Vector quantization techniques for speech coding.

Sumesh. Kaul

University of Windsor

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

Recommended Citation
https://scholar.uwindsor.ca/etd/4514
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. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

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. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS RÉCU
VECTOR QUANTIZATION TECHNIQUES
FOR SPEECH CODING

by

Sumesh Kaul

A thesis
presented to the University of Windsor
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
The Department of Electrical Engineering

Windsor, Ontario, 1984

(c) Sumesh Kaul, 1984
To my parents

Mr. & Mrs. Padam Nath Kaul

and my younger brother

Sanjay

for their love, encouragement and support.
ABSTRACT

Efficient transmission and storage of speech is the main concern of this study. Most of this work is devoted to Vector or Block Quantization of various parameters of speech to obtain an "optimum" Linear Prediction Coding (LPC) speech compression system.

In the past detailed studies and comparisons have been made on low bit rate coding schemes based upon scalar quantization; but such systems have effectively reached their theoretical performance limits. Currently various new techniques stemming from different linguistic models of speech, probabilistic and spectral properties of speech are being developed to block quantize the speech parameters. One such technique is called LPC vector quantization, the topic of my Thesis.

The first stage of this work deals with high quality speech synthesis. Perceptual difference between the original speech and the simulated speech is used as the criterion to evaluate the performance of different schemes.
The second stage discusses the development of a codebook that will yield acceptable reconstruction, while providing significant data compression. Two different techniques of generating a codebook are outlined: In the first technique a codebook consisting of LPC parameters of speech is derived by taking the average of successive frames of each phoneme of the English language, extracted from a directory of spoken words. The second technique generates a codebook from a long training sequence of speech data. This technique is based on detecting the clusters of LPC parameters of speech. Experimentally, a large data base of speech is used to verify the performance of each technique. Results indicate that codebook generated through these two techniques offer a greater amount of speech compression with intelligibility of speech preserved. Also, the codebook constructed from the second technique is text and speaker independent. However, due to the granular nature of the reconstructed speech and the practical difficulty faced in synthesis of nasal and plosive sounds, some undesirable distortion is introduced.

The third stage discusses a new technique of generating a codebook that has significantly larger number of entries to facilitate a more accurate reconstruction of speech at the receiver. The processing time is reduced by operating the codebook in a two-level mode. Experiments, indicate that
the speech reconstructed from this codebook has a higher
Signal-to-quantization noise ratio (i.e., it alleviates the
distortion level).

Finally, various tree-search structures for codebooks
which significantly reduce the extensive arithmetic computa-
tions at the transmitter, are discussed.
ACKNOWLEDGMENTS

I owe biggest debt of gratitude to Dr. M. Shridhar, my thesis supervisor whose constant advice, enthusiasm and encouragement helped my work in a long way. His influence on both, this thesis, and my career, has been profound.

I would also like to thank Dr. M. Ahmadi, Prof. P. H. Alexander and Dr. J. J. Soltis for their interest and helpful suggestions. I am additionally grateful for Dr. Miller's assistance in the creation of the speech data base for this study.

 Appreciation is also due to fellow graduate student Mr. Kamandeep Singh, who generously offered his help in typing and proofreading this thesis.

Finally, the help of many graduate students during the period of my study is sincerely appreciated.
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Linear Prediction Model Of Speech</td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>Time Domain Representation</td>
<td>9</td>
</tr>
<tr>
<td>(b)</td>
<td>Frequency Domain Representation</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Linear Speech Production Model</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>An Example Depicting The Operation Of Centre Clipping</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>A Block Diagram Of The Pitch Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Linear Prediction Synthesizer</td>
<td>34</td>
</tr>
<tr>
<td>3.1</td>
<td>Block Diagram Of An LPC Vocoder</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Basic Concept Of Vector Quantization</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Block Diagram For Codebook Constructed</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>From Phonemes</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>(a) Two-Dimensional Minimum Distortion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Partition</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>(b) Centroids Of Figure (5.1a)</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>Illustration Of The Operation Of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Splitting</td>
<td>72</td>
</tr>
<tr>
<td>5.3</td>
<td>Block Diagram For Codebook Constructed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Using Splitting Technique</td>
<td>74</td>
</tr>
<tr>
<td>5.4</td>
<td>Two-Level Codebook</td>
<td>76</td>
</tr>
</tbody>
</table>
5.5 Implementation Of A Two-Level Codebook
(a) Encoder-------------------------------------- 78
(b) Decoder-------------------------------------- 78

5.6 Binary Tree-Search Structure For A
4-Bit Codebook------------------------------- 86

5.7 Non-Binary Tree-Search Structure For
A 4-Bit Codebook------------------------------- 90

B.1 (a) Original Speech Waveform (segment of 2 msec)
Sentence - "PAPA NEEDS TWO SINGERS"--------- 108
(b) Synthesized Waveform ('p' = 12 )
Sentence - "PAPA NEEDS TWO SINGERS"--------- 108

B.2 (a) Synthesized Waveform ('f' = 8 )
Sentence - "PAPA NEEDS TWO SINGERS"--------- 109
(b) Scalar Quantized Waveform ('p' = 8 )
Sentence - "PAPA NEEDS TWO SINGERS"--------- 109

B.3 Scalar Quantized Waveform With Non-Overlap
Technique (frame rate of 50 Hz).
Sentence - "PAPA NEEDS TWO SINGERS"--------- 110

B.4 (a) Reference Speech Waveform (Scalar Quantized)
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE"--------------------------- 111
(b) Vector Quantized Waveform (32-Level codebook).
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE"--------------------------- 111
B.5 (a) Vector Quantized Waveform (64-Level codebook).
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE" 112

(b) Vector Quantized Waveform (32/32 Level codebook).
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE" 112

B.6 (a) Vector Quantized Waveform (64/16 Level codebook).
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE" 113

(b) Vector Quantized Waveform (32/32 Level codebook) with Non-Overlap Technique.
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE" 113

B.7 Vector Quantized Waveform (64/16 Level Codebook) with Non-Overlap Technique.
Sentence - "EACH FORCE IN ITS OWN WAY IS CREATIVE" 114
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Average Distortion For Both Open And Closed Test Samples</td>
<td>83</td>
</tr>
<tr>
<td>II</td>
<td>Signal-to-Quantization Noise Ratio</td>
<td>84</td>
</tr>
<tr>
<td>III</td>
<td>Tree-Search Structures For A 10-Bit Codebook</td>
<td>89</td>
</tr>
<tr>
<td>IV</td>
<td>Overall Bit Rate For Quantizers</td>
<td>94</td>
</tr>
</tbody>
</table>
NOMENCLATURE

A codebook or reproduction alphabet
A's areas of the 'p' cylindrical sections of the vocal tract model
A_j jth second level codebook
A_0 an initial N-level codebook
A_s an N-level codebook after sth iteration
A(z) inverse filter
a_i's predictor or filter coefficients of the all-pole filter
C_i's cepstrum coefficients
C_{ij} the row i, column j element of a covariance matrix
C_R code rate of a quantizer
D_r speech data rate (or bit rate)
d(x,x) distortion measure for an input-output pair
E total squared error
e(n) prediction error between the actual data sample s(n) and the predicted sample \hat{s}(n)
F_r analysis frame rate
F_s sampling frequency
G gain of the all-pole filter
G_i's log area coefficients
H(z) transfer function of the all-pole filter

- x -
J  number of codewords in a second level codebook

K_i's  reflection coefficients

L  length of the training sequence

N  number of codewords in a codebook

NF  number of speech frames

NL  number of speech samples in an analysis frame

N_1  number of samples by which successive frames of speech overlap

P  pitch period for voiced speech

P  order of the all-pole filter

P(z)  pre-emphasis filter

1/P(z)  de-emphasis filter

s(t)  continuous speech waveform

s(n)  discrete speech sequence

\hat{s}(n)  predicted or estimated speech sample

\hat{s}(n)  de-emphasized speech sample

T_j  jth partition cell of the input training sequence

\|T_j\|  number of training vectors in the jth partition cell

u(t)  continuous excitation signal

u(n)  discrete excitation sequence

u_G(t)  glottal volume velocity waveform

u_L(t)  volume velocity waveform at the lips
$u_k^j$  
$k$th codeword of the $j$th second level codebook

$v(n)$  
pseudo-random number function

$X$  
LPC vector

$i$  
reproduction vector

$x_i$'s  
itth element of an LPC vector

$y_j$  
jth codeword in a codebook
CONTENTS

ABSTRACT .......................................... ii
ACKNOWLEDGEMENTS ................................ v
LIST OF FIGURES .................................. vi
LIST OF TABLES ................................... ix
NOMENCLATURE .................................. x
Chapter

I. INTRODUCTION .................................. 1

Speech Communication ............................. 1
  Importance Of Efficient Speech Communication 2
  Waveform coders .................................. 3
  Concept Of Speech Vocoder ......................... 4
  Speech Physiology ................................ 5
  Linear Prediction (LPC) Model .................... 7
  Speech Vcoders Using LPC Model ................. 8
  Scalar Quantization ................................ 10
  Vector Quantization .............................. 11
  Review Of Relevant Literature ................. 13
  Research Objectives ............................. 15
  Thesis Outline .................................. 15

II. ANALYSIS AND SYNTHESIS OF SPEECH USING LINEAR
  PREDICTION ...................................... 17

  Introduction ..................................... 17
  Speech Production Models ....................... 18
  Linear Prediction Model Of Speech Production 21
  Linear Prediction Analysis ...................... 22
  Linear Prediction Coefficients ................. 22
  Pitch And V/U V Analysis ....................... 25
  Pitch Period Estimation Using The
    Autocorrelation Function .................... 26
  Gain Computation ................................ 31
  Linear Prediction Synthesis ................... 33
  Other Speech Parameters ....................... 36
  Reflection Coefficients ....................... 36
  Log Area Ratio Coefficients ................... 37
  Cepstrum Coefficients .......................... 38
  Data Collection & Preprocessing ............... 38
  Analysis/Synthesis Details .................... 39
  Computer Simulation & Results ............... 40

-xiii-
III. LINEAR PREDICTIVE VOCODER ........................................... 42
   Introduction ......................................................... 42
   Transmitter Section ............................................... 44
   Scalar Quantization of Parameters ............................... 46
   Receiver Section ................................................... 47
   Synthesis filter and De-emphasis ................................ 47
   Experimental Results .............................................. 48

IV. LPC VECTOR QUANTIZATION ............................................ 51
   Introduction ......................................................... 51
   Block Quantizers ................................................... 52
   Nearest Neighbour Rule ........................................... 55
   Distortion Measure ................................................ 55
   Codebook Generation ............................................... 56
      Speech Sounds ................................................... 57
      Phonemes .......................................................... 57
      Codebook Generation Steps .................................... 58
   Results .................................................................. 61

V. VECTOR QUANTIZATION USING CLUSTERING TECHNIQUE ............... 63
   Introduction ......................................................... 63
   Vector Quantization Overview .................................... 63
   Performance Parameters Of A VQ .................................. 64
   Preview Of VQ Design ............................................... 65
      Centroid Definition .............................................. 66
   VQ design algorithm ............................................... 68
   Splitting Technique ................................................ 71
      Algorithm Using Splitting Technique ........................ 73
   Two-Level VQ Algorithm .......................................... 75
   Two-Level Encoder/Decoder ....................................... 77
   Experiments And Results .......................................... 79
      Training Sequence And Analysis Considerations ........... 80
      Performance Evaluation ....................................... 80
      Perceptual Listening Tests .................................... 80
      Quantitative Evaluation ...................................... 81
   Tree-Search Technique ............................................. 85
      Generation Of Tree Structures ................................ 85
      Modified Tree Structure ...................................... 88
   Bit Rate of VQ ...................................................... 91

VI. SUMMARY AND CONCLUSIONS ............................................ 95
   LPC Analysis/Synthesis Of Speech ................................ 96
   LPC Vocoder ......................................................... 96
   Vector Quantization Techniques .................................. 97

VII. SUGGESTIONS FOR FUTURE RESEARCH ................................. 99
<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.  COVARIANCE/AUTOCORRELATION METHODS</td>
<td>101</td>
</tr>
<tr>
<td>B.  SPEECH WAVEFORMS</td>
<td>107</td>
</tr>
<tr>
<td>C.  LISTING OF COMPUTER PROGRAMS</td>
<td>115</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>146</td>
</tr>
<tr>
<td>VITA AUCToris</td>
<td>149</td>
</tr>
</tbody>
</table>
Chapter I

INTRODUCTION

1.1 SPEECH COMMUNICATION

Speech is probably the most important source of communication among humans. Humans are unique in their ability to transmit information with their voice. Speech sounds radiated into air are detected by the human ear and recognised by the brain. Acoustic transmission and reception of speech works only over limited distances.

Historically, investigators have endeavoured earnestly to communicate at distances. For example, the ancient Greeks used intricate systems of signal fires to communicate between cities. Later, drum sounds were used to communicate over distances, again calling upon the sense of sound.

The advent of electronics coupled with digital techniques has substantially changed this picture. At present, due to the development of commercial telephony people can effectively communicate with each other in the most natural mode of communication (i.e. speech sounds). Also, the development of large high-speed digital computers has enabled speech communication to take on a new dimension, i.e. man-
machine communication by voice. This has mainly been made possible by the ability of the computers to store and rapidly manipulate large quantities of speech data.

1.1.1 Importance of Efficient Speech Communication

The amount of data contained in a speech signal (in digital form) is enormous. For example, one second of digital speech is represented by 8000 speech samples (using 8 KHz as sampling rate). Thus, at normal bit rates \(^1\) (expressed in bits per second), it takes a great deal of bandwidth (transmission capacity) to transmit such signals. In situations where for economical and other reasons, only limited bandwidth is available, it is often required to maximize the number of users for a given transmission medium. This problem can be solved effectively by compressing the speech data (using data compression techniques), so as to reduce the bit rate significantly.

Computers that can speak require a sophisticated vocabulary of speech words. They use this vocabulary to form a wide variety of messages and contexts. Due to the large size of these vocabularies, an enormous amount of speech data is needed to be stored in a digital computer memory. For exam-

\(^1\) For commercial telephony, speech is transmitted at a rate of 64 Kbps. Here, each speech sample is represented by 8 binary words (bits) and a sampling rate of 8 KHz is used.
ple, a vocabulary comprising typically of 2 minutes of speech would require approximately a memory of $8 \times 10^6$ bits, at a data rate of 64 Kbps. The speed and size limitations of computer memories make it necessary to use speech compression techniques to store the speech data efficiently.

Efficient communication suggests, transmission or storage of minimum information necessary to specify a speech signal and to bring forth a desired response. Thus, the basic problem is to design a compression system that can transmit or store speech sounds with maximum efficiency, while maintaining the perceptual information. Speech compression systems have traditionally fallen into two general classes: waveform coders and Vocoder [speech coders].

1.2 WAVEFORM CODERS

Waveform coders deal with the representation of an analog speech signal waveform in a digital form, so that the waveform can be reconstructed from its digital representation. These coders can communicate a digital approximation of the original speech signals at bit rates, typically between 16000 and 64000 bits per second.

Perhaps, the simplest form of a waveform coder is Pulse-Code Modulation scheme (PCM), in which a digitized
speech signal is uniformly quantized in amplitude. One widely used variation of the PCM for commercial telephony is based on a non-uniform quantization [1] of the signal amplitude according to a logarithmic scale rather than linear scale. Other waveform coders in use are Adaptive Differential Pulse-Code Modulation (ADPCM) and Adaptive Delta Modulation (ADM) [2,3]. All of these coders have three steps in common namely: Sampling the analog signal, followed by the quantization process of the digital signal, and finally coding the quantized speech sample using binary number representation. The performance of these coders is well documented elsewhere [4,5].

1.3 CONCEPT OF SPEECH VOCODER

The minimum bit rate (maximum speech compression), that one can achieve from PCM, ADPCM and ADM, is the sampling rate itself. For example, a speech compression system based on a waveform coder (operating at a rate of 10 Kbps), with acceptable speech quality, can only store 100 seconds of speech in $10^6$ bits of memory. Thus, if one wishes to further compress the speech below the sampling rate, then a different approach is required.

One approach is to determine the physical characteristics of the speech production, perception and language and
then to incorporate these characteristics into the transmission system. Ideally, these characteristics are described by few independent parameters, and these parameters serve as the information bearing signals. Transmission systems in which an conscious effort is made to exploit these factors are generally referred to as speech vocoders.

Vocoders can operate in the low bit rate region by estimating, digitizing, and communicating, the parameters (describing vocal tract & excitation) of the speech production mechanism. The receiver then reconstructs the original speech signal from these parameters. Since this approach is based on acoustical theory of speech production, it is appropriate to outline the essential features of such a theory before proceeding to a description of the speech vocoders.

1.3.1 *Speech Physiology*

The acoustic speech waveform is an acoustic pressure wave which originates from the voluntary physiological movements of the major parts of anatomical structure (i.e. chest cavity, throat, mouth, nose etc.) involved in speech generation.

In the normal speech production, the chest cavity expands and contracts, thus forcing air from the lungs out
through trachea past the glottis (the opening between the vocal cords is called the glottis). If the vocal cords are tensed, as for the voiced sounds (like vowels), they will vibrate, modulating the air into discrete puffs or pulses. If the vocal cords are spread apart, the air stream passes through the glottis and is unaffected. Finally, the air stream passes through the pharynx cavity (also known as vocal tract), and depending on the position of the trap door velum, is expelled either through the mouth cavity or through the nasal cavity or both and perceived as speech.

The vocal tract is a nonuniform acoustic tube which extends from the glottis to the lips and is about 17 cm long for an average adult male. The vocal tract varies in shape and size as a function of time. Its non-uniform cross-sectional area depends strongly on the position of the articulators (e.g. lips, tongue, jaw, velum etc.) and varies from 0 cm$^2$ at the closure to about 20 cm$^2$. The vocal tract has certain normal resonant modes of vibration called formants, that depend heavily on the exact position of the articulators.
1.3.2 **Linear Prediction (LPC) Model**

Many models have been proposed to describe the complicated process of speech production. None of these models, alone, can account for all of the observed characteristics of speech. However, for convenience, it is desired to have models that are linear as well as time-invariant. Since speech signal is a non-stationary, quasi-periodic waveform, speech models are derived under the following simplifying assumptions:

1. The vocal tract system and the source of excitation are independent; thus the vocal tract can be excited by any of the possible sources of excitation.

2. The characteristics of speech are time-invariant over short segments of time (approximately 20-30 msecs.). This enables the determination of linear time-invariant models for short-segments of speech.

