10)).le.0.5).and.(len.ge.10))goto78
c
c                 Form template from one signal frame.
c                 Decision based on min.error.
c
      do 71       m1 = 1,lenpk
       frq(m1)    = pnk(mne(1),m1,1)
       whi(m1)    = pnk(mne(1),m1,2)
 71   prnorm            = pnk(mne(1),m1,2) + prnorm
```

```
c
c     Template matching.
c
      do 75          m1 = 2,n4
       do 74         m2 = 1,lenpk
        dist1        = 256.
        do 72        m3 = 1,pnk(mne(m1),5,1)
 72     dist1        = amin1(dist1,abs(pnk(mne(m1),m3,1)-frq(m2)))
       if            (dist1.le.dst(m2))goto73
       goto          74
 73    prob          = prob + whi(m2)
       nscore= nscore + 1
 74    continue
 75  continue
      prob           = 100. * prob / (prnorm * (n4-1))
      write          (4,3071) lenpk
      write          (6,3071) lenpk
      if             (prob.lt.50.0)goto999
      ntok           = 2
      write          (4,3072) prob
      write          (6,3072) prob
      mh1            = 0
      do 76          m1 = 1,n4-1
       do 76         m2 = m1+1,n4
 76   mh1            = max0(mh1,iabs(mne(m1)-mne(m2)))
      if             (mh1.le.6)goto 77
      goto           999
 77   write          (4,3076)
      write          (6,3076)
      goto           999
c
c                    Print comments for possible characteristic sound.
c
 78   write          (4,3073)
      write          (6,3073)
      if             ((len.ge.15).and.(e(mne(15)).le.0.5))goto79
      goto           81
 79   write          (4,3074)
      write          (6,3074)
 81   if             ((len.eq.20).and.(e(mne(20)).le.0.5))goto82
      goto           999
 82   write          (4,3075)
      write          (6,3075)
3071                 format  (/,9x,'# of peaks in template is ',i2)
3072                 format  (/,9x,'It is a knock with weigh. prob. ',f7.3,' % .')
3073                 format ,(/,9x,'It is a characteristic signal but not a
```

```
     @ knock on wood.')
3074            format (/,9x,'Above statement is very probable.')
3075            format (/,9x,'Above two statements are true.')
3076            format (/,9x,'Above statement is based in a limited # of frames')
999             return
      end
```

```
c
c
c                File toutp.ftn
c
c               Task build with tms,tprep,tmain,tpost,tplotm,libnew.
c
c               Subroutines
c
c                1. toutl(entry: tout9)
c                2. times
c
c
c
c
c
      subroutine toutl(a,e,pnk,r,mf)
c
c                It is written for test purposes. It outputs in LP
c               test values of its arguments
c
      dimension   a(20,0:10),e(20),r(20,0:10),pnk(20,5,2)
      common              /bl1/k2,L,len,n0,nfr,npo,ps1
c
      write       (4,3400)
      write       (4,3401) npo,nfr,mf,len
      write       (4,3400)
      write       (4,3402)
      write       (4,3400)
      do 91       m1 = 1,len
 91   write       (4,3403) m1,r(m1,0),m1,e(m1)
 3400             format (/)
 3401             format (9x,'npo = ',i3,8x,'nfr  = ',i3,8x,'mf = ',i3,
     @                    8x,'len = ',i3)
 3402             format (9x,'R(x,0), Error min. (from zero to nfr)'/)
 3403             format (/,9x,'R(',i2,',0) = ',e13.6,10x,'e(',i2,') = ',f12.9)
 999              return
c
      entry       tout9(pnk,nscore)
c
      write       (4,3400)
      write       (4,3404)
      write       (4,3405)
      write       (4,3406) nfr
      write       (4,3400)
```