Linear prediction analysis of speech initiated by Atal [6,7,8], has proven quite popular and successful for use in speech bandwidth compression. In this method, speech is modeled as the output of an all-pole filter $H(z)$ that is excited by a sequence of impulses separated by the pitch period for voiced sounds, or pseudo-random noise for unvoiced sounds. These assumptions imply that within a frame.

---

2 A frame represents a short segment of the speech signal (typically between 20 msecs and 30 msecs).
of speech the output speech sequence is given by:

\[ s(n) = - \sum_{i=1}^{p} a_i s(n-i) + u(n) \quad (1.1) \]

where, 'p' is the order of the filter, \( u(n) \) is the appropriate input excitation, and the \( a_i \)'s are the filter coefficients characterising the filter. Fig. 1.1 illustrates the frequency-domain, as well as the equivalent time-domain model of LPC speech production.

Since speech is non-stationary, quasi-periodic in nature, it is necessary that the filter coefficients be periodically updated during successive frames of speech.

To reconstruct speech using this technique requires a priori knowledge of the filter coefficients, the pitch, and the gain of the filter, extracted from each speech frame.

1.4 **SPEECH VOCODERS USING LPC MODEL**

Linear prediction approach leads to two different techniques of data compression, namely: vocoder based on scalar quantization (LPC vocoder) and vocoder based on vector quantization (LPC-VQ vocoder).
(a) Time Domain Representation

\[ \sum_{i=1}^{p} a_i s(n-i) \]

TIME-VARYING FILTER COEFFICIENTS

(b) Frequency Domain Representation

\[ H(z) = \frac{1}{1 + \sum_{i=1}^{p} a_i z^{-i}} \]

Fig. 1.1 Linear Prediction Model of Speech
1.4.1 Scalar Quantization

In an LPC vocoder, four main parameters namely: pitch, gain, voiced/unvoiced decision, and the filter coefficients are extracted from each frame of speech. Traditionally, these parameters have been estimated using the so-called autocorrelation estimation method [8]. These parameters are digitized by scalar quantization techniques [9] in which each parameter is quantized and coded separately.

The efficient transmission of speech over LPC vocoders suggest that the four main parameters contain sufficient information to specify the acoustic speech signal accurately. For example, a frame of speech (typically 200 samples) can be accurately represented by 11 measurements: 8 filter coefficients (using 8th order model); the pitch period of the voice in voice sounds; the gain of the filter; and a binary indication concerning whether the sound is voiced or unvoiced.

The main advantage of LPC vocoders is the very low rate (typically 4600 - 9200 bps) at which highly intelligible speech is transmitted. For example, a speech compression system based on the principle of an LPC coder (operating at a rate of 8 Kbps), with acceptable speech quality, can store the speech data in a much more efficient way.
i.e. 1/8 of a byte is needed per speech sample) as compared to commercial PCM system where each sample will require one full byte for storage.

Traditional LPC vocoders have already reached their theoretical performance limits [9], in regard to their ability to store or transmit speech efficiently. To compress the speech signal further, far below the current limits offered by the scalar quantizers, without losing the information content, a different type of Vocoder is developed. This works on the principle of Vector Quantization, in which the speech parameters are block quantized; in contrast, for scalar quantization, each parameter is individually quantized to one of the allowed levels.

1.4.2 Vector Quantization

To illustrate the concept of Vector Quantization, we shall consider an example of a Post & Telegraph office, which has a directory or codebook, containing several standard telegram messages with an index attached to each of them. To transmit any one of these messages, only the index corresponding to the telegram is sent across the channel. The receiving station uses this index to retrieve the message content of the telegram from the reproduction codebook (same as the one at the transmitter). Hence, one can note
that data compression has been achieved by mapping the whole block of information to an index number.

In a parallel manner, one can think of a directory or codebook made up of 'N' distinct speech sounds, and each speech sound is represented by an LPC vector (filter coefficients grouped together). Here, each input speech vector is mapped into one of the 'N' possible distinct vectors, and the index corresponding to the best matched vector is transmitted across the channel. The receiver uses this index to retrieve the best matched vector from the reproduction codebook, and next, reconstructs (synthesizes) the speech from the reproduction vector. By using the above approach, a frame of speech (typically 200 samples) can now be represented by just '4' measurements (LPC vector, pitch, gain & V/UV) as against '11' for conventional LPC vocoder, thus achieving a reduction by a factor of 3.

Using vector quantization technique, one can achieve a low bit rate of 1000 bps or even lower without losing intelligibility of speech. A Vector Quantizer operating at 1000 bps (with sampling frequency as 8 kHz), can store information content of about 64 samples in just one single byte. In other words, one can perceive this as reduction in memory space, by a factor of 1/64, as compared to commercial PCM systems.
The main thrust of this thesis is towards the development of various techniques (based on vector quantization) that achieve significant data compression, and towards the development of tree-search structures that efficiently implement those techniques.

1.5 Review of Relevant Literature

Some work has been done over the past several years on the subject of speech compression using vector quantization techniques. Most closely related to this study is the pioneering work by Gray et al. [24]. They proposed a new design for a vector quantizer based on a long training sequence of speech data. The codebook for the quantizer was generated through an iterative process much like a clustering algorithm involving a large number of spectral comparisons.

Rebolledo et al. [28] applied the above technique to speech-like waveforms and used a codebook consisting of vectors in the form of reflection coefficients. They tested the codebook on several speakers and claimed reasonably good results. Wong et al. [34] expanded the work of Gray by doing rigorous comparisons between scalar quantization and vector quantization techniques. The results reported by Wong are very encouraging and fascinating.
Recently a vector quantization algorithm was developed [35] and demonstrated to be presently the most efficient approach to the speech coding and compression problem. Compared to the existing LPC vocoder with scalar quantization which operates at a rate of about 4600 bits per second, the new method has a typical bit rate of about 800 bits per second.

In general, all these quantizers seem to work in an ad hoc fashion (i.e. their performance varies from speaker to speaker). Also, not much effort has been expended on the development of tree-search structures which could effectively reduce the extensive arithmetic computations that limit the use of these quantizers for practical implementation. It is therefore appropriate and necessary to develop some alternative designs, based upon vector quantization properties, that will yield quantizers which are text and speaker independent. Also, tree-search structures for quantizers are to be developed which will significantly reduce the processing time, thereby making real time implementation feasible.
1.6 RESEARCH OBJECTIVES

Based upon the problem stated above, the main objectives of this research are as follows:

1. To form a high quality speech synthesis model.

2. To review and demonstrate the working of a linear predictive voice-coder, operating at the rate of 9200 kbps.

3. To introduce various new schemes of vector quantization, so as to reduce the bit rate, while preserving the message content of the speech signal.

4. Finally, to examine the possibility of sequential search of codebooks, using some kind of tree-like structure, so as to increase the speed of implementation.

1.7 THESIS OUTLINE

In Chapter 2, the linear prediction of speech is reviewed in detail. Also the various models of the acoustic tube (vocal tract) are briefly mentioned.

Chapter 3, describes the operation of LPC vocoder based upon parametric representation of waveform. In this chapter, we also establish a typical bit rate of an LPC vocoder.
Chapter 4, discusses a new approach of achieving compression in speech signals, known as Vector Quantization. Further, a new approach of generating a codebook from phoneme's of English language is outlined.

Chapter 5, discusses in detail the generation of codebooks, using long training sequence of speech data. Also this chapter throws some light, on the topic of efficient implementation of various codebooks.

Chapter 6, summarizes the main contribution and conclusions of this work.

Finally, in Chapter 7, future research ideas and the area where VQ techniques can be applied, are considered.

Appendix A gives a brief description of the Covariance & Autocorrelation method of linear prediction analysis.

In Appendix B, illustrations of the speech waveforms (original, synthesized, scalar quantized, and vector quantized) for two sentences are included.

In Appendix C, a listing of the computer programs written in FORTRAN V is provided.
Chapter II

ANALYSIS AND SYNTHESIS OF SPEECH USING LINEAR PREDICTION

2.1 INTRODUCTION

The technique of linear prediction [6,10,11] has been one of the most significant developments in providing an efficient representation of speech signals. In general, speech synthesized by the linear prediction method is of high quality and suffers very little degradation in the analysis-synthesis process.

In the linear prediction method, the filtering properties of the vocal tract, the radiation at the mouth opening, and the glottal wave are represented by an all pole digital filter, typically with 12 poles at a sampling frequency of 10 KHz. The all-pole model of the vocal tract is quite accurate for the vowel and vowel-like sounds. However, the model does not provide an accurate representation of the nasal sounds in the speech signal. The model requires the inclusion of "zeros" to account for nasal sounds. These zeros can arise if the nasal tract is coupled to the main vocal tract through the velum opening - as is the case for nasal consonants and nasalized vowels. Fortunately, "zeros"
in the vocal tract transfer function can be approximated by including additional poles in the linear model. A further advantage of the LPC model is that the parameters of the model are easily derived from the digitized speech samples, using the minimum squared error criterion.

2.2 SPEECH PRODUCTION MODELS

Based on the two main assumptions described in sec. 1.3.2, linear speech production model (shown in Fig. 2.1) was developed by Pant [8,12], in the late 50's, which was later elaborated by Flanagan [12,18].

The glottal volume velocity waveform \( u_G(t) \) is modeled as the output of a two-pole low pass filter with an estimated cutoff at about 100 Hz. The input to this filter is an impulse train \( u(t) \) with period 'P' for voiced sounds and flat spectrum random noise for unvoiced sounds. The vocal tract model results in an all pole model, consisting of a cascade of small number of two-pole resonators. Each resonance is defined as a formant with corresponding centre frequency and bandwidth. The volume velocity waveform at the lips \( u_L(t) \) is finally transformed into an acoustic pressure waveform some distance away from the lips, representing the speech waveform \( s(t) \).
Fig. 2.1 Linear Speech Production Model
Using z-transform terminology, speech production model shown in Fig. 2.1, can be described as:

\[ S(z) = U(z) \cdot G(z) \cdot V(z) \cdot L(z) \]

where, \[ S(z) \leftrightarrow s(nT) = s(t) \quad t = nT \]  \hspace{1cm} \text{---(2.1)}

The driving function to the glottal shaping model is \[ U(z) \leftrightarrow u(n), \] a train of unit samples, spaced by the pitch period \( P = IT \), where, \( I \) is a positive integer, i.e.

\[ U(z) = \sum_{n=0}^{\infty} (z^{-I})^n = \frac{1}{1-z^{-1}} \quad \text{for } |z| > 1 \]

The glottal shaping model \( G(z) \) is of the form,

\[ G(z) = \frac{1}{(1 - e^{-cT} z^{-1})^2} \]  \hspace{1cm} \text{---(2.2)}

and the lip radiation model \( L(z) \) is given by:

\[ L(z) = 1 - z^{-1} \]  \hspace{1cm} \text{---(2.3)}

The vocal tract \( V(z) \) is given by an all pole vocal tract model \( V(z) \), consisting of \( K \) formants \([8]\), described by

\[ V(z) = \frac{1}{\left| \sum_{i=1}^{K} \frac{-c_i T}{1 - 2e^{-i T} \cos (b_i T) z^{-1} + e^{-2c_i T} z^{-2}} \right|} \]  \hspace{1cm} \text{---(2.4)}
2.3 **Linear Prediction Model of Speech Production**

Atal & Hanauer [6] were the first to directly apply linear prediction technique to speech analysis and synthesis. They suggested that the glottal model and the vocal tract model can be represented by an all pole, time varying digital filter.

Rewriting eq. 2.1, we have:

\[
\frac{S(z)}{U(z)} = \frac{(1 - z^{-1})}{(1 - e^{-cT} z^{-1}) \left\{ \prod_{i=1}^{K} \left[ 1 - 2e^{-c_i T} \cos (b_i T) z^{-1} + e^{-2c_i T} z^{-2} \right] \right\}}
\]

---(2.5)

The numerator term \((1 - z^{-1})\) can be cancelled by one of the denominator term \((1 - \exp(-c* T) z^{-1})\) since \('c*T' is generally smaller than unity [8]. A further simplification of the discrete model is given as:

\[
\frac{S(z)}{U(z)} = \frac{G}{A(z)} \quad ---(2.6)
\]

where, \(G/A(z)\) is an all-pole filter. The transfer function of this linear filter can be shown as:

\[
H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 + \sum_{i=1}^{P} a_i z^{-i}} \quad ---(2.7)
\]
where, 'G' is the gain factor, \( a_i \)'s are the linear prediction coefficients (\( a_0 \) is usually normalized to unity), and 'P' is the order of the filter.

2.4 **Linear Prediction Analysis**

Linear prediction analysis is used to evaluate the following four control parameters of the linear prediction model from the natural speech signal.

1. \( a_i ; i = 1,2,3,\ldots, p \)
2. Gain factor G.
3. Voiced/Unvoiced (V/UV) decision.
4. Pitch period (P) for voiced speech.

Next we will see how these parameters can be determined directly from the speech samples.

2.4.1 **Linear Prediction Coefficients**

The all pole model of Eq. (2.7) can be characterized by a difference equation of the form:

\[
    s(n) = - \sum_{i=1}^{p} a_i s(n-i) + G u(n) \quad (a_0 = 1) \tag{2.8}
\]

The excitation function \( u(n) \) is zero except for one sample at the beginning of every pitch period for voiced sounds. Thus:

\[
    s(n) = - \sum_{i=1}^{p} a_i s(n-i) ; \quad n > 0 \tag{2.9}
\]
Hence for \( n > 0 \), the speech sample \( s(n) \) is a linear combination of the previous \('p'\) samples. For \( n = 0 \), equation 2.9 will not be satisfied exactly. Since the model is not perfect in this sense, the linearly predicted sample will only be a close approximation to \( s(n) \). Let us denote this predicted sample as \( \hat{s}(n) \), so that:

\[
\hat{s}(n) = - \sum_{i=1}^{P} a_i s(n-i)
\]  

\[\text{---(2.10)}\]

Rewriting equation 2.8,

\[
s(n) + \sum_{i=1}^{P} a_i s(n-i) = G u(n)
\]  

\[\text{---(2.11)}\]

\[
s(n) - \hat{s}(n) = G u(n) = e(n)
\]  

\[\text{---(2.12)}\]

The driving function term \( G u(n) \) can now be interpreted as the prediction error, \( e(n) \), between the actual speech sample \( s(n) \) and the predicted sample \( \hat{s}(n) \). The \( a_i \)'s are chosen so as to minimize the total squared error (of the frame under consideration), given by:

\[
E = \sum_{n=n_0}^{n_1} e^2(n) 
\]  

\[\text{---(2.13a)}\]

\[
= \sum_{n=n_0}^{n_1} \left| s(n) + \sum_{i=1}^{P} a_i s(n-i) \right|^2
\]  

\[\text{---(2.13b)}\]
Here \((n_0 \text{ to } n_1)\) defines the index error over which the error minimization occurs. Now equation \((2.13b)\) can be written as:

\[
E = \sum_{n=n_0}^{n_1} \left| \sum_{i=0}^{p} \sum_{j=0}^{p} a_i s(n-i) s(n-j) a_j \right| \quad ---(2.14)
\]

Defining

\[
C_{ij} = \sum_{n=n_0}^{n_1} s(n-i) s(n-j) \quad ---(2.15)
\]

The total squared error 'E' can then be equivalently written as:

\[
E = \sum_{i=0}^{p} \sum_{j=0}^{p} a_i C_{ij} a_j \quad ---(2.16)
\]

Minimization of 'E' is obtained by setting the partial derivative of 'E' w.r.t. \(a_k\) \((k=1,2,3,\ldots, p)\) to zero and solving. Thus, from \((eq. 2.16)\),

\[
\frac{\partial E}{\partial a_k} = 0 = 2 \sum_{i=0}^{p} a_i C_{ik}
\]

Since \(a_0 = 1\)

\[
\sum_{i=1}^{p} a_i C_{ik} = -C_{0k}; \quad k = 1,2,3,\ldots, p \quad ---(2.17)
\]
Eq. (2.17) leads to a set of \( p \) linear algebraic equations. In general, there are a variety of techniques that can be applied to solve a system of \( p \) linear equations in \( p \) unknowns.

In order to effectively implement a linear predictive analysis system, it is necessary to solve the linear equations in an efficient manner. There are two popular methods which lead to an efficient solution of eq. (2.17). These are referred to as covariance method, developed by Atal & Hanauer [6] and the auto-correlation method, developed by Itakura & Saito [13,14], and Markel & Gray [8,15]. The description of these two methods is given in the Appendix A.

2.4.2 Pitch And \( V/UV \) Analysis

Pitch period estimation (or equivalently, fundamental frequency estimation) is one of the most important problems in speech processing [18]. Because of its importance, many solutions to this problem have been proposed [8,16,17,19]. All of these proposed schemes have their limitations, and hence it is safe to conclude that no presently available pitch detection scheme can be expected to give perfectly satisfactory results across a wide range of speakers.

In this section, a particular pitch detection scheme first proposed by Sondhi [17] and later on modified by Ha-
biner and Schaffer [18] will be discussed. This technique uses short-time autocorrelation function to estimate the pitch period.

2.4.2.1 Pitch Period Estimation Using The Autocorrelation Function

In this method, the pitch period is determined by picking the largest peak in the autocorrelation function. In some cases the largest peak can be attributed to the damped oscillations of the vocal tract response, which are responsible for the shape of each period of the speech wave. This problem of the extraneous peaks in the autocorrelation function can be greatly alleviated by centre clipping [17] the speech signal, prior to computing the autocorrelation function. The centre clipped speech signal is obtained by a non-linear transformation. The operation of the centre clipper is depicted in Fig. 3.2.2. For this segment of speech, the maximum amplitude, \( A_{\text{max}} \), is found and the clipping level, \( C_L \), is set equal to a fixed percentage of \( A_{\text{max}} \) (Sondhi used 30%). It can be seen that the output speech sample (centre clipped speech) is equal to the input minus the clipping level. For samples below the clipping level, the output is zero. This operation greatly eliminates the spurious responses due to the vocal tract.

---

3 This figure is taken from the text book by Rahiner & Schafer [18].
Fig. 2.2 An Example Depicting The Operation Of Centre Clipping.

( After Rabiner & Schafer )
Rabiner & Schafer [18] developed an algorithm for pitch detection, using the principle of centre clipping and the short-time autocorrelation function. In our study, we have slightly modified the said algorithm, to reduce the number of computations involved and also to improve the accuracy of the algorithm. The details of the algorithm are depicted in Fig. 2.3 and the steps involved are listed below:

1. The analog speech sample is sampled at a rate of 10 KHz.

2. Segments of length 30 msec (300 samples) are selected at 10 msec intervals. Thus, the segments overlap by 20 msec.

3. Each speech segment is filtered using a third order lowpass filter, whose cutoff frequency is 900 Hz.

4. The average absolute magnitude is computed for the first and the second 100 samples of the speech segment. The minimum of the two average magnitudes, denoted by $B_{\text{min}}$, is compared to a threshold determined by measuring the average absolute magnitude level for 50 msec (500 samples) of background noise. If the value of $B_{\text{min}}$ is above the threshold, implying that the segment is speech and not noise, then the algorithm proceeds to the next stage, otherwise the segment is classed as silence and no further action is taken.
Fig. 2.3 Block Diagram Of The Pitch Algorithm
5. This step deals with the voiced/unvoiced decision, which is based on the density of zero crossings in the given speech segment. A low zero crossing rate would imply that the speech segment is voiced, and one proceeds further to determine the pitch period of the segment. On the other hand, a high zero crossing rate would mean that the segment is unvoiced and no further action is taken on this segment.

6. The clipping level is determined on a fixed percentage (e.g. 75%) of the minimum of the maximum absolute values in the first and last 100 samples of the speech segment.

7. Using this clipping level, the speech signal is processed by a 3-level centre clipper and the autocorrelation function is computed over a range spanning the expected range of pitch periods (4.0 - 15 msecs).

8. The largest peak of the autocorrelation function is located. Next, a confirmatory test (whether the segment is voiced or unvoiced, for sure) is carried out by comparing the peak values to a fixed threshold (e.g. 20% of the autocorrelation function with zero lag). If the peak falls below the threshold, the segment is classed as unvoiced and if it is above, then the pitch period is defined as the location of the largest peak.
This algorithm was implemented on a digital computer. Experimentally, it was found to perform reasonably well, over a wide spectrum of speakers.

2.4.3 Gain Computation

It is possible to relate the gain factor, \( G \), to the excitation signal and the prediction error by referring back to eq. 2.8 & eq. 2.12

\[
e(n) = s(n) + \sum_{i=1}^{p} a_i s(n-i) = G u(n) \quad (2.18)
\]

From eq. 2.18, it is clear that the error signal \( e(n) \) is proportional to the input excitation source \( u(n) \). Therefore we postulate that the energy in the excitation source is equal to the energy in the error signal \([18]\), which is represented by \( E \) in eq. 2.13a. Hence we have:

\[
E = \sum_{n=0}^{N_l-1} e^2(n) = G^2 \sum_{n=0}^{N_l-1} u^2(n) \quad (2.19)
\]

For voiced speech, value of excitation function \( u(n) \) chosen, is given below:

\[
u(n) = \begin{cases} 
1 & \text{for } n = 0 \\
0 & \text{for } n \neq 0
\end{cases} \quad (2.20)
\]
Assuming 'NL' data samples per analysis frame, the gain factor $G$, for voiced speech, can be defined as:

$$G = \sqrt{E} = \left| \sum_{n=0}^{NL-1} e^2(n) \right|^k
---(2.21)$$

Now for unvoiced speech, the excitation function is represented by a pseudo-random number generator, which has zero mean and unit variance. Matching energies over NL samples, the gain factor 'G' per sample for unvoice speech is computed as:

$$G^2 \sum_{n=0}^{NL-1} v^2(n) = \sum_{n=0}^{NL-1} e^2(n)
---(2.22)$$

Dividing both sides by 'NL', we have:

$$\frac{G^2}{NL} \left| \sum_{n=0}^{NL-1} v^2(n) \right| = \frac{1}{NL} \left| \sum_{n=0}^{NL-1} e^2(n) \right|
---(2.23)$$

But,

$$\frac{1}{NL} \sum_{n=0}^{NL-1} v^2(n) = 1
\text{ Variance of the random numbers is given as unity}$$

Hence,

$$G = \left| \frac{1}{NL} \sum_{n=0}^{NL-1} e^2(n) \right|^k
= \left| \frac{E}{N} \right|^k
---(2.24)$$
The fact that the quantity \( R \) (total squared error) is a by-product of linear prediction analysis, makes the computation of gain very efficient by using the method outlined above.

2.4.4 Linear Prediction Synthesis

The speech signal is synthesized by means of the same parametric representation as was used in the analysis. Fig. 2.4 shows the linear prediction synthesizer after Atal & Banauer [6]. The control parameters supplied to the synthesizer are the voiced-unvoiced decision, pitch period, gain factor, and the 'p' predictor coefficients. The impulse generator produces an unit impulse at the start of each pitch period. Pseudo-random number generator produces uniformly distributed random samples which have zero mean and unit variance. The selection between the impulse generator and the pseudo-random generator is made by the voiced-unvoiced switch. The amplitude of the excitation signal is adjusted by the amplifier, with gain 'G'. Next, past 'p' linearly predicted values of the speech signal are combined with the excitation signal \( u(n) \), to form the nth sample of the synthesized speech signal. Finally, the speech samples are passed through a D/A converter to obtain the continuous speech waveform \( s(t) \).
Fig. 2.4 Linear Prediction Synthesizer

(After Atal & Hanauer)
Since speech is considered stationary only for a short period of time, hence, the synthesizer control parameters are updated at the beginning of every pitch period. This type of synthesis is called pitch synchronous synthesis, whereas if the control parameters are updated once every segment or frame of speech, the process is called pitch asynchronous synthesis [6,8,18,20]. Pitch asynchronous synthesis requires interpolation of the control parameters which is not so straightforward [6]. Also the stability of the all pole filter cannot be guaranteed by obtaining the predictor coefficients through direct linear interpolation.

Pitch synchronous synthesis in the present work was performed using a variable frame length of 'P' samples for voiced speech (P is the pitch period), and a fixed frame length of 100 samples for unvoiced speech. The operation of synthesis can be shown by these equations:

1. For voiced speech (0 ≤ n ≤ NL=1; NL=P )

\[
\hat{s}(n) = \sum_{i=1}^{P} a_i \hat{s}(n-i) + G u(n) \quad ---(2.25)
\]

\[
\begin{cases} 
  u(n) = 1 \text{ for } n = 0, p, 2p \ldots \\
  u(n) = 0 \text{ for } n \neq 0, p, 2p \ldots 
\end{cases}
\]

2. For unvoiced speech (0 ≤ n ≤ NL=1; NL=100 )

\[
\hat{s}(n) = \sum_{i=1}^{P} a_i \hat{s}(n-i) + G v(n) \quad ---(2.26)
\]

\[
\begin{cases} 
\text{Variance of } v(n) \text{ is} \\
\text{given as unity}
\end{cases}
\]
2.5 OTHER SPEECH PARAMETERS

In this section, a brief description of various other speech parameters obtained by forming different models of the vocal tract tube, is given.

2.5.1 Reflection Coefficients

The modeling of the vocal tract tube as a nonuniform acoustic tube formed by concatenating 'p' uniform Cylindrical sections of equal length [8,18], leads to a set of parameters called the reflection coefficients. The reflection coefficients \( K_i : i=1,2,3\ldots, p \) are related to the linear prediction coefficients \( a_i : i=1,2,3\ldots, p \) by the following relations [8].

\[
K_m = a^{(m)}_m
\]

\[
a^{(m-1)}_i = \frac{a^{(m)}_i - K^2_m a^{(m)}_{m-1}}{1 - K^2_m} \quad \text{---(2.28)}
\]

for, \( m = p, p-1, \ldots, 1 \)

\( i = 0, 1, 2, \ldots, m-1 \)

Reflection coefficients play an important role in the analysis and synthesis of speech. They act as an important tool for verifying the necessary and sufficient conditions.
for the stability of the synthesis model (all pole filter). The stability conditions are given as:

$$|K_m| < 1 \text{ for, } m = 1, 2, 3, \ldots, p \quad ---(2.29)$$

Also, speech can be reconstructed directly from the reflection coefficients using synthesis structures of lattice form [21]. It has been shown that speech synthesized directly from the reflection coefficients, is as good in quality, as the one obtained from linear prediction [8].

2.5.2 **Log Area Ratio Coefficients**

The acoustic tube model yields another set of parameters that are related to the ratios of the areas of the cylindrical sections. These are called the Log Area Ratio Coefficients ($g_i; i=1,2,3\ldots, p$) and these may be derived from the reflection coefficients as follows [18]

$$g_i = \log \left| \frac{A_i + 1}{A_i} \right|$$

$$= \log \left| \frac{1 - K_i}{1 + K_i} \right| \quad ---(2.30)$$

1. ≤ $i$ ≤ p

Where ($A_i; i=1,2,3\ldots, p$) are the areas of the $'m'$ cylindrical sections of the model.
2.5.3 **Cepstrum Coefficients**

The term "Cepstrum" was introduced by Bogert et al [18] and has come to be accepted terminology for the inverse Fourier transform of the logarithm of the magnitude spectrum of a signal. One can obtain the Cepstrum coefficients \( C_i ; \ i=1,2,3,\ldots,\ p \) by computing the cepstrum of the inverse filter \( A(z) \), where inverse filter is characterized by:

\[
A(z) = 1 + \sum_{i=1}^{p} a_i z^{-i}
\]  

---(2.31)

These coefficients are related to the linear prediction coefficients \( a_i \)'s [18] as follows:

\[
C_i = a_i - \sum_{j=1}^{i-1} \frac{i-1}{i} C_j a_{i-j}
\]  

---(2.32)

\[
C_i = a_i
\]

where, \( i = 2, 3, 4, \ldots p \).

2.6 **DATA COLLECTION & PREPROCESSING**

Different trained speakers spoke into a high-fidelity microphone and the speech was recorded on a high quality tape recorder in a noise free environment.

Recorded analog speech was passed through a bandpass filter with cutoff frequencies at 200 Hz and 3.2 KHz. The filtered speech was then digitized at 10 KHz sampling rate
by using TUSTIN X-1500 A/D converter having 14 bit resolution. The digitized speech was stored in the disk storage medium of the Data General NOVA 840 minicomputer. This minicomputer is also equipped with a tape unit device for additional storage requirements.

A further preprocessing step prior to the parameter extraction, involved pre-emphasis of digitized speech data. This was accomplished by passing the digitized speech signal through a single zero filter \((1 - \mu z^{-1}; 0.9 < \mu < 1.0)\) [8]. This procedure was used to obtain a more accurate estimate of the spectral characteristics of the vocal tract tube.

2.7 ANALYSIS/SYNTHESIS DETAILS

Since speech is stationary only for a short interval of time, analysis of speech was carried out by using small segments of speech. Each segment or frame of speech was made up of 200 samples (duration of 20 msecs).

For the autocorrelation method of linear prediction, proper windowing is quite important. The overall effect of a good window is to produce more distinct resonance structure, narrow bandwidth estimation, and above all, more realistic sounding synthetic speech. Hence, each frame of
speech data was multiplied by a Hamming window.

Linear prediction coefficients were extracted from each frame of speech at a rate of 100 times per second (frame moved in steps of 100 samples). Pitch synchronous analysis/synthesis being less complex than pitch asynchronous analysis/synthesis [6,8], was used.

2.8 COMPUTER SIMULATION & RESULTS

Various sentences spoken by different male speakers were synthesized on a digital computer (NOVA 840). These sentences represented all typical speech sounds (described in detail in sec. 4.5.2)

Results of the informal perceptual listening tests can be summed up as follows:

1. The quality of the synthesized speech for p=12 was found to be almost as good as the original speech. Increasing the order of 'p' beyond 12 did not show any significant improvement in the quality of the synthesized speech. Hence, one can reasonably assume p=12 as quite sufficient to provide an adequate representation of the speech signal.

* Traditionally, Hamming window has been used, however, other windows (such as the Kaiser window and the Hanning window) can also give reasonably good results.
2. There was slight degradation in speech quality at p=8. This was especially true for nasal and plosive sounds.

3. For 'p' as low as 2, the synthetic speech was intelligible although poor in quality.

4. The use of Hamming window in analysis, made the synthetic speech sound better (pleasant) as compared to the speech obtained by using an implicit rectangular window.
Chapter III
LINEAR PREDICTIVE Vocoder

3.1 INTRODUCTION

One of the most important applications of linear predictive analysis has been the area of low bit rate encoding of speech, for transmission (the LPC coder) and for storage (for computer voice response systems). The term Vocoder is a contraction of voice-coder as defined by Dudley in 1939 [22]. Since the invention of the vocoder, great effort and time have been expended on improving the quality of speech from the various voice-coding devices, while maintaining the real-time processing capabilities. An excellent summary of that effort up to 1966 is presented by Schroeder [23]. Among the various existing voice-coding systems, the linear predictive voice-coder or vocoder has invited the most interest due to the ease in implementation and comparatively good speech quality and intelligibility. Fig. 3.1 shows a block diagram of an LPC Vocoder.

The main function of the analyser is to extract a set of parameters which describe a segment of the given speech data. These include the linear predictor coefficients, the
Fig. 3.1. Block Diagram of an LPC Vocoder
gain, the voiced/unvoiced decision and the pitch for the voiced segments. These parameters are then quantized, and encoded for transmission and/or storage.

The decoding section decodes these transmitted parameters back into the analysis parameters. The synthesizer then reconstructs the synthetic speech waveform, \( s(n) \), from these analysis parameters.

3.2 TRANSMITTER SECTION

The speech waveform is passed through an analog band-pass filter with cutoff frequencies at 200 Hz and 3.2 KHz. The filtered speech is then sampled at 10 KHz using a 14 bit analog-to-digital converter. The speech samples are then applied to a pre-emphasis filter, whose transfer function \( P(z) \) is given by:

\[
P(z) = 1 - \mu z^{-1}
\]

---(3.1)

For speech analysis the value of \( \mu \) is not critical, and values in the range from 0.9 to 1.0 yield roughly equivalent results [8]. Throughout this study, the value of \( \mu \) is taken as 0.95.

Linear predictive analysis (discussed in detail in sec. 2.4) is performed on the speech segment or frame of
length 'NL', every $1/F_r$ units of time, where $F_r$ is the analysis frame rate, and is defined as:

$$F_r = \frac{\text{Sampling rate}}{NL - N_1}$$

where, 'N_1' is the number of samples by which successive frames of speech overlap.

In summary, the linear prediction analysis performed on a frame of speech, gives the following parameters as the output:

1. Predictor coefficients or filter coefficients ($a_i : i=1,2,3,\ldots, p$), with $a_0$ equal to 1;

2. Gain factor term, 'G';

3. Voiced/unvoiced decision;

4. Pitch period or fundamental frequency, if the frame is classified as voiced;

The above information is then used at the receiver, to reconstruct the speech waveform.
3.2.1 **Scalar Quantization of Parameters**

The transmission or storage of the LPC encoded speech requires that the LPC analysis output parameters be properly and efficiently quantized. The excitation parameters, comprising essentially of a voicing decision and a pitch period, are quantized separately from other parameters. The gain term, is also quantized independently of the filter coefficients.

Aside from the pitch and gain parameters, the other parameters to be quantized are the predictor coefficients of the all-pole model. In general, it has been observed [6] that the predictor coefficients are quite sensitive to the quantization process. This is due to the fact that small errors in the predictor coefficients can result in large errors in the pole location, often rendering the model unstable.

Now, the transformation of the linear predictor coefficients to another set of coefficients, known as Reflection coefficients, ensures the stability of the model [7,15]. In general, the reflection coefficients are the best set to use as transmission parameters [7], since they exhibit the following properties:

1. The reflection coefficients given by \( \{ \alpha_i, i=1,2,3,\ldots, p \} \), are distributed over the interval \([-1,1]\).
2. \(|K_m| < 1, \ (m = 1, 2, \ldots, p)^4\) ensures stability of the synthesis filter; here 'p' is the order of the linear predictor.

In this study, the 'p' reflection coefficients were quantized separately, using a 10-bit quantizer with uniform step size.

3.3 RECEIVER SECTION

At the receiver, the decoder simply transfers the transmission parameters back into the quantized analysis parameters. These parameters uniquely and completely define the synthetic speech of one frame length, obtained by the synthesis model.

3.3.1 Synthesis filter and De-emphasis

The synthesis filter should mathematically implement the reciprocal of the inverse filter \(A(z)\). Implementation of \(1/A(z)\) is quite simple; however, the transmission parameters are the reflection coefficients and not the inverse filter coefficients \(\{a; \ i=1, 2, 3, \ldots, p\}\). Therefore, the reflection coefficients \(\{K; \ i=1, 2, 3, \ldots, p\}\), are transformed back into filter coefficients \(\{a; \ 's\}\), using a recursive procedure [7, 8]. Next, the speech is reconstructed from these parameters using linear prediction synthesis method (as discussed
in sec 2.4.4). Finally, the operation of de-emphasis is performed on the synthesized speech data, by implementing the reciprocal of the pre-emphasis filter, defined as:

\[
\frac{1}{P(z)} = \frac{1}{1 - \mu z^{-1}} \tag{3.2}
\]

The de-emphasized synthetic speech sample, denoted by \( \hat{s}(n) \) at an instant 'n' is computed as:

\[
\hat{s}(n) = s(n) + \mu \hat{s}(n-1) ; \hat{s}(n)=0 \text{ for } n < 0 \tag{3.3}
\]

The operation of de-emphasis is used to suppress the high frequency noise present in the synthesized speech data.

3.4 EXPERIMENTAL RESULTS

In this study, an LPC Vocoder was successfully implemented on a digital computer. The various conditions and bit allocation employed, are given below,

1. Sampling frequency \( (f_s) = 10 \) kHz;

2. Length of a frame of speech data \( (N_L) = 200 \) samples;

3. Order of the predictor \( (p) = 8; \)

4. Frame rate \( (f_r) = 100 \) frames/sec \( (N_1 = 100 \) samples)
5. Quantization of Pitch: 7 bits per frame (included one bit for voicing decision);

6. Quantization of Gain factor: 5 bits per frame;

7. Quantization of predictor coefficients: 10 bits per coefficient, hence a total of 80 bits per frame;

Now, the speech data rate of an LPC vocoder is given by,

\[ D_r = B_f \times F_r \text{ bits/sec} \]  \hspace{1cm} (3.4)

where, \( B_f \) is the total number of bits needed per analysis frame.

\( F_r \) is the frame rate.

Using the various conditions and bit allocations stated above, \( F_r = 100 \) frames per second and \( B_f = (7 + 5 + 80) = 92 \) bits per frame. The total speech data rate, therefore, is \( D_r = 92 \times 1000 = 92000 \) bits per second.