```
      do 93       m1 = 1,len
       write      (4,3407) m1,(pnk(m1,m2,1),m2=1,pnk(m1,5,1))
   93 write       (4,3408) (pnk(m1,m2,2),m2=1,pnk(m1,5,1))
      write       (4,3400)
      write       (4,3409) nscore,len
   3404           format (9x,'Peaks : (255 corresponds to 3.125 KHz)')
   3405           format (9x,'Absolute Magnitudes'/)
   3406           format (9x,'nfr = ',i3)
   3407           format (/,9x,i2,4('    ',f10.2,5x))
   3408           format (12x,   4('    ',f15.9))
   3409           format (2x,' Score = ',i3,' out of ',i3,' non-noise frames')
   9999           return
      end

c
c
      subroutine  times
c
c                  To print in ti: and lp: the time it takes for
c                  auto. coef. to be calculated, for nfr(ames)).
c
      double precision    tc
      call        time(tc)
      write       (6,3420) tc
      write       (4,3420) tc
   3420           format (10x,'Time of autocorrelation processing = ',8a)
   999            return
      end
```

```
c
c
c           File tplotm.ftn
c
c            It is written to plot the PSD of a frame.
c           Task build with tms,tprep,tmain,tpost,toutp,libnew.
c           For time-domain signal run 2dproj.
c
c           Subroutine
c
c            1. tplotm
c
c
c
c
c
      subroutine    tplotm(p)
c
      dimension  nx(7),nxyh(1),nyn(10),nynh(2),nframe(4),p(0:255)
      common              /bl1/k2,L,len,n0,nfr,npo,ps1
c
      data          nx      /' 0','.2','.4','.6','.8','1.',' w'/
      data          nyn/'- ','.2','- ','.4','- ','.6','- ','.8','- ','1.'/
      data          yxa,xya,xyb,yxb/0.01,0.01,-0.05,-0.05/
      data          xmax,xmin,ymax,ymin/1.2,-0.1,1.2,-0.1/
      data          nframe  /'Fr','am','e ',' '/
c
      data          idev,itap/1,7/
c
c             Initialize
c
      write         (6,3003)
 3003             format  (2x,'What device ?  (17=tectronix,25=calcomp): ',$)
      read          (6,3004) idev,itap
 3004             format (2i1)
      if            ((idev.ne.1).and.(idev.ne.2))goto 999
      if            ((itap.ne.5).and.(itap.ne.7))goto 999
c
 205              call      plotin(xmax,xmin,ymax,ymin,idev,itap)
c
c             Plot.
c
      hml           = float(L-1)
      pmx           = 0.0
      do 23         m1 = 0,L-1
```

```
 23 pmx          = amax1(pmx,abs(p(m1)))
    do 24        m1 = 0,L-1
 24 p(m1)        = p(m1) / pmx
c
    call         movee  (0.0,p(0))
    do 25        m1 = 1,L-1
 25 call         draw   (m1/hml, p(m1))
c
c                Draw axis.
c
    call         movee  (0.0,0.0)
    call         draw   (1.1,0.0)
    call         movee  (0.0,0.0)
    call         draw   (0.0,1.1)
    do 26        m1 = 1,10
      call       movee  (m1/10. ,-yxa)
 26 call         draw   (m1/10. ,+yxa)
    do 27        m1 = 1,10
      call       movee  (-xya,m1/10.)
 27 call         draw   (+xya,m1/10.)
    call         tsend
 999             return
    end
```