The operation of an LPC vocoder was tested on a variety of sentences, spoken by different trained speakers. The text of some of the sentences used is described below:

1. PAPA NEEDS TWO SINGERS.
2. CASE THIS BOND PLEASE.
3. MY NAME IS MILLER.
4. EACH FORCE IN ITS OWN WAY IS CREATIVE.
In informal listening tests, the quality of the synthetic speech obtained from the LPC vocoder was found to be very close to that of the original speech, for a wide range of speakers and spoken text.

The speech data rate can further be compressed from the 9200 bps to 4600 bps, without seriously affecting the synthesis quality. This is accomplished by using a uniform frame length (200 samples), with no overlap between successive frames. The frame rate is 50 frames per second (using 10KHz as sampling rate). However, there is a lower limit of data rate below which one cannot go for synthesizing speech through linear prediction using scalar quantization techniques [9]. Thus, to achieve a bit rate lower than 4600 bps without causing serious quality degradation, it is apparent that a more efficient quantization scheme is mandatory. The study is henceforth devoted to an LPC quantization technique, called Vector Quantization. This technique is capable of encoding the speech at a rate lower than 2000 bps, while maintaining the intelligibility of the speech signal.
Chapter IV

LPC VECTOR QUANTIZATION

4.1 INTRODUCTION

Although the first use of Vector Quantization techniques can be traced back to 1957, its application in the area of speech compression is recent. The historical development of this area is covered in detail elsewhere [24,25]. The goal of a system based on vector quantization is speech compression: to reduce bit rate or digital storage memory requirements, while maintaining the necessary quality of the speech data.

During the past few years several design algorithms have been developed for a variety of vector quantizers, and the performance of these codes has been studied for speech waveforms and speech-linear predictor parameter vectors [26,27,28,34]. In general, these quantizers are found to be locally optimum, i.e. they do not perform reasonably well over a wide range of speakers and text. Hence, it is appropriate and necessary to develop some alternative design that will yield quantizers which are text and speaker-independent.
In this chapter, we introduce a new technique of obtaining a vector quantizer from Phonemes of the English language. The performance of this quantizer is tested by using a variety of sentences, spoken by five trained speakers.

4.2 BLOCK QUANTIZERS

An N-level \( p \)-dimensional vector (or block) quantizer is a mapping, \( q \), that assigns to each input vector, \( \mathbf{x} = (x_1, x_2, \ldots, x_p) \), a reproduction vector, \( \hat{x} = q(\mathbf{x}) \), chosen from a finite collection of vectors, namely, \( \mathbf{A} = \{Y_j; j = 1, 2, \ldots, N\} \). The finite collection of vectors, \( \mathbf{A} \), is often called a codebook and its entry, \( Y_j \), a codeword. The quantizer \( q \) is completely described by the codebook, \( \mathbf{A} \), together with the partition, \( \mathbf{T} = \{T_j; j = 1, 2, \ldots, N\} \), of the input vector space into sets \( T_j = \{\mathbf{x} : q(\mathbf{x}) = Y_j\} \) of input vectors, mapped into the jth reproduction vector or codeword.

The above quantization methodology can easily be applied to speech coding. Earlier in section 1.3.2, it has been observed that speech is a slowly time-varying, quasi-periodic signal. This allows us to divide the speech signal, \( \{s(n); n=1, 2, \ldots, M\} \) into segments and represent each segment by a vector of finite dimension. In the LPC analysis method, the input vector \( \mathbf{x} \) is given by a set of predictor coefficients. Also, the codewords in \( \mathbf{A} \) are vectors (in the
form of predictor coefficients), describing the short time transfer characteristics of the vocal tract.

Each input speech vector is mapped to the codebook entry (codeword) index corresponding to the best matched vector. Speech compression or bit rate reduction is accomplished by using the indexes as storage or transmission parameters.

A vector quantizer can be divided into two main blocks: an encoder, which views the input vector \( \mathbf{X} \) and generates the index of the reproduction vector from the codebook, specified by \( q(\mathbf{X}) \); and a decoder, which uses this index to generate the reproduction vector \( \hat{\mathbf{X}} \). A vector quantizer can be used to communicate over a digital channel by placing the encoder at the transmitter and the decoder at the receiver and sending the index of the codeword across the channel. Fig. 4.1 depicts the operation of a vector quantizer. The operation of the encoder is more complex. In effect, it partitions the input space into a collection of cells according to the mapping given by \( q(\mathbf{X}) \). Each cell is associated with one codeword in the codebook. The operation of the decoder is simpler: it stores the codebook and looks for the reproduction vector indexed by the encoder.
Fig. 4.1 Basic Concept of Vector Quantization
4.3 **NEAREST NEIGHBOUR RULE**

Given a distortion measure \( d \) for an input-output vector pair \((X, \hat{X})\) and a codebook \( \hat{A} \), the nearest neighbour rule can be stated as follows. For each input vector \( X \), the mapping \( \varphi \) is to assign to \( X \) a \( \hat{X} = Y_j \in \hat{A} \), so that

\[
d(X, \hat{X}) = \min_{Y_1 \in \hat{A}} d(X, Y_i) \quad --- (4.1)
\]

Here \( d(X, \hat{X}) \) is the distortion caused by reproducing an input vector \( X \) by a reproduction vector \( \hat{X} \).

4.4 **DISTORTION MEASURE**

The choice of distortion measure \( d(X, \hat{X}) \) is a key component in the vector quantization technique. A number of spectral distortion techniques have been discussed by Barksdale & Gray [29] and Gray et al. [30]. All of these distortion measures, in theory, can be used in the vector quantization implementation. However, most of these distortion measures involve extensive arithmetic computations. Hence, in this study, the simplest distortion measure known as the square error distortion was used. It is defined as,

\[
d(X, \hat{X}) = ||X - \hat{X}||^2 = \sum_{i=0}^{p-1} |x_i - \hat{x}_i|^2 \quad --- (4.2)
\]
Here, the input and the reproduction vector spaces are in 'p' dimensional Euclidean space.

Now given such a distortion measure, one can quantify the performance of a vector quantizer (codebook) by an average distortion \( E \{ d(\mathbf{X}, \hat{\mathbf{X}}) \} \) (where 'E' is the expected value) between the input and the final reproduction vector. A vector quantizer is said to be optimal (good), if it yields a small average distortion. In general, the average distortion of the codebook is obtained by taking the time average given as,

\[
D_{\text{avg}} = L^{-1} \sum_{i=0}^{L-1} d(\mathbf{X}_i, \hat{\mathbf{X}}_i)
\]

\( L \to \infty \)

This is based upon the assumption that speech signal is stationary and ergodic, thereby implying that the LPC vector process is also stationary and ergodic.

4.5 **CODEBOOK GENERATION**

Codebook forms the heart of a vector quantizer. Therefore, quite naturally it has invited the most interest amongst various leading researchers. A number of different methods for generating the codebook have been proposed [24]. In this section, a new method of obtaining a codebook from phonemes of the English language, is discussed. Before going
further into detail, it is quite important that the reader gets familiar with the various types of sounds that make up human speech.

4.5.1 **Speech Sounds**

Speech signals are composed of a sequence of sounds. These sounds and the transition between them serve as a symbolic representation of information.

The study of the rules of the language, which govern the arrangement of the sounds or symbols is the domain of linguistics, whereas the study and classification of the speech sounds is called phonetics.

4.5.2 **Phonemes**

Most languages can be described in terms of a set of distinctive sounds, known as Phonemes. In American English, there are 42 phonemes classified as Vowels, Diphthongs, Semi-vowels, and Consonants. Each of these are further broken down into sub-classes, depending upon the manner and place of articulation of the sounds within the vocal tract.

Each of these phonemes can be classified as either a continuant, or a noncontinuant sound. Continuant sounds are produced by a fixed (non time-varying) vocal tract configur-
ration excited by the appropriate source. The class of con-
tinuant sounds include the vowels, the fricatives (both
voiced and unvoiced), and the nasals. The rest (diphthongs,
semitones, stops and affricatives) are produced by a chang-
ing vocal tract configuration. These are therefore classed
as noncontinuants.

The above methodology can be used to block quantize
the speech data. One can think of an encoder (with codebook
constructed from phonemes), which maps the input speech sig-
nal into various phoneme sounds. The indices of these pho-

nemes are transmitted across the channel. A decoder, uses
these indices to retrieve the required phonemes from the
reproduction codebook and reconstructs the speech by conca-
tenating them in the proper sequence.

The description of the codebook generation from the
phoneme sounds, is given below.

4.5.3 Codebook Generation Steps

Fig. 4.2 depicts a block diagram of a Codebook gener-
at ed from phonemes. The various steps involved are described
below:

1. A word directory consisting of 42 English words spo-
ken by a trained speaker, is formed. Each word is
made up of at least one phoneme sound.
Fig. 4.2 Block Diagram for Code Book
Constructed from Phonemes
2. All the 42 words are preprocessed and digitized as described in sec. 3.2. Next, the phonemes are extracted from the spoken directory of words, by the operation of windowing. For example, the nasal consonant /m/ is obtained by truncating the word (mcx). The operation of truncation is accomplished through an implicit rectangular window. The size of the rectangular window can vary, depending on the duration of various phoneme sounds. An additional 22 sounds characterizing the transition between various phonemes (such as Vowels & Diphthongs), are also extracted.

3. Each phoneme sound data (including the 22 transient sounds) is segmented into successive frames (length 200 samples), which overlap each other by 100 samples. Next, an LPC analysis is done on each frame and the predictor coefficients obtained are grouped together to form an LPC vector. All the LPC vectors computed from successive frames of phoneme data are stored in a buffer or in memory.

4. The relative variations in the values of various LPC vectors obtained from each phoneme sound, are checked visually. Those LPC vectors which closely match one another, are first grouped together and then averaged out.
5. Each average LPC vector constitutes one codeword or template of the codebook.

6. The final codebook contains 64 codewords or averaged LPC vectors.

4.6 RESULTS

The word directory consisted of approximately 2 minutes of speech (in short words), spoken by one trained speaker. The speech data was pre-filtered between 200 Hz and 3200 Hz, and then sampled at 10 KHz and digitally pre-emphasized. For this study, a pre-emphasis factor of .95 and filter order of 8 was chosen. Analysis window or frame of 200 samples was chosen with shift of 100 samples.

On an average, each phoneme sound was segmented into 15 frames of speech. LPC parameters were extracted from each frame using the autocorrelation method (discussed in sec. 2.4).

The vector quantizer (its codebook generated from the phonemes sounds) was first tested on four speech segments spoken by one speaker (this speaker also contributed to the generation of the word directory). The average duration of these segments was 2 seconds. The pitch (including VUV decision) and the gain were both left at the original LPC rate (sec. 3.4) of 7 and 5 bits per frame, respectively. The LPC
coefficients, however, were compressed to 7 bits per frame (as against 80 bits per frame in the LPC vocoder). The overall compression achieved was from 9200 bits/sec (as in section 3.4) to 1800 bits/sec.

Informal listening tests revealed that synthesized speech obtained from the LPC-Vector quantizer was perfectly intelligible. However, when the quantizer was tested on speech segments spoken by other speakers (only male speakers were considered), a significant amount of degradation in the quality of the synthesized speech was observed.

After extensive subjective tests, it was concluded that the codebook obtained from the phonemes was Speaker Dependent. This was probably due to the fact that phoneme sounds are strongly influenced by the articulation, and by the context in which they occur. Also a great deal of variability is to be expected among speakers, producing the same phoneme sound. This variability is due to the inherent differences between the vocal tracts of speakers.

This prompted the author to investigate further into the area of LPC Vector Quantization, so as to design quantizers which are to a great extent speaker independent. In chapter V, various new design methods for obtaining an optimal Vector Quantizer using a long training sequence of speech data, are discussed.
Chapter V

VECTOR QUANTIZATION USING CLUSTERING TECHNIQUE

5.1 INTRODUCTION

In this chapter, an algorithm for the design of a good block or vector quantizer, based on a long training sequence of speech data is presented. The basic properties of the algorithm are discussed and demonstrated by examples. Also, a new technique of generating a codebook, which reduces the processing time by operating in a two level mode is discussed.

Finally, tree-search techniques, which significantly reduce the computational time at the transmitter, are discussed.

5.2 VECTOR QUANTIZATION OVERVIEW

As discussed in sec. 4.2, a vector quantizer is a source coder that consists of the following:

1. A finite reproduction codebook of vectors or codewords, represented by \( \mathbf{\hat{a}} = \{ \mathbf{y}_j \}_{j=1}^{N} \).
2. A nonnegative distortion measure \( d(X, Y) \) between each source vector \( X \) and codeword \( Y \in \mathcal{A} \).

3. A minimum distortion or nearest-neighbour encoding rule, \( q \), such that \( q(X) = Y_j \) if and only if \( d(X, Y_j) \leq d(X, Y_k) \) for all \( j \neq 1 \).

4. A partition, \( T = \{T_j: j = 1, 2, \ldots, N\} \), of the input vector space. The partition subsets or cells \( T_j = \{X: q(X) = Y_j\} \) for \( j = 1, 2, \ldots, N \). That is, \( T_j \) collects all input vectors mapping into the \( j \)th codeword in \( \mathcal{A} \).

5.3 **Performance Parameters of a VO**

There are two main parameters that provide quantitative performance measures of a vector quantizer. One is the code rate, represented by \( 'CR' \) and defined as \( CR = \log_2 N \) bits/frame. The code rate is just the number of bits needed to communicate or exactly specify one codeword. The second parameter is the average distortion defined as:

\[
D_{avg} = \left\{ d(X, q(X)) \right\}
\]

\[
= \frac{1}{N} \sum_{j=1}^{N} \mathbb{E} \left\{ d(X, Y_j) \bigg| X \in T_j \right\} p_r(X \in T_j)
\]

\[(5.1)\]
where \( E[d(X,Y_j) \mid X \in T_j] \) is the conditional expected distortion, given \( X \in T_j \), or, equivalently, given \( q(X) \geq Y_j \). \( \Pr(X \in T_j) \) is the probability of \( X \) being an element in cell \( T_j \).

Hence, it is desirable to keep both the code rate and the average distortion, as low as possible when designing a vector quantizer.

5.4 PREVIEW OF VO DESIGN

Traditional approaches for designing an optimum codebook \( A \) (by using eq. 5.1), unfortunately require a complete model of the input source vectors. Fortunately, there is a new design method for optimum quantizers \( [31] \) not requiring a probabilistic model, but instead using a long training sequence of source data \( \{X_i \mid i=0, 1, 2, \ldots, L-1\} \). Here, the expectations (Statistical averages) are approximated with long time averages i.e.

\[
E[d(X, q(X))] = L^{-1} \sum_{i=0}^{L-1} d(X_i, q(X_i)) \quad \text{--- (5.2)}
\]

Now, the operation of a vector quantizer can be thought of as partitioning the input vectors into cells, where all input vectors yielding a common reproduction are grouped together. Such a partition according to a minimum distortion rule is called Voronoi partition \([32]\). For exam-
ple, consider a two dimensional codebook containing four codewords. Also, consider a training sequence of twelve, two dimensional input vectors. Each input vector is mapped into the nearest-neighbour codeword, using minimum distortion rule. The partition of the training sequence is shown in Fig. 5.1a. The four circles represent the codewords of the codebook. The X's represent the training sequence of input vectors, and the voronoi regions are the quadrants containing the circles.

5.4.1 Centroid Definition

Consider a training sequence \( \{X_i; i=0,1,\ldots, L-1\} \), with a partition \( T \), then the Euclidean Centroid \([24]\) of the partition cell \( T_j \) is given by:

\[
\text{Cen}(T_j) = \frac{1}{||T_j||} \sum_{i: X_i \in T_j} X_i
\]

where, \( \text{Cen}(T_j) \) denotes the vector sum of all input vectors in the cell \( T_j \), and, \( ||T_j|| \) is the number of training vectors in the cell.

The euclidean centroids of Fig. 5.1a are depicted in Fig. 5.1b (the numerical values of the centroids are given in [24]). These centroids are represented as circles in the
Fig. 5.1a Two-Dimensional Minimum Distortion Partition

Fig. 5.1b Centroids of Fig. (5.1a)
figure. It is quite apparent that the centroid computation
has moved the codewords, to better represent the input vec-
tors which yielded those codewords. In other words, if one
replaces the codewords of Fig. 5.1a by these new centroids,
the average distortion would decrease. Also, the new code-
words yield a different minimum distortion partition of the
input sequence, as indicated by the broken line in Fig.
5.1b. This is the key to the Vector Quantization algorithm
which is described, in the next section.

5.5 Vector Design Algorithm

The algorithm works on a long training sequence of
speech data (in the form of LPC vectors). Each LPC vector is
made up of 'p' predictor coefficients, where 'p' is the vec-
tor dimension or block length. This algorithm is based on an
approach of Lloyd [33] and was first proposed by Gray et al
[24]. The heart of this algorithm is the computations of
the centroids of various partitions or clusters of the input
training set of vectors. Due to the clustering nature of the
algorithm, it is also sometimes referred to as the Clustering
algorithm. The operation of the algorithm can be de-
scribed as follows:

1. Initialization: Given is a training sequence of LPC
vectors \( \{X_i; i=0,1,\ldots, L-1\} \) and a distortion \( \varepsilon > 0 \).
Let \( \hat{A}_0 \) be an \( N \)-level codebook (\( N \) is the number of
codewords in the codebook). The input vectors \( X_i \)'s) are defined in \( 'p' \) dimensional euclidean space. Set \( s = 0 \) and \( D_{old} = \infty \).

2. Given the initial codewords \( \{Y_j; j=1,2,\ldots, M\} \) of the codebook \( \hat{X}_S \), find the minimum distortion partition \( \mathcal{T}^S = \{T_j; j=1,2,\ldots, M\} \) of the training sequence. That is, group all the vectors in the training sequence \( \{X_i; i=0,1,\ldots, L-1\} \), according to the nearest-neighbour codeword, by forming the sets

\[
T_j^S = \{ X_i : d(X_i, Y_j) \leq d(X_i, Y_l) \} \quad \text{for all } j \neq l.
\]

3. Next, compute the average distortion

\[
D = \frac{1}{L-1} \sum_{i=0}^{L-1} d(X_i, q(X_i))
\]

\[= \frac{1}{L-1} \sum_{i=0}^{L-1} \min_{l} d(X_i, Y_l) \quad \tag{5.4}
\]

4. If \( D_{old} - D < \epsilon \), stop, with the final codewords given by \( \{Y_j; j=1,2,\ldots, M\} \). Otherwise continue.

5. Find the centroid, \( \text{Cent}(T_j^S) \), for each partition cell of the input vectors. The new centroid is the vector which minimizes the average distortion for the vectors in the cell \( T_j^S \). Replace the codewords of the previous codebook \( \hat{X}_S \) by the new centroids, i.e.
\hat{\alpha}_{s+1} = \{ y_j : y_j = \text{Cen}(T^S_j) \} \text{ for } j = 1, 2, \ldots, N.

6. Replace 's' by 's+1' and go to step 2.

If, at some point, the partition $T^S$ has a cell $T^S_j$ which has no input training vectors, then the algorithm assigns the old codeword as centroid (i.e. $\text{Cen}(T^S_j) = y_j$) and continues.

Experimentally, it was observed that the algorithm always converges. The mathematical proof of this is given elsewhere [31].

In order to design an optimal quantizer, it is important that the training sequence be very long ($L \rightarrow \infty$) so that it contains vectors representing all possible speech sounds. Also, the initial codewords of the codebook $\hat{\alpha}_0$ should be properly chosen. The former condition can easily be satisfied, but the latter one is not so straightforward. There are several ways to choose the initial codebook $\hat{\alpha}_0$, required by the algorithm. One of the methods is to choose the first $M$ vectors of the training sequence as the initial codewords of $\hat{\alpha}_0$. However, this approach is discarded, as heuristically one would like the initial codewords to be well separated, and $M$ consecutive samples may not be. An obvious modification, more natural for highly correlated
speech data is to select several widely spaced codewords from the training sequence. This approach will yield a good quantizer, but in general, there may be numerous codewords well spaced in the training sequence and many of them may yield a poor quantizer.

A new method of choosing initial codewords of $\hat{A}_0$, which gave satisfactory results, is discussed next. This approach is based on "Splitting" technique.

5.6 *SPLITTING TECHNIQUE*

In this method, one can start with a small codebook and recursively construct larger ones. Fig. 5.2 depicts the operation of splitting. Here, one first finds the centroid of the entire training sequence ($X_0, X_1 \ldots, X_{L-1}$), as shown in (Fig. 5.2a) for a two dimensional input set. This single codeword is then split to form two codewords (Fig. 5.2b). The splitting operation is performed by slightly perturbing the initial codeword, to form a second distinct codeword. The VQ algorithm is then run to get a good 2-level codebook as indicated in (Fig. 5.2c). Next, the two good codewords of the 2-level codebook are further split to form 4 new codewords (Fig. 5.2d) which are initial codewords for a 4 level codebook. Once again the algorithm is run to produce a good 4-level codebook shown in (Fig. 5.2e). The dotted lines in (Fig. 5.2c) and (Fig. 5.2e) indicate the voronoi regions.
Fig. 5.2: Illustration of the Operation of Splitting
The design procedure continues in this way, i.e. the final codeword of one stage is split to form an initial codeword for the next. Thus, one can begin with a $N$-level quantizer with $M = 2^R$, $R = 0, 1, 2, \ldots$, and continue until the initial codewords for an $N$-level quantizer are achieved. The operation of the algorithm using "Splitting" technique is as follows:

5.6.1 Algorithm Using Splitting Technique

1. Initialization: Set $M = 1$. $\hat{A}(1) = \text{Cen}(X)$ (the centroid of the entire training sequence.)

2. Given the reproduction codebook $\hat{A}(M)$ containing $M$ codewords $(Y_j; j=1, 2, \ldots, M)$, "Split" each vector $Y_j$ into two close vectors, namely, $Y_j + \Delta$ and $Y_j - \Delta$, where $\Delta$ is a fixed perturbation vector. Set $\hat{A}_0 (2M) = \{(Y_j + \Delta, Y_j - \Delta); j=1, 2, \ldots, M\}$.

3. Run the clustering algorithm for a $2M$ level quantizer on $\hat{A}_0 (2M)$ to produce a good reproduction codebook $\hat{A}(2M)$.

4. Check if $M = M/2$? If so, replace $M$ by $2M$ and return to step 2. Otherwise the algorithm terminates.

The final quantizer described by $\hat{A}(M)$ or $\hat{A}$, has good codewords, given by $(Y_j; j=1, 2, \ldots, M)$. Fig. 5.3 depicts the
Speech Data (LPC Vectors)

Training Sequence
\(X_i; i=0,1,2,..., L-1\)

Compute Centroid of
The Training Sequence

\(M=1\)
\(Y_1=\text{Gen}(X)\)

Split Each Vector
\(Y_j + \Delta\)
\(Y_j - \Delta\)
\((j=1,2,..., M)\)

Code book Generation
(Clustering Algr.)

\(M=2M\) No
\(M=N/2\) Yes

Final Code book \(A(N)\)
\((Y_j; j=1,2,..., N)\)

Fig. 5.3 Block Diagram for Code book Construction Using Splitting Technique
block diagram of the whole procedure. The main advantage of using the above mentioned scheme is that, given a training sequence of data, the final codewords of the codebook, \( \hat{A} \), would always be unique.

5.7. **TWO-LEVEL VQ ALGORITHM**

In this section, the development of a new codebook, that has a significantly larger number of entries to facilitate a more accurate reconstruction of speech at the receiver is discussed.

The block diagram of the two-level VQ algorithm is given in Fig. 5.4, and can be described in the following steps:

1. **Initialization**: Training sequence of the input LPC vectors is given by \( \{X_i : i = 0, 1, 2, \ldots, L-1\} \).

2. Using the algorithm outlined in sec. 5.6.1, get the optimal codebook \( \hat{A} \), containing \( N \) codewords \( \{Y_j : j = 1, 2, \ldots, N\} \). Next, collect each minimum distortion partition cell \( T = \{T_j : j = 1, 2, \ldots, N\} \), cf the training sequence. These partition cells or clusters are subsets of the main set, given by \( \{X_i : i = 0, 1, 2, \ldots, L-1\} \).

3. Each partition cell is a sub-sequence of the main training set \( \{X_i : i = 0, 1, 2, \ldots, L-1\} \). Thus, there are \( N \) training sub-sequences of LPC vectors.
Training Sequence
\( (x_i; 0, 1, \ldots, L-1) \)

Code Book
\( (y_j; j=1, 2, \ldots, N) \)

Partition 'Γ'
\( (t_j; j=1, 2, \ldots, N) \)

Clustering Algorithm working on Sub-sets of Training Sequence

Codewords of \( K(u_k) \)
\( (j=1, 2, \ldots, N) \)
\( (k=1, 2, \ldots, J) \)

Two Level Code Book

Fig. 5.4 Two-Level Codebook
4. Next, \( M \) new codebooks are obtained by running the clustering algorithm on \( M \) training sub-sequences (or \( M \) subsets of the original training sequence). Each of these codebooks \( \hat{\Lambda}^j \), have \( J \) codewords \( \{ u_{kj}^j \} \), \( k = 1, 2, \ldots, J \).

5. Finally, the codewords of the first codebook \( \hat{\Lambda} \) and of the second stage codebooks \( \hat{\Lambda}^j : j = 1, 2, \ldots, M \) are combined together to form one complete final codebook. This codebook is called a two-level codebook, since it has an \( M \)-level codebook stored in the first level and \( M \) \( J \)-level codebooks stored in the second level.

5.7.1 Two-Level Encoder/Decoder

Fig. 5.5 depicts the operation of the two-level encoder/decoder. An encoder first maps the input vector \( \mathbf{I} \) into its nearest-neighbour codeword, \( \{ I_j \} \in \hat{\Lambda} \) (first level codebook). This is followed up by mapping the input vector \( \mathbf{I} \) into the minimum distortion reproduction codeword \( \{ u_{kj}^j \} \in \hat{\Lambda}^j \) (jth second level codebook). Finally, the indices \( j \) and \( k \) are encoded and then transmitted across the channel.

The decoder works only with the second level of the two-level codebook (see Fig. 5.5b). Once again, the index \( j \) gives the location of the jth second level codebook, and
Fig. 5.5 a Encoder

Index j

2nd Level Codebook

Index k

Fig. 5.5 b Decoder

Synthesis Block

Fig. 5.5 Implementation of an Two-Level Codebook
the index \( k \) gives the location of the reproduction code-
word (of input vector \( I \)) to be picked from the codebook \( \hat{ \lambda } \).

The computational time (at the encoder end) is signifi-
cantly reduced by using the technique described above. For
example, a 10-bit codebook (1024 codewords) generated using
the algorithm of sec. 5.6.1, will search everyone of the
prestored codewords (i.e. 1024 searches) so as to find the
best match for each input vector \( I \). However, a 10-bit two-
level codebook (\( N=32, \ J=32 \)) involves only 64 searches to
find the best match codeword or vector.

5.8 EXPERIMENTS AND RESULTS

Vector Quantizers were designed and tested for a var-
iety of code rates and data sets. The codebooks were tested
by using large segments of speech, spoken by five different
male speakers. These tests included both, the closed tests
(inside the training sequence) and the open tests (outside
the training sequence) speech samples.

The data base (consisting of 4 minutes of speech),
was constructed from the first speaker. To avoid including
non-speech sounds in the data base, the speech was recorded
in a noise free environment, using high fidelity analog re-
cording equipment. The analog speech was then properly pre-
processed and digitized, as described in sec. 3.2.
5.8.1 Training Sequence And Analysis Considerations

The training sequence (consisting of LPC vectors) was extracted from the main database. An 8-order LPC analysis based on the autocorrelation method was chosen, with analysis frame length of 200 samples and with a shift of 100 samples. Since the codebook is constructed from the training sequence of LPC vectors, those vectors extracted from non-speech (e.g., silent) frames will influence the codebook just as much as those vectors of voiced/unvoiced speech frames. In order to eliminate the possibility of allocating codewords to represent non-speech signals, a simple energy threshold test was established to discard silent frames of speech, before performing LPC analysis. The final training set consisted of 10,000 LPC vectors.

5.8.2 Performance Evaluation
5.8.2.1 Perceptual Listening Tests

The first experiment consisted of designing two codebooks (one having 32 codewords and the other having 64 codewords), from the final training set, using algorithm described in sec. 5.6.1. In both cases, the convergence was achieved in fewer than 13 iterations, for a convergence threshold (ε) of .001.
Two test data sets were established to evaluate the performance of the codebooks. The first set consisted of four sentences, 200 frames each, extracted from the training sequence. Informal listening sessions indicated that both codebooks (32 & 64 codewords), when tried on the first data set, retained good intelligibility. Next, the performance of the codebooks was tested using the second data set. This set contained seven segments of speech constructed from the rest of the speakers (speakers not in the training set). Subjectively, there was a clear increase in quantization noise and fuzziness of the speech.

The second experiment comprised of testing two design cases (N=32 & J=32 for the first case; N=64 & J=16 for the second case), of the two-level codebook, constructed from the same training set. Both the codebooks (32, 32 & 64, 16) were perceived to be much better (highly intelligible) and smoother than the full searched codebooks (32 & 64 codewords), for both open and closed test samples.

5.8.2.2 Quantitative Evaluation

Quantitative evaluation study of various codebooks was done using two performance criterions, namely: Average Distortion, and Signal-to-Quantization Noise Ratio (SQNR). Average distortion of codebooks was computed as described in
sec. 5.4, using square error distortion measure. To maintain continuity, codebook obtained from the phoneme sounds is also included. The resultant distortion obtained for both closed and open test samples is given in Table I. The quantitative difference between the SQNR (using the closed test) and the SQNR (using the open test), for the various codebooks, are summarized in Table II. The SQNR was computed using the following expression

\[
\text{SQNR} = \frac{\sum_{i=1}^{NL} \sum_{j=1}^{NF} x_{ij}^2}{\sum_{i=1}^{NL} \sum_{j=1}^{NF} (x_{ij} - \hat{x}_{ij})^2} \quad \text{----(5.5)}
\]

where, \( x_{ij} \) is the \( i \)th sample of the \( j \)th frame of the reference speech (speech synthesized from the LPC Vocoder) and \( \hat{x}_{ij} \) is the \( i \)th sample of the \( j \)th frame of the 'VQ' speech.

The quantitative results reinforce the quality loss (incurred for the open test, as against the closed test samples) observed from the subjective tests. Note that over 3 dB is lost by using a codebook designed for one speaker on another.
<table>
<thead>
<tr>
<th>CODEBOOK</th>
<th>SPEAKER INDEPENDENT (OPEN TEST)</th>
<th>SPEAKER DEPENDENT (CLOSED TEST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONEMES</td>
<td>0.960972</td>
<td>0.653175</td>
</tr>
<tr>
<td>32-LEVEL</td>
<td>1.026659</td>
<td>0.417650</td>
</tr>
<tr>
<td>64-LEVEL</td>
<td>0.691929</td>
<td>0.293550</td>
</tr>
<tr>
<td>32/32 LEVEL</td>
<td>0.364240</td>
<td>0.118749</td>
</tr>
<tr>
<td>64/16 LEVEL</td>
<td>0.292045</td>
<td>0.114195</td>
</tr>
</tbody>
</table>

Table I  Average distortion for both open and closed test samples.
<table>
<thead>
<tr>
<th>CODEBOOK</th>
<th>SPEAKER INDEPENDENT (OPEN TEST)</th>
<th>SPEAKER DEPENDENT (CLOSED TEST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONES</td>
<td>1.178 dB</td>
<td>4.031 dB</td>
</tr>
<tr>
<td>32-LEVEL</td>
<td>2.761 dB</td>
<td>6.139 dB</td>
</tr>
<tr>
<td>64-LEVEL</td>
<td>2.987 dB</td>
<td>6.461 dB</td>
</tr>
<tr>
<td>32/32 LEVEL</td>
<td>3.187 dB</td>
<td>7.225 dB</td>
</tr>
<tr>
<td>64/16 LEVEL</td>
<td>3.449 dB</td>
<td>7.424 dB</td>
</tr>
</tbody>
</table>

Table II  Signal-to-Quantization Noise Ratio for various codebooks.
5.9 Tree-Search Technique

The extensive computations required at the source end, make full search vector quantizers very difficult to implement for practical purposes. The main reason being that these quantizers search every codeword (in a codebook), in order to find the closest match for each input vector.

The computational burden can be significantly reduced by forming a tree-search structure for the codebooks, and making sequential search instead of full search. To illustrate such a tree-search structure, a binary tree for a 4-bit (16 codewords) case is shown in Fig. 5.6.

5.9.1 Generation of Tree Structures

Using Fig. 5.6 as an example, the procedure of generating a binary tree structure is explained next.

1. Make the initial node or point as the centroid of the entire set.

2. Split the centroid to form two initial codewords and run the clustering algorithm to obtain two good final codewords. These codewords form the first level of the binary tree.

3. Next, split the two codewords to obtain 4 initial codewords for a 2-bit codebook. Instead of running a
TREE-SEARCH STRUCTURE CODEBOOK
4-BIT

FIG. 5.6
full search clustering algorithm on the resulting 4-word codebook, the training sequence is divided into two parts, collecting together all those vectors encoded into a common codeword in the 1-bit codebook (first level of the tree). For each of these sub-sequences of training vectors, one then finds a good 1-bit code using the algorithm. The final codebook (so far) consists of the four codewords in the two 1-bit codebooks designed for the two sub-sequences (second level in Fig. 5.6)

4. Step three is repeated so as to build up the tree structure.

An encoder selects one of the reproduction codewords from the tree structured codebook \(2^R\) codewords) not by an ordinary full search of this codebook, but, instead it searches the codebook sequentially. That is, it uses the first 1-bit codebook (first level of the tree) designed on the whole training set to select a second 1-bit codebook (second level of the tree). Next, the encoder chooses the minimum distortion codeword from this codebook which leads to the third 1-bit codebook (third level of the tree). This procedure is repeated till the encoder reaches the \(R\)th level, and it then picks the best codeword in the \(R\)th 1-bit codebook. Thus, the encoder makes a succession of \(R\) minimum distortion choices from \(R\) binary codebooks, or a total of \(R\) binary searches, as against \(2^R\) full searches.
One should note, that the performance of a tree-searched encoder is no longer optimal as compared to a full search encoder. The search, however, is much more efficient if done sequentially than is a full search. Thus, one may trade performance for efficiency of implementation.

In addition to a binary tree, one may consider numerous other tree structures [35]. A table of some tree structures and the corresponding number of codebook searches for a 10-bit codebook is provided in the table III.

5.9.2 Modified Tree Structure

A new tree structure which considerably improves the performance of a binary tree-search codebook, has been developed. The modified tree structure for a 4-bit case is shown in Fig. 5.7. It can be observed, that the tree no longer represents a binary tree, as some of the nodes have more than two branches, denoted by dotted lines. These dotted lines or links were formed by using the following procedure:

1. A full search 4-bit codebook was generated using a training set of 10000 vectors. All the intermediate results (codewords and partition cells) were stored and tabulated.
<table>
<thead>
<tr>
<th>Branching Level</th>
<th>Number of Branches at Each Node</th>
<th>Number of Searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Binary Search) 10</td>
<td>ALL 2's</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>ALL 4's</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>8, 8, 8, 2</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>16, 16, 4</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>32, 32</td>
<td>64</td>
</tr>
<tr>
<td>2</td>
<td>64, 16</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>128, 8</td>
<td>136</td>
</tr>
<tr>
<td>2</td>
<td>256, 8</td>
<td>260</td>
</tr>
<tr>
<td>2</td>
<td>512, 2</td>
<td>514</td>
</tr>
<tr>
<td>(Full Search) 1</td>
<td>1024</td>
<td>1024</td>
</tr>
</tbody>
</table>

Table III  Tree-search structures for a 10-bit codebook.
FIG. 5.7 NON-BINARY TREE-SEARCH STRUCTURE FOR A 4-BIT CODEBOOK
2. It was observed that 5548 vectors of the training set, clustered around codeword one (or node 1) of the first 1-bit codebook and the rest clustered around the second codeword. Next, 2760 vectors from the sub-sequence of 5548 vectors, clustered around node 1 of the second 1-bit codebook and the rest (i.e. 2788) clustered around node 2 (second level of the tree). Also, an additional 766 training vectors from the node 2 of previous 1-bit codebook (first level of the tree), now, clustered around node two of the second binary codebook, hence a link between the two nodes was formed, denoted by a dotted line. This procedure was repeated and the appropriate links were established at each level of the tree structure.

The modified tree structure maintains the quality (same as full search codebook) with computational efficiency still preserved.