```
c
c
c   File 2dproj.ftn
c
c
c                   It reads data from adc.dat file, converts them,
c   normalizes, and plots (2560 points max).
c
c   Notes:
c     1. Length of adc.dat is 5120 words / 10240 bytes (20 blocks /
c        records).
c     2. Dumps real time data into file adcr.dat (unformated)
c        3. Ten percent margin is left.
c
      dimension  iname(22),itec(4),ical(4),n(5120),nx(7),ny(7),nxyh(1)
      dimension  nyn(10),nynh(2)
      virtual          dr(5120)
c
      data          nx       /' 0','.2','.4','.6','.8',' 1',' t'/
      data          ny       /' 0','.2','.4','.6','.8',' 1',' A'/
      data          nyn/'- ','.2','- ','.4','- ','.6','- ','.8','- ','1 '/
      data          yxa,xya,xyb,xps,yps,yxb/0.025,0.01,-0.05,0.0,0.0,-0.1/
      data          xmax,xmin,ymax,ymin/1.2,-0.1,1.2,-1.2/
c
c   Initialize
c
      call          assign  (5,'tt3:')
      call          assign  (6,'ti:')
      call          assign  (7,'tt6:')
c
      write         (6,111)
  111               format  (2x,'What device ?  (17=tec, 25=cal): ',$)
      read          (6,121) idev,itap
  121               format  (2i)
      if            ((idev.ne.1).and.(idev.ne.2))goto 999
      if            ((itap.ne.5).and.(itap.ne.7))goto 999
c
      call          plotin(xmax,xmin,ymax,ymin,idev,0,itap)
c
      write         (6,262)
  262               format  (2x,'How many points ?(max. 5120): ',$)
      read          (6,*)  maxlen
      if            ((maxlen.gt.5120).or.(maxlen.lt.1))  maxlen=100
c
c   Read data, normalize.
c
```

```
      open        (unit=1, name='adc.dat', readonly,
     @                    type='old', access='direct')
      m           = maxlen / 257 + 1
      do 313      k = 1,m
       kk         = (k-1)*256 + 1
       read       (1'k)    (n(j), j=kk,(kk+255))
  313             continue
c
      close       (unit=1)
      do 314      k = 1,maxlen
       if         (n(k).le.2047)dr(k) = n(k)/2047.
       if         (n(k).gt.2047)dr(k) = n(k)/2047. - 2.
  314             continue
c
c     Normalize
c
      drmx = 0.0
      do 1300     m1 = 1,maxlen
 1300             drmx   = amax1(drmx,abs(dr(m1)))
      do 1301     m1 = 1,maxlen
 1301             dr(m1) = dr(m1) / drmx
c
c     Plot.
c
      hml         = float(maxlen)
      call        movee(1./hml,dr(1))
      do 315      k = 2,maxlen
       call       draw (k/hml, dr(k))
  315             continue
c
c     Real time data in file adcr.dat, for future consideration.
c     First entry indicates the no. of elements.
c
      open        (unit=2, name='adcr.dat', type='new')
      write       (2,*)    maxlen
      do 316      k = 1;maxlen
       write      (2,*)    dr(k)
  316             continue
      close       (unit=2)
c
c     Draw axis.
c
      call        movee(0.0,yps)
      call        draw (1.1,yps)
      call        movee(xps,ymin)
      call        draw (xps,ymax)
```

```
      do 353  i = 1,10
       call          movee(i/10. ,yps-yxa)
 353                 call      draw (i/10. ,yps+yxa)
      do 363         i = -10,10
       call          movee(xps-xya,i/10.)
 363                 call      draw (xps+xya,i/10.)
c
c     Label axis.
c
      do 375         k = 2,6
       h1            = 0.2*(k-1.) - 0.02
       call          movee  (h1,yxb)
       nxyh(1)       = nx(k)
       call          chars  (nxyh,1,4)
 375                 continue
c
      call           movee  (1.07,yxb)
      nxyh(1)        = nx(7)
      call           chars  (nxyh,1,4)
c
      do 385         k = 2,7
       h1            = 0.2*(k-1.) - 0.01
       call          movee  (xyb,h1)
       nxyh(1)       = ny(k)
       call          chars  (nxyh,1,4)
 385                 continue
      do 395         k = 2,6
       h1            = -0.2*(k-1.)
       call          movee  (2*xyb,h1)
       nynh(1)       = nyn(2*k-3)
       nynh(2)       = nyn(2*k-2)
       call          chars(nynh,2,4)
 395                 continue
c
      nxyh(1)        = nx(1)
      call           movee  (xyb,0.0)
      call           chars  (nxyh,1,4)
c
      call           movee(xmin,ymin)
      call           tsend
 999                 stop
      end
```