5.10 BIT RATE OF Vo

In all the experiments, the sampling rate chosen was 10 KHz and LPC's were computed 100 times per second. That is, analysis frame rate was 100 frames/sec.
Now, the bit rate of a quantizer can be shown as:

\[
\text{Bit rate} = CR \cdot F_r \text{ bits/sec} \quad (5.6)
\]

where, \( CR \) is the code rate of the quantizer.

\( F_r \) is the frame rate.

A 6-bit vector quantizer operated at a bit rate of 600 bits/sec and a 10-bit quantizer operated at a rate of 1000 bits/sec. The above bit rates (for both, 6 & 10 bit VQ's) were reduced to half (300 for 6-bit and 500 for 10-bit VQ's), by doing the LPC analysis in a non-overlap mode. That is, no overlap between successive frames of speech was considered. However, at the receiver, the speech was reconstructed at a rate of 100 frames/sec, in an overlap fashion. The LPC coefficients for the intermediate frames were obtained through direct interpolation.

Perceptually, very little difference could be found between the two sets of synthetic speech, obtained by using overlap and non-overlap techniques respectively, at the transmitter.

The bit rate could be further reduced by using a sampling rate of 8 KHz (as the nyquist frequency is only 3.2 KHz). The overall bit rate (including the pitch, the gain
and the V/UV decision parameters), for various vector quantizers is tabulated in Table IV. Note, the number of bits allocated for the pitch, the gain and the V/UV decision is the same as described in sec. 3.4.
<table>
<thead>
<tr>
<th>CODEBOOK</th>
<th>BIT RATE FOR (Pitch, Gain, V/UV) bits/sec.</th>
<th>BIT RATE FOR LPC'S bits/sec.</th>
<th>TOTAL BIT RATE bits/sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>32,32 LEVEL</td>
<td>1200</td>
<td>1000</td>
<td>2200</td>
</tr>
<tr>
<td>64,16 LEVEL</td>
<td>1200</td>
<td>1000</td>
<td>2200</td>
</tr>
<tr>
<td>64 LEVEL</td>
<td>1200</td>
<td>600</td>
<td>1800</td>
</tr>
<tr>
<td>32 LEVEL</td>
<td>1200</td>
<td>500</td>
<td>1700</td>
</tr>
</tbody>
</table>

**NON-OVERLAP ANALYSIS FRAME RATE (50 frames/sec.)**

<table>
<thead>
<tr>
<th>CODEBOOK</th>
<th>BIT RATE bits/sec.</th>
<th>TOTAL BIT RATE bits/sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>32,32 LEVEL</td>
<td>600</td>
<td>1100</td>
</tr>
<tr>
<td>64,16 LEVEL</td>
<td>600</td>
<td>1100</td>
</tr>
<tr>
<td>64 LEVEL</td>
<td>600</td>
<td>900</td>
</tr>
<tr>
<td>32 LEVEL</td>
<td>600</td>
<td>850</td>
</tr>
</tbody>
</table>

**REDUCED SAMPLING RATE (8kHz) WITH FRAME RATE (50 frames/sec.)**

<table>
<thead>
<tr>
<th>CODEBOOK</th>
<th>BIT RATE bits/sec.</th>
<th>TOTAL BIT RATE bits/sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>32,32 LEVEL</td>
<td>480</td>
<td>880</td>
</tr>
<tr>
<td>64,16 LEVEL</td>
<td>480</td>
<td>880</td>
</tr>
<tr>
<td>64 LEVEL</td>
<td>480</td>
<td>720</td>
</tr>
<tr>
<td>32 LEVEL</td>
<td>480</td>
<td>680</td>
</tr>
</tbody>
</table>

Table IV Overall bit rate for various quantizers.
Chapter VI
SUMMARY AND CONCLUSIONS

The objective of this thesis was to study the performance of a linear prediction analysis/synthesis model; to review and demonstrate the working of a LPC vocoder (operating at the rate of 9200 Kbps); and finally use this knowledge to examine and develop various vector quantization schemes for speech data compression.

Throughout this work the quality of the processed speech (obtained from various quantizers) was judged by informal perceptual listening tests, as well as by quantitative measures such as SQNR (Signal-to-Quantization Noise Ratio). However, the most crucial factor was a qualitative one - the perceptual speech quality, which ultimately determined whether the quantizer was good or bad.

The main contribution and conclusions of this research can be summarized as follows:
6.1 **LPC Analysis/Synthesis of Speech**

A 12th order \( p=12 \) synthesis model was sufficient to provide an adequate representation of the speech signals. A slight degradation in the quality of the synthesized speech (especially for nasal & plosive sounds) was noticeable for \( p=8 \), and although poor in quality, speech could be synthesized with \( p \) as low as 2.

6.2 **LPC Vocoder**

A LPC vocoder capable of operating at 9200 bps (a compression ratio of 7:1 over PCM) was developed. Exhaustive testing with speech data revealed that the signal reconstructed at the receiver had sufficient fidelity (when compared with original signal) and further, the information content was fully preserved.

A further reduction by a factor of 2 (i.e. a rate of 4600 bps) was found to be possible if non-overlap LPC analysis (i.e. a LPC analysis frame rate of 50 Hz) was performed on the digitized speech.
6.3 Vector Quantization Techniques

The codebook constructed from the phoneme sounds was found to be speaker dependent; that is, the codebook worked well on the speaker from whose speech, the phonemes were derived. However, a significant amount of degradation in the quality of synthetic speech was observed when test utterances from other speakers were utilized.

Satisfactory results were obtained by using the codebook constructed from a long training sequence of data. A series of subjective tests confirmed that the codebook was text independent and to a great extent speaker independent.

Codebooks operating in a two-level mode significantly reduced the processing time, while maintaining the quality of the synthetic speech (same as above mentioned codebook).

A binary tree structure for implementing the codebook selection was developed. This structure reduced the processing time, but affected the performance of the codebook. A new non-binary tree-search structure for implementing the codebook was found to perform as well as the full search codebook, but with significantly reduced processing time.
In conclusion, this work shows, that a significant amount of speech compression (i.e. bit rates as low as 680 bits/sec.) could be achieved for storage and transmission applications.
Chapter VII

Suggestions for Future Research

Many future research areas are suggested by this work:

1. A great deal of study needs to be done concerning the codebook generation. In general, there are a number of methods for initializing the codebook generation and carrying out the cell splitting. Different techniques should be compared so that both rate of convergence and size of resultant distortion can be compared.

2. A longer training sequence comprised of several speakers (including female speakers), should be considered. Intuitively, one feels that it should yield codebooks with better performance (on the average) outside of the training sequence.

3. One could further reduce the bit rate by using a variable analysis frame rate (i.e., skipping adjacent frames in a non-overlap analysis). The intelligibility of the speech signal may still be preserved as speech characteristics do not change much from frame to frame.
4. The processing time, at the transmitter end, can be further reduced by working with a codebook of smaller dimension. Also, increasing the dimension of the reproduction codebook, at the receiver end, may yield better synthetic quality.

5. Tree-search structures should be dealt in detail as they could ultimately lead to a real-time hardware realization of various vector quantization techniques.

6. Finally, vector quantization technique could be applied in template generation for word recognition and speaker identification.
Appendix A

COVARIANCE/AUTOCORRELATION METHODS
APPENDIX A

Covariance/Autocorrelation Methods

To solve for the linear predictor coefficients \( \{ a_i \} \) (equation 2.17 in chapter II), one must compute the quantities \( C_{ij} \) for \( 1 < i \leq p \) and \( 1 \leq j < p \).

Now, rewriting eq. (2.15) in chapter II:

\[
C_{ij} = \sum_{n=n_0}^{n_1} s(n-i) s(n-j) \quad \text{--- (A.1)}
\]

The limits of summation so far were purposely left unstated. There are two popular approaches to that question leading to two different methods of linear prediction analysis.

Covariance Method

In this method, \( n_0 \) is set equal to \( p \) (order of the model) and \( n_1 = N-L-1 \), with the main assumption that \( s(n) \) exists or is available for \( (n=0, 1, 2, \ldots, N-L-1) \). The error (eq. 2.13a) is minimized only over the interval \( [p, N-L-1] \), and all 'NL' speech samples are used in calculating the covariance matrix elements \( C_{ij} \) of eq. (A.1) i.e.

- 102 -
\[ E = \sum_{n=p}^{NL-1} e(n) \]  \[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (A.2) \]

Eq. (2.17) leads to

\[ \sum_{i=1}^{p} a * C_{ij} = -C_{0j} \quad ; \quad 1 \leq j < p \]  \[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (A.3) \]

Eqs. (A.3) can be expressed in the form of \((p \times p)\) matrix which is given below,

\[
\begin{bmatrix}
C & C & C & \ldots & C \\
11 & 12 & 13 & \ldots & C \\
C & C & C & \ldots & C \\
21 & 22 & 23 & \ldots & C \\
C & C & C & \ldots & C \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
C & C & C & \ldots & C \\
p1 & p2 & p3 & \ldots & pp \\
\end{bmatrix}
\begin{bmatrix}
a \\
1p \\
2p \\
3p \\
\vdots \\
pp \\
\end{bmatrix}
= \begin{bmatrix}
C \\
01 \\
C \\
02 \\
C \\
03 \\
\vdots \\
C \\
0p \\
\end{bmatrix}
\]  \[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (A.4) \]

This matrix is symmetric (i.e. \(C_{ij} = C_{ji}\)) and also positive definite. The coefficients \(C_{ij}\) are obtained from a correlation of the speech samples, given by

\[ C_{ij} = C_{ji} = \sum_{n=p}^{NL-1} s(n-i) s(n-j) \]  \[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (A.5) \]

Since the matrix is symmetric, the above summation is evaluated only for \(j=0,1,\ldots \quad i\) and \(i=0,1,\ldots \quad p\). Thus, only \(p(p+1)/2\) coefficients are to be calculated, and the solution of eq. (A.3) is generally obtained by Cholesky decomposition method [18].
Autocorrelation Method

The autocorrelation method is defined by setting $n_0 = -\infty$ and $n = \infty$. The speech samples $s(n)$ are only defined in the interval $0 \leq n \leq NL - 1$ and are zero elsewhere. These limits allow $C_{ij}$ to be simplified as

$$C_{ij} = \sum_{n=-\infty}^{\infty} s(n-i) s(n-j)$$

--- (A.6)

Note, due measures have to be taken so that $s(n-i)$ or $s(n-j)$ do not take negative index values for any value of $i$, $j \& n$. Also, the speech process $s(n)$ is assumed to be stationary and ergodic (i.e. time average of the speech data can be obtained). Thus, $C_{ij}$ will depend upon $|i-j|$.

$$C_{ij} = \sum_{n=-\infty}^{\infty} s(n) s(n+|i-j|)$$

$$= \sum_{n=0}^{NL-1-|i-j|} s(n) s(n+|i-j|)$$

$$= R(|i-j|)$$

--- (A.7)

where, $R(|i-j|) = R(l) = \sum_{n=0}^{NL-1} s(n) s(n+l) \quad l \geq 0$ --- (A.8)

The above equation is the short time autocorrelation function evaluated for $|i-j|$. Since $R(l)$ is an even function, it follows that
\[ C = R(\mid i-j \mid) \quad i=1,2,\ldots, p \]
\[ ij \quad j=1,2,\ldots, p \quad (A.9) \]

That is,
\[ C \begin{array}{ccc}
11 & 22 & 33 \\
C & C & C \\
12 & 23 & 32 \\
\end{array} \]

Therefore, Eq. (A.3) can be written as:
\[ \sum_{i=1}^{p} a \ast R(\mid i-j \mid) = -R(j) \quad : 1 \leq j \leq p \quad (A.10) \]

The set of equations given by eq. (A.9) can be expressed in a matrix form as

\[
\begin{bmatrix}
R(0) & R(1) & R(2) & \ldots & R(p-1) \\
R(1) & R(0) & R(1) & \ldots & R(p-2) \\
R(2) & R(1) & R(0) & \ldots & R(p-3) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
R(p-1) & R(p-2) & R(p-3) & \ldots & R(0) \\
\end{bmatrix} \begin{bmatrix}
a \\
av \\
\vdots \\
av \\
ap \\
\end{bmatrix} = \begin{bmatrix}
-1 \\
-2 \\
-3 \\
\vdots \\
-p \\
\end{bmatrix} \quad (A.11) 
\]

This \((p \times p)\) matrix of autocorrelation values is a Toeplitz matrix, i.e., it is symmetric and all the elements along any diagonal are equal. Also, the matrix is positive definite. These special properties of the above matrix leads to gener-
al algorithms for the solution of eq. (A.9), and the most efficient one is given by Levinson's recursion algorithm [18].

**Comparison between Autocorrelation and Covariance Method**

The two major issues in the computation of the predictor coefficients are the number of multiplications (computational efficiency) and the stability.

For the covariance method, the computation of the correlation matrix requires \((N \times p)\) multiplications and by using Cholesky decomposition procedure, it requires a number of multiplications proportional to \(p^3\). For the autocorrelation method, the computation of the autocorrelation requires somewhat less computation (\((N \times p)\) multiplications for the correlation matrix, and about \(p^2\) multiplications by using Levinson's-Robinson method). Also, between the two methods one observes that stability is a problem in the covariance method (i.e. it does not guarantee the stability of \(H(z)\) given in eq. 2.7), whereas in the autocorrelation method, \(H(z)\) is guaranteed to be stable.
Appendix B

SPEECH WAVEFORMS
Fig. B.1 (a) Original speech waveform (segment of 200 msec.)

Sentence: 'PAPA NEEDS TWO SINGERS'

Fig. B.1 (b) Synthesized waveform (p=12)

Sentence: 'PAPA NEEDS TWO SINGERS'
Fig. B.2 (a) Synthesized waveform ('p'=8)
Sentence: 'PAPA NEEDS TWO SINGERS'

Fig. B.2 (b) Scalar quantized waveform ('p'=8)
Sentence: 'PAPA NEEDS TWO SINGERS'
Fig. B.3 Scalar quantized waveform with non-overlap technique (frame rate of 50 Hz.)

Sentence: 'PAPA NEEDS TWO SINGERS'
Fig. B.4 (a) Reference speech waveform (scalar quantized)
Sentence: 'EACH FORCE IN ITS OWN WAY IS CREATIVE'

Fig. B.4 (b) Vector quantized waveform (32-level codebook)
Sentence: 'EACH FORCE IN ITS OWN WAY IS CREATIVE'
Fig. 3.5 (a) Vector quantized waveform (64-level codebook)
Sentence- 'EACH FORCE IN ITS OWN WAY IS CREATIVE'

Fig. 3.5 (b) Vector quantized waveform (32/32 level codebook)
Sentence- 'EACH FORCE IN ITS OWN WAY IS CREATIVE'
Fig. B.6 (a) Vector quantized waveform (64/16 level codebook)
Sentence- 'EACH FORCE IN ITS OWN WAY IS CREATIVE'

Fig. B.6 (b) Vector quantized waveform (32/32 level codebook) with non-overlap technique
Sentence- 'EACH FORCE IN ITS OWN WAY IS CREATIVE'
Fig. B.7 Vector quantized waveform (64/16 level codebook) with non-overlap technique

Sentence: 'EACH FORCE IN ITS OWN WAY IS CREATIVE'
Appendix C

LISTING OF COMPUTER PROGRAMS
*****  THIS PROGRAM SEGMENTS THE SPEECH DATA  *****
*****  INTO FRAMES AND COMPUTES THE LPC  *****
*****  COEFFICIENTS FOR EACH FRAME  *****

PROGRAM NAME : GENPREMP

REAL KM
DIMENSION ISS(2000), X(1000), H(200), NAME(10), S(200)
DIMENSION ALPHA(21), KM(20), A1(20, 21)
ACCEPT 'ENTER THE NAME OF INPUT SPEECH FILE'
READ(11, 78) (NAME(I), I=1, 10)
FORMAT(10A2)
ACCEPT 'LENGTH OF THE SPEECH FILE ', IK
ACCEPT '# OF RECORDS ', NREC
M=8
N=200

SPEECH SAMPLES ARE READ HERE.

ILEN=IB+2
OPEN 0, NAME, LEN=ILEN
CALL READRN(0, 1, ISS, NREC, IER)
CALL CHECK(IER)
CLOSE 0
ACCEPT 'NAME OF THE PITCH FILE '
READ(11, 14) (NAME(I), I=1, 10)
FORMAT(10A2)
ACCEPT 'LENGTH OF THE PITCH FILE ', IPL
OPEN 1, 'FILELPC'
OPEN 2, 'ALPHA'
OPEN 0, NAME
OPEN 4, 'FILEREF'

GENERATING HAMMING WINDOW

PI=8*ATAN(1.)
DO 92 I=1, 200
  H(I)=.54-.46*COS(PI*(FLOAT(I-1)/193.0))
92 CONTINUE

PRE-EMPHASIS PROCEDURE

X(I)=FLOAT(ISS(I))/10
ABC=IB+4
OPEN 3, 'PREMP1', LEN=ABC
DO 1 I=1, NREC
  DO 2 K=1, IB
    IF(I.EQ.1.AND. K.EQ.1)GO TO 1
    LL=(I-1)*1000+K
    XX(K)=FLOAT(ISS(LL))-.95*FLOAT(ISS(LL-1))
  CONTINUE
  CALL WRITRN(3, I, X, 1, IER)
  TYPE 'VALUE OF I=', I
  CALL CHECK(IER)
1 CONTINUE
CLOSE 3
OPEN 3, 'PREMP1', LEN=4
KLN=1
DO 88 KLP=1, IPL
READ(0) IS
IF(IS .EQ. 1) GO TO 2
CALL READRN(I, KLM, X, 200, IER)
CALL CHECK(IER)
DO 21/I=1, 200
S(I)=X(I)/100+H(I)
21 CONTINUE

LINEAR PREDICTOR COEFFICIENTS OF ALL ZERO 8TH ORDER INVERSE FILTER ARE COMPUTED HERE.

CALL LPC(S, ALPHA, KM, N, M, A1)

LPC COEFFICIENTS ARE STORED

WRITE(1) (A1(I, KK), KK=1, 9)

REFLECTION COEFFICIENTS ARE STORED

WRITE(4) (KN(KK), KK=1, 8)

SQUARED ERROR VALUE IS STORED

WRITE(2) ALPHA(9)
IF(IS .EQ. 0) IS=100
KLN=KLN+IS
TYPE KLN
GO TO 88
KLN=KLN+100
TYPE KLN
CONTINUE
CLOSE 4
CLOSE 3
CLOSE 0
CLOSE 2
CLOSE 1
STOP
END
SUBROUTINE IMPLEMENTS "LEVINSON'S RECURSIVE METHOD" FOR OBTAINING LINEAR PREDICTOR COEFFICIENTS OF HIGHER ORDER INVERSE FILTER FROM PREVIOUS LOWER ORDER FILTER IN A RECURSIVE WAY.

PROGRAM NAME : LFC

SUBROUTINE LPC(S, ALPHA, KM, N, N, A)
REAL KM
DIMENSION A(21), B(21), R(21), S(21), N, N, KM(20), ALPHA(21)
COMPUTING AUTO CORRELATION OF SPEECH SAMPLES
M1=M+1
DO 1 K=1, M1
SUM=0
N1=N-K+1
DO 1 J=1, N1
J1=J+K-1
SUM=SUM+S(J)+S(J1)
R(K)=SUM
ALPHA(1)=R(1)
A(1)=1
KM(1)=R(2)/R(1)
R(2)=KM(1)
A1(1,1)=1
A1(1,2)=R(2)

ALPHA AND BETA HAVE SAME VALUES
ALPHA(2)=ALPHA(1)-(KM(1)**2)*ALPHA(1)

COMPUTING COEFFICIENTS OF BACKWARD PREDICTION FILTER
DO 2 I=2, N
DO 3 J=1, I
J1=I-J+1
B(J)=A(J1)
CONTINUE

COMPUTING NEW VALUES OF REFLECTION COEFFICIENT 'KM'
SUM=0
DO 4 K=1, I
IK=I+2-K
SUM=SUM+R(IK)*A(K)
KM(I)=SUM/ALPHA(I)

COMPUTING THE NEW COEFFICIENTS OF HIGHER ORDER FORWARD PREDICTION FILTER
I1=I+1
DO 5 L=2, I
A(L)=A(L)+KM(I)*B(L-1)
R(I1)=KM(I)

END
NEW VALUE OF ALPHA IS COMPUTED

\[ A(I) = A(I) - (K(I) * A(I))^2 * A(I) \]

DO 9 K=1, I
A(I,K)=A(K)
9 CONTINUE
IF(A(I) .LE. 0) GO TO 10
CONTINUE
GO TO 11
10 WRITE(10,13)
13 FORMAT(SX, 'INSUFFICIENT ACCURACY', \)
11 RETURN
END
********** SPEECH SYNTHESIS PROGRAM **********

PROGRAM NAME : RMAIN

DIMENSION ISS(200), X(200), NAME1(10), NAME(10), UDEL(150)
DIMENSION SS(200), DEL(200), ALPHA(9), A(8)
DATA DEL/200=0.
ACCEPT 'LENGTH OF THE PITCH FILE', IPL
ACCEPT 'ENTER THE NAME OF THE PITCH FILE'
READ(11,14) (NAME1(I), I=1,10).
FORMAT(10A12)
ACCEPT 'ENTER THE NAME OF THE SYN. FILE'
READ(11,12) (NAME(I), I=1,10)
FORMAT(10A12)
ACCEPT 'SCALING FACTOR', SC
ACCEPT '(0) FOR RECT/HAMM (I) FOR HAMM+PREMP', IFLAG
DEL(1)=1.0
TEMP=0.0
MFL=0
M=0
DO 77 I=1,8
SS(I)=0.0

SPEECH SAMPLES ARE READ HERE.

OPEN 0, 'FILELPC'
OPEN 1, 'FRAN'
OPEN 2, 'NAME1'
OPEN 3, 'NAME'
OPEN 1, 'ALPHA'

PITCH PERIOD OF EACH SEGMENT IS READ HERE.

KLN=0
KLN=KLN+1
IF(KLN.GT.IPL) GO TO 33
READ(2) IS
IF(IS.EQ.1) GO TO 2
READ(0) (A(I), I=1,8)
READ(4) ALPHA(9)

LINEAR PREDICTOR COEFFICIENTS OF ALL ZERO 8TH ORDER
INVERSE FILTER ARE COMPUTED HERE.

IF(IS.EQ.0) GO TO 3

SPEECH SYNTHESIS OF VOICED SEGMENT OF LENGTH ONE PITCH PERIOD
IS DONE HERE.

GAIN=SQRT(ALPHA(9))
DO 11 I=1,15
SUM=0
DO 1 II=1, M
K=I-II+8
SUM=SUM-A(I)*SS(K)
CONTINUE
SS(I+8)=SUM+DEL(I)*GAIN
CONTINUE

SYNTHESIZED SPEECH SEGMENT IS STORED HERE.

IF(IFLAG.NE.0)GO TO 95
DO 91 LK=1,IS
ISS(LK)=50*SS(LK+8)
GO TO 42
91 X(I)=.95*TEMP+SS(9)
DO 40 I=2,IS
X(I)=.95*X(I-1)+SS(I+8)
TEMP=X(IS)
DO 41 LK=1,IS
ISS(LK)=50*X(LK)
41 WRITE(3) (ISS(LL),LL=1,IS)
DO 73 I=1,8
SS(I)=SS(IS+I)
MPL=MPL+IS
TYPE MPL
GO TO 88

THIS IS SILENCE PART OF THE SYNTHESIZED SPEECH.

DO 1 LL=1,100
ISS(LL)=0
WRITE(3) (ISS(LL),LL=1,100)
TEMP=0.0
DO 71 I=1,8
SS(I)=0.0
MPL=MPL+100
TYPE MPL
GO TO 88

SPEECH SYNTHESIS OF UNVOICED SEGMENT OF LENGTH 100 SAMPLES IS
DONE HERE.

IS=100
GAIN=SQRT(ALPHA(9)/IS)

EXCITATION VALUES NEEDED FOR SYNTHESIS OF UNVOICED SEGMENT
OF SPEECH ARE READ HERE.

READ(1) (UDEL(I),I=1,IS)
DO 55 I=1,IS
SUM=0
DO 55 II=1,N
K=I-II+8
SUM=SUM-A(II)*SS(K)
55 CONTINUE
SS(I+8)=SUM(UDEL(I))*GAIN
CONTINUE
IF(IFLAG.EQ.0)GO TO 90
GO TO 95
CLOSE 0
CLOSE 1
CLOSE 2
CLOSE 3
CLOSE 4
STOP
END
SUBROUTINE NAME: GAUSS

IX - STARTING VALUE. ANY ODD INTEGER NUMBER WITH
     NINE OR LESS DIGITS.

S - THE DESIRED STANDARD DEVIATION OF THE NORMAL DISTRIBUTION.

AM - THE DESIRED MEAN OF THE NORMAL DISTRIBUTION.

V - THE VALUE OF THE COMPUTED RANDOM VARIABLE.

SUBROUTINE GAUSS(IX, S, AM, V)
A=0.0
DO 50 I=1,12
   CALL RANDU(IX, IY, Y)
   IX=IY
50   A=A+Y
   V=(A-.6)*S+AM
RETURN
END
SUBROUTINE NAME: RANUD

IX - STARTING VALUE , ANY ODD INTEGER WITH NINE OR LESS DIGITS.
IY - RESULTANT INTEGER NUMBER REQUIRED FOR THE NEXT ENTRY TO
     THIS SUBROUTINE.
YFL - THE RESULTANT UNIFORMLY DISTRIBUTED , FLOATING POINT,
     RANDOM NUMBER IN THE RANGE 0.0 TO 1.0

SUBROUTINE RANUD(IX, IY, YFL)
  IY=IX*16387
  IF(IY)<6.6
      IY=IY+32767+1
  YFL=IY
  YFL=YFL*3.05185E5
RETURN
END
**** SCALAR QUANTIZATION ROUTINE ****
PROGRAM NAME : QUANT

DIMENSION Y(1024), IY(S), X(S), A(S)
XMAX=1.0
XMIN=-1.0
DELTA=(XMAX-XMIN)/1024.0
DELTA2=DELTA/2.0
Y(S13)=DELTA2
Y(S12)=-DELTA2
SUM=DELTA
DO 1 I=1,511
Y(S13+I)=DELTA2+SUM
Y(S12-I)=-DELTA2-SUM
SUM=SUM+DELTA
DO 10 JJ=1,1024,S
IA=S
WRITE(10,11) (Y(I),I=JJ,IA)
FORMAT(1X,8(F8.4,1X))
IA=IA+S
CONTINUE
OPEN 1, 'FILEREF', LEN=52
OPEN 2, 'QUANTREF', LEN=52
DO 2 I=1,10000

LPC COEFFICIENTS ARE READ HERE

CALL READRN(1,1,X,L,IER)
IF(IER.EQ.9)GO TO 3

LPC'S ARE QUANTIZED HERE

CALL PAM(X,IY)
WRITE(10,16) (IY(JL), JL=1, S)
FORMAT(1X,8(I5,1X))
DO 4 JJ=1, S
L=IY(JJ)
A(JJ)=Y(L)

QUANTIZED LPC'S ARE STORED HERE

CALL WRITRN(2,1,A,L,IER)
CONTINUE
CLOSE 1
CLOSE 2
STOP
END
PROGRAM NAME: PAM

SUBROUTINE PAM(X, IY)
DIMENSION X(S), IY(S)
A=511.5
B=512.5
DO 1 I=1, S
1 IY(I)=A*X(I)+B
CONTINUE
RETURN
END
SUBROUTINE TO PERFORM CONVERSION
FROM REFLECTION COEFFICIENTS
TO LPC COEFFICIENTS

PROGRAM NAME : STEPUP

SUBROUTINE STEPUP(A, RC, M)
DIMENSION A(9), RC(8), B(9)
A(1)=1
A(2)=RC(1)
DO 30 MINC=2, M
DO 10 J=1, MINC
  JB=MINC-J+1
  B(J)=A(JB)
10  CONTINUE
DO 20 IP=2, MINC
  A(IP)=A(IP)+RC(MINC)+B(IP-1)
  A(MINC+1)=RC(MINC)
20  CONTINUE
RETURN
END
*************** PITCH DETECTION PROGRAM BASED ON ***********
******** AUTOCORRELATION METHOD ********
*************** PROGRAM NAME : PITCHDET ***************

INTEGER FLAG, STH, FRAME, 2N
REAL NUM
DIMENSION ISS(2000), NAME(10), NS(300), KOUNT(100)
DIMENSION IPER(2500), NAME1(10), D(5), S(300), R(250), NUM(4), DEN(4)
DATA D/5*0/
DATA NUM/.03570816, -0.00699362, -0.00899362, .03570816/
DATA DEN/1., -2.34656585, 2.01190019, -6.14109219/
ACCEPT 'ENTER THE NAME OF THE INPUT FILE'
READ(11, 12a) (NAME(I), I=1, 10)

FORMAT(10a2)
ACCEPT 'ENTER THE VALUE OF ZEROTHRESH ' , ZERO
ACCEPT 'ENTER THE VALUE OF THRESH ' , CTHRESH
ACCEPT 'ENTER THE VALUE OF AUTO THR ' , ATH
ACCEPT 'STARTING SPEECH SAMPLE NO. ' , IS
ACCEPT 'LENGTH OF THE INPUT FILE IS ' , LENGTH
ACCEPT 'FLAG FOR SILENCE ' ,FLAG
ACCEPT 'FRAME # ' , FRAME
TYPE 'ENTER THE PITCH FILENAME'
READ(11, 500) (NAME1(I), I=1, 10)

500 FORMAT(10a2)
KKK=0
OPEN 5, NAME1
CALL SILEN(STH, MAG)
DO 1 J=1, 2500

445 IF(IS.GT.LENGTH) GO TO 59
OPEN 0, NAME, LEN=2
CALL READRN(0, IS, NS, 300, IER)
CALL CHECK(<IER)
CLOSE 0
DO 66 III=1, 300
S(III)=FLOAT(NS(III))

66 CHECK FOR SILENCE PORTION OF THE SPEECH

IF(KKK.EQ.1) GO TO 38
SUM=0
SUM1=0
DO 222 III=1, 200
SUM1=SUM1+ABS(S(III))
SUM=SUM+ABS(S(III-100))
SNAG=AMINI(SUM, SUM1)
IF(SNAG .GE. FLOAT(MAG)) GO TO 221
FLAG=0
FRAME=FRAME+1
WRITE(10, 322) FRAME

222 FORMAT('0', 2x, 'SEGMENT NO. ', I4, 2x, 'IS SILENCE')
IS=IS+100
IPER(FRAME)=1
WRITE(9) IPER(FRAME)
GO TO 1

221 IF(FLAG.EQ.1)GO TO 30
DO 2 I=1,100
IF(ABS(S(I)).LT.FLOAT(STH))GO TO 2
IS=IS+I-1
FLAG=1
KKK=1
WRITE(10,301) IS+1
221 FORMAT(3F12.2), 'STARTING SPEECH SAMPLE # OF NEXT FRAME IS ', IS
GO TO 145
CONTINUE

SETTING THE CLIPPING THRESHOLD

223 KKK=0
IF(S(I).GE.0)KOUNT(I)=1
IF(S(I).LT.0)KOUNT(I)=-1
ZN=0
DO 777 I=1,99
IF(S(I+1).GE.0)KOUNT(I+1)=1
IF(S(I+1).LT.0)KOUNT(I+1)=-1
777 ZN=ZN+ABS(KOUNT(I)-KOUNT(I+1))
ZN=ZN/2
IF(ZN.GE.ZERO)GO TO 90
DO 888 I=1,300
XIN=S(I)

HERE THE SPEECH SAMPLES ARE FILTERED BY DIGITAL LOW-PASS FILTER WITH A CUTOFF AT 900KHz.

C08 CALL DIRECT(DEN, NUM, 3, D, XIN, OUT)
S(I)=XOUT
RMAX=S(I)
DO 6 I=1,100
IF(S(I).GE.RMAX) RMAX=S(I)
6 CONTINUE
RMAX1=S(200)
DO 7 I=200,300
IF(S(I).GE.RMAX1) RMAX1=S(I)
7 CONTINUE
CLIP=AMIN1(RMAX, RMAX1)*CThRES
SLIP=-CLIP

3 LEVEL CENTRE CLIPPING OF THE SPEECH SAMPLES

DO 8 I=1,300
IF(S(I).GE.CLIP)GO TO 41
IF(S(I).LE.SLIP)GO TO 40
S(I)=0.
8 GO TO 8

41 S(I)=1.
GO TO 8

43 S(I)=-1.
CONTINUE

PITCH DETECTION SCHEME USING AUTO CORRELATION METHOD

DO 9 I=50,150
   SUM=0
   N11=300-I+1
   DO 11 I=1,N11
      SUM=SUM+S(I)*S(I+I-1)
   END
   R(I)=SUM
   SUM=0
   DO 10 I=1,300
      SUM=SUM+S(I)*S(I)
   END
   R(1)=SUM
   VTHR=ATH*R(1)
   RMAX=R(50)
   DO 11 I=50,150
      IF(RMAX .GE. R(I)) GO TO 11
      RMAX=R(I)
      IP=I-1
   END
   CONTINUE
   IF(RMAX .GE. VTHR) GO TO 55
9 CONTINUE
   FRAME=FRAME+1
   TYPE=2
   WRITE(10,100) FRAME
   FORMAT(0.2X,'SPEECH SEGMENT NO.',I4,2X,'ANALYZED IS UNVOICED')
   IS=IS+100
   IPER(FRAME)=0
   GO TO 300
55 CONTINUE
   WRITE(10,100) FRAME
   FORMAT(0.2X,'SPEECH SEGMENT NO.',I4,2X,'ANALYZED IS VOICED')
   WRITE(10,55) IP
   FORMAT(0.2X,'PITCH PERIOD OF THIS SEGMENT =',I3)
   IS=IS+IP
   IPER(FRAME)=IP
300 CONTINUE
   IPER(FRAME)
END
*************** THIS SUBROUTINE COMPUTES THE THRESHOLDS ********
********* FOR SILENCE PART AND SILENCE TO SPEECH ********
********* TRANSITION OF SPEECH ********
*****************************************
PROGRAM NAME : SILEN

SUBROUTINE SILEN(STH, MAG)
INTEGER S(500), STH, SUM
DIMENSION NAME(10)
ACCEPT 'NAME OF THE INPUT FILE'
READ(11, 11) (NAME(I), I=1,10)
11 FORMAT(10A2)
ACCEPT 'LENGTH OVER ABS. PEAK IS SEARCHED ', IP
ACCEPT 'LENGTH OVER WHICH THRESH COMPUTED ', LEN
OPEN 1, NAME
READ(1) (S(I), I=1, IP)
CLOSE 1
SUM=ABS(S(I))
DO 2 I=1, IP
IF(SUM.GT. ABS(S(I))) GO TO 2
SUM=ABS(S(I))
2 CONTINUE
STH=SUM+17
TYPE STH
SUM=0
DO 3 I=1, LEN
SUM=SUM+ABS(S(I))
3 CONTINUE
MAG=SUM
TYPE MAG
RETURN
END
SUBROUTINE DIRECT(A, P, M, D, XIN, XOUT)
DIMENSION A(4), P(4), D(5),
XOUT=0.0
D(1)=XIN
DO 1 J=1, M
  JJ=M+1-J
  XOUT=XOUT+D(JJ+1)*P(JJ+1)
  D(J+1)=D(J+1)-A(JJ+1)*D(JJ+1)
  D(JJ+1)=D(JJ)
  XOUT=XOUT+D(1)*P(1)
1  RETURN
RETURN
END
*** THIS PROGRAM COMPUTES THE AVERAGE LPC (PHONEME) VECTORS AND STORES THEM IN A FILE ***
PROGRAM NAME : AVGLPC

DIMENSION S(300), SUM(S), X(S)
ACCEPT 'STARTING RECORD # ', NREC
ACCEPT '# OF RECORDS TO BE READ ', NREC
OPEN 0, 'FILELPC', LEN=32
CALL READRN(0, NREC, S, NREC, IER)
CALL CHECK(IER)
CLOSE 0
DO 1 I=1, S
  SUM(I)=0.0
  N=NREC*S
DO 2 I=1, N, S
  II=I+N
  DO 3 K=I, II
    L=(K-I+1)
    SUM(L)=SUM(L)+S(K)
  CONTINUE
DO 4 I=1, S
  X(I)=SUM(I)/NREC
WRITE(10, 8) X(I), I=1, S
FORMAT('0', 20X, F10.7, 2X, F10.7, 2X, F10.7, 2X, F10.7)

CODENW ID OF THE PHONEME CODEBOOK IS STORED HERE

OPEN 1, 'CODEBOOK', LEN=32
ACCEPT 'STARTING RECORD # OF THE CODEBOOK ', NREC
CALL WRITRN(1, NREC, X, 1, IER)
CALL CHECK(IER)
CLOSE 1
STOP
END
*************** THIS PROGRAM MAPS THE INPUT LPC VECTORS **********
********** INTO THEIR NEAREST NEIGHBOUR CODEWORD **********
******************************************************************************

PROGRAM NAME : VECT.

REAL MIN
DIMENSION ERROR(64), NAME1(10), SS(8), A(8), NAME(10)
ACCEPT 'LENGTH OF THE PITCH ARRAY' , IB
NREC=1
ACCEPT 'NAME OF THE PITCH FILE'
READ(11,88) (NAME(I), I=1,10)
88 FORMAT(10A2)
OPEN 2, NAME

BLOCK QUANTIZATION OF LPC COEFFICIENTS IS INITIATED

ACCEPT 'NAME OF THE CODE FILE'
READ(11,7) (NAME(I), I=1,10)
7 FORMAT(10A2)
OPEN 3, NAME
TYPE 'NAME OF THE CODEBOOK'
READ(11,41) (NAME1(I), I=1,10)
41 FORMAT(10A2)
ACCEPT 'ENTER THE # OF VECTORS IN THE CODE BK' , INUM
OPEN 1, 'FILELPC', LEN=32
OPEN 0, NAME1, LEN=32
COUNT=0, 0
DIST=0, 0
DO 5, II=1, IB
READ(2) IS
IF(IS .EQ. 1) GO TO 8
CALL READR1(1, NREC, SS, 1, IER)
DO 2 I=1, INUM

READ ONE RECORD FROM THE CODEBOOK

CALL READR0(0, I, A, 1, IER)
CALL CHECK(IER)

COMPUTE THE MEAN SQUARE ERROR

ERROR(I)=0
DO 3 K=1, 8
ERROR(I)=ERROR(I)+(SS(K)-A(K))**2
3 CONTINUE

FINDING THE NEAREST VECTOR IN THE CODEBOOK WHICH MATCHES THE GIVEN SPEECH VECTOR

IP=1
MIN=ERROR(1)
DO 6 L=2, INUM
IF(ERROR(L).GE.MIN) GOTO 6
MIN=ERROR(L)
6 CONTINUE
IP=L
CONTINUE
DIST=DIST+MIN
COUNT=COUNT+1
WRITE(3) IP
NREC=NREC+1
CONTINUE
DIST=DIST/COUNT
WRITE(12,12) NAME, DIST
FORMAT(1X, 'CODEBOOK NAME IS--', 10A2, 'AVERAGE MSE = ', F10.6)
CLOSE 0
CLOSE 1
CLOSE 2
CLOSE 3
STOP
END
PROGRAM FOR GENERATING A CODEBOOK USING SPLITTING TECHNIQUE

PROGRAM NAME : GRAYPA

REAL MIN
INTEGER POINTER
DIMENSION F(64,8), Z(512), Z1(256), Y(8), S(8), X(8), ERROR(64)
DIMENSION NAME(10), COUNT(64), NAME1(10).
ACCEPT 'TOTAL LENGTH OF TRAINING SEQUENCE ', ITRAN
ACCEPT 'VALUE OF THRESHOLD VALUE ', THR
ACCEPT '# OF BITS PER VECTOR ', ISL
DO 98 I=1,8
22 Z(I)=0.0
    TYPE 'INPUT FILE NAME'
    READ(11,102) (NAME(I), I=1,10)
102 FORMAT(10A2)
    TYPE 'OUTPUT FILE NAME'
    READ(11,103) (NAME1(I), I=1,10)
103 FORMAT(10A2)
    OPEN 0, NAME, LEN=32
    OPEN 1, NAME1, LEN=32
DO 99 I=1,ITRAN
    CALL READRN(0.1,5,1,IER)
98     K=1.8
23    Z(K)=Z(K)+S(K)
22    CONTINUE
99     K=1.8
23    Z(K)=Z(K)/ITRAN
22    TYPE Z(K)
22    Z(K)=Z(K)-1
22    Z(K+8)=Z(K)+.2
    CALL WRITRN(1,1,2,2,IER)
DO 96 IC=1,ISL
    IL=2**IC
    DOLD=(<9.99)**50
    NREC=1
DO 12 KLM=1,1000
    DNEW=0.0
0     FIND PARTITIONS OF THE INPUT SPEECH DATA
DO 7 L=1,IL
    DO 8 M=1,8
        P(L,M)=0.0
7     COUNT(L)=0.0
8      OPEN 2, 'POINT64'
    OPEN 3, 'POINT32'
DO 2 I=1,ITRAN
    CALL READRN(0.1,5,1,I4)
2      J=1,IL
    CALL READRN(1,1,5,1,I4)
8      ERROR(J)=0.0
7      DO 4 K=1,8
8      ERROR(J)=ABS(S(K)-X(K))+ERROR(J)
7     CONTINUE
MIN-ERROR(1)
POINTER=1
FIND THE MINIMUM ERROR VECTOR
DO 5 K=2, IL
   IF(MIN. LE. ERROR(K)) GOTO 5
   MIN=ERROR(K)
   POINTER=K
   CONTINUE
   IF(IL .EQ. 32) WRITE(3) POINTER
   IF(IL .EQ. 64) WRITE(2) POINTER
   DNEN=DNEN+MIN
   DO 9 N=1, 8
      P(POINTER, N)=P(POINTER, N)+S(N)
      COUNT(POINTER)=COUNT(POINTER)+1
   CONTINUE
   CHECK THE DISTORTION CRITERION
   CLOSE 2
   CLOSE 3
   DNEN=DNEN/ITRAN
   TEMP=(DOLD-DNEN)/DNEN
   TYPE 'ITERATION # ', NREC
   TYPE 'NEW VALUE OF DISTORTION ', DNEN
   NREC=NREC+1
   DOLD=DNEN
   IF(IL .LT. 32) GOTO 50
   WRITE(12, 51) DNEN, IL
   FORMAT(1X, F10. 7, 2X, 12)
51   NOW TO COMPUTE THE CENTROID OF ALL PARTITIONS
50   IF(TEMP .LT. THR) GOTO 10
   DO 10 I=1, IL
   IF(P(I, 1). EQ. 0) GOTO 10
   DO 11 K=1, 8
      V(K)=P(I, K)/COUNT(I)
   11   CALL WRITRN(1, I, V, 1, IER)
   CONTINUE
   TYPE 'CODEBOOK LEVEL IS ', IL
   CONTINUE
40   IF(IC .EQ. ISL) GOTO 41
   CALL READRN(1, 1, 21, IL, IER)
   IF(IL .NE. 32) GOTO 52
   OPEN 4, 'BLOCK32', LEN=32
   CALL WRITRN(4, 1, 21, IL, IER)
   CALL CHECK(IER)
   CLOSE 4
52   IA=1
   IB=2
   IX=0
   DO 89 KK=1, IL
      DO 88 KO=IA, IB
         Z(KO)=Z1(KO-IX)-.005
95      Z(KO+8)=Z1(KO-IX)+.005
      IA=IA+16
      IB=IB+16
   IX=IX+8
ID=IL+2
CALL WRITRM(1, 1, Z, ID, IER)
CONTINUE
CLOSE 0
CLOSE 1
STOP
END
REAL MIN
INTEGER POINTER
DIMENSION XNAME(64), YNAME(64)
DIMENSION P(32, 8), Z(256), XI(128), Y(8), S(8), X(8), ERROR(32), COUNT(32)
ACCEPT 'SELECT THE MAX. # OF LEVELS ', ISL
ACCEPT '# OF FILES TO BE ENTERED ', INUM
ACCEPT 'ENTER FILE NAMES SET 1'
DO 51 J=1, INUM
READ(11, 52) XNAME(J)
52 FORMAT(A4)
CONTINUE
DO 53 J=1, INUM
WRITE(10, 54) XOM
54 FORMAT(1X, A4)
CONTINUE
ACCEPT 'ENTER FILE NAME SET 2'
DO 56 I=1, INUM
READ(11, 55) YNAME(I)
55 FORMAT(A4)
CONTINUE
DO 57 I=1, INUM
WRITE(10, 53) YOM
53 FORMAT(1X, A4)
CONTINUE
USING SPLITING METHOD
ACCEPT 'VALUE OF THRESHOLD VALUE ', THR
CLUSTERING ALGORITHM WORKS ON THE SUB-SETS OF TRAINING SEQUENCE OF SPEECH LPC VECTORS
DO 60 JJ=1, INUM
FILE WHICH CONTAINS LPC VECTORS OF A PARTITION CELL IS OPENED HERE

TYPE 'FILE IN PROCESS IS', JJ
XOM=XNAME(JJ)
YOM=YNAME(JJ)
OPEN 0, XOM, LEN=32
OPEN 1, YOM, LEN=32
DO 99 I=1, 8
Z(1)=0.0
DO 99 I=1, 10000
CALL READRN(0, I, S, L, IER)
IF (IER.EQ.9) GO TO 100
DO 98 K=1, 8
   Z(K)=Z(K)+S(K)
99 CONTINUE
100 ITRAN=I-1
   TYPE 'NO OF RECORDS = ', ITRAN
DO 97 K=1, 8
   Z(K)=Z(K)/ITRAN
   TYPE Z(K)
   Z(K)=Z(K)-.1
   Z(K+8)=Z(K)+.2
   CALL WRITRN(L,1, Z, 2, IER)
DO 96 IC=1, ISL
   IL=2*IC
   DOLD=(<9, 99)+*50)
   NREC=1
DO 12 KLM=1, 1000
   DNEN=0. 0

FIND PARTITIONS OF THE INPUT SPEECH DATA

DO 7 L=1, IL
DO 8 M=1, IL
   C(L,M)=0. 0
DO 2 I=1, ITRAN
   CALL READRN(0, I, S, L, IER)
   CALL CHECK(IER).
   DO 3 J=1, IL
   CALL 'READRN(1, J, X, L, IER)
   CALL CHECK(IER)
   ERROR(J)=0. 0
   DO 4 K=1, 8
   ERROR(J)=ABS(S(K)-X(K))+ERROR(J)
   CONTINUE
   MIN=ERROR(1)
   POINTER=1

FIND THE MINIMUM ERROR VECTOR

DO 5 K=2, IL
   IF (MIN .LE. ERROR(K)) GO TO 5
   MIN=ERROR(K)
   POINTER=K
   CONTINUE
   DNEN=DNEN+MIN
   DO 9 N=1, 8
   P(POINTER, N)=P(POINTER, N)+S(N)
   COUNT(POINTER)=COUNT(POINTER)+1
   CONTINUE
   CHECK THE DISTORTION CRITERION
   DNEN=DNEN/ITRAN
   TEMP=(DOLD-DNEN)/DNEN
   NREC=NREC+1
   TYPE 'ITER #', NREC, ’LEVEL #’, IL, ’DISTORTION=’, DNEN
DOLD=DNEW
IF(TMP. LE. THR) GO TO 40

NOW TO COMPUTE THE CENTROID OF ALL PARTITIONS.

DO 10 I=1, IL
IF(P(I,1).EQ.0) GO TO 10
DO 11 K=1, 8
P(I,K)=P(I,K)/COUNT(I)
11 Y(K)=P(I,K)
CALL WRITRN(1, I, Y, 1, IER)
10 CONTINUE
12 CONTINUE
GO TO 61
CALL READRN(1, 1, Z1, IL, IER)
IA=1
IB=8
IX=0
DO 28 KK=1, IL
DO 28 KO=IA, IB
Z(KO)=21(KO-IX)-.035
28 Z(KO+8)=Z1(KO-IX)+.035
IA=IA+16
IB=IB+16
IX=IX+8
ID=IL+2
CALL WRITRN(1, 1, Z, ID, IER)
28 CONTINUE
61 CLOSE 0
CLOSE 1
TYPE 'FILE #1, IJ,' IS COMPLETED'
62 CONTINUE
STOP
END
**SPEECH SYNTHESIZED FROM VECTOR**

**QUANTIZED LPC COEFFICIENTS**

**PROGRAM NAME**: TVECSYN

```
DIMENSION NAME(10), C(S), ISS(20000), ALPHA(9), UDEL(100), SS(200)
DIMENSION NAME(10), DEL(200), R(S), B(S), WX(200)
DATA DEL/200+0./
ACCEPT 'NAME OF THE PITCH FILE'
READ(11, 14) (NAME(i), i=1,10)
FORMAT(10A2)
OPEN 0, NAME
ACCEPT 'LENGTH OF THE PITCH ARRAY ', IB
ACCEPT 'NAME OF CODED FILE'
READ(11, 111) (NAME(I), i=1,10)
FORMAT(10A2)
ACCEPT 'NOT FOR PREV. FR. ', YY
ACCEPT 'NOT FOR CURR. FR. ', XX
OPEN 1, NAME

ENTER # OF CODEWORDS IN THE CODEBOOK

ACCEPT '# OF VECTORS IN 1ST LEVEL C. BK ', INUM
ACCEPT '# OF VECTORS IN 2ND LEVEL C. BK ', ISL
ACCEPT 'SCALING FACTOR ', SC
TYPE 'NAME OF THE CODEBOOK'
READ(11, 15) (NAME(I), i=1,10)
FORMAT(10A2)

SELECT RECT. OR HAMMING WINDOW

ACCEPT ' (0) FOR RECT & (1) FOR HAMMING OPER. ', IFLAG
TYPE 'NAME OF THE SYN. FILE'
READ(11, 107) (NAME1(I), i=1,10)
FORMAT(10A2)
DEL(1)=1.0
KP=0
TEMP=0.0
IFLAG=0
N=8
DO 77 I=1, S
77 SS(I)=0
OPEN 2, 'FRAN'
OPEN 3, 'ALPHA'
OPEN 4, NAME, LEN=32
DO 88 KLM=1, IB
READ(2) IS
IF(IS.EQ.1) GO TO 2

COMPUTE THE LOCATION OF THE REPRODUCTION VECTOR

READ(1) IX
READ(1) IV
NREC=INUM+(IX-1)*ISL+IV
```
IF(IALOG.EQ.0)GO TO 41
CALL READR(4, NREC, A, 1, IER)
DO 30 I=1,8
C(I)=XX*AC(I)+YY*B(I)
30 B(I)=A(I)
GO TO 31
CALL READR(4, NREC, C, 1, IER)
DO 40 I=1,8
40 B(I)=C(I)
READ(3) ALPH(A(9))
IFLAG=1
IF(IS.EQ.0) GO TO 3
GAIN=SRT(ALPH(A(9)))
DO 11 I=1,IS
SUM=0
DO 1 II=1,M
K=I-II+8
SUM=SUM+C(II)*SS(K)
1 CONTINUE
SS(I+8)=SUM+DEL(I)*GAIN
CONTINUE
IF(IALOG.EQ.0)GO TO 90
GO TO 95
2 DO 4 LL=1,100
3 ISS(LL+KP)=0
DO 7 I=1,8
7 SS(I)=0
KP=KP+100
IFLAG=0
TEMP=0.0
TYPE KP
GO TO 88
3 IS=100
GAIN=SRT(ALPH(A(9))/IS)
READ(2) (UDEL(I), I=1,IS)
DO 55 I=1,IS
SUM=0
DO 56 II=1,M
K=I-II+8
SUM=SUM+C(II)*SS(K)
55 CONTINUE
SS(I+8)=SUM+UDEL(I)*GAIN
CONTINUE
IF(IALOG.EQ.0)GO TO 95
2 DO 91 LK=1,IS
3 ISS(LK+KP)=SC*SS(LK+8)
GO TO 42
0 OPERATION OF DE-EMPHASIS ON THE SYNTHESIZED
SPEECH SAMPLES
95 NW(1)=.95*TEMP+SS(9)
DO 400 I=2,IS
400 NW(I)=.95*NW(I-1)+SS(I+8)
TEMP=NW(IS)
DO 410 LK=1, IS
   ISS(LK+KP)=50*WX(LK)
410
DO 73 I=1, 8
   SS(I)=SS(IS+I)
   KP=KP+IS
   TYPE KP
   CONTINUE
   CLOSE 1
   CLOSE 2
   CLOSE 1
   CLOSE 0
   OPEN 0, NAME=1

DE-EMPHASIZED SPEECH SAMPLES ARE STORED HERE

WRITE(0)'(ISS(I), I=1,KP)
CLOSE 0
STOP
END
PROGRAM TO COMPUTE
*** THE SIGNAL-TO-QUANTIZATION NOISE RATIO ***
*** FOR A CODEBOOK ***

PROGRAM NAME : SQNR

DIMENSION IS1(2500),NAME2(10),NAME1(10),NAME1(10),IS2(2500)
DIMENSION SI(2500),S2(2500)
TYPE 'UNQUANTIZED FILE NAME'
READ(11,1) (NAME1(I),I=1,10)
FORMAT(10A2)
TYPE 'QUANTIZED FILENAME NAME'
READ(11,1) (NAME2(I),I=1,10)
ACCEPT '# OF POINTS PER RECORD ',IB
ACCEPT '# OF RECORDS ',NREC
ACCEPT 'IDENTITY OF THE DATA FILE'
READ(11,1) (NAME2(I),I=1,10)
IC=IB+2
SUM=0.0
SUM1=0.0
OPEN 1,NAME,LEN=IC
OPEN 2,NAME1,LEN=IC
DO 3 I=1,NREC
CALL READWK1(1,1,IS1,L1ER)
CALL READWK2(1,2,IS2,L1ER)
DO 4 J=1,IB
SI(J)=FLOAT(IS1(J))
S2(J)=FLOAT(IS2(J))
SUM=SUM+SI(J)+S2(J)
SUM1=SUM1+(SI(J)-S2(J))**2
3 CONTINUE
COMPUTING SQNR

D=SUM/SUM1
SNR=10.0*LOG10(D)
WRITE(12,5) (NAME2(I),SNR)
FORMAT(1X, 'OPER. NAME IS ',10A2,2X, 'SNR = ',F10.6, 'DB')
CLOSE 1
CLOSE 2
STOP
END
REFERENCES


VITA AUCTORIS

Sumesh Kaul

1960  Born on 13th of May in Srinagar, Kashmir, India

1975  Completed High School Education at C. M. S.
      Tyndale Biscoe School, Srinagar.

1976  Completed Pre-University Education at Amar
      Singh College, Srinagar.

1981  Completed the Degree of B.E (in Electronics
      & Communications) at Regional Engineering
      College, Madras University, India.

1982  Served as Marketing Engineer in Bio-Medical
      Division at Blue Star Ltd, New Delhi, India.

1984  Candidate for the Degree of Master of Applied
      Science (Electrical Engg.) at the University
      of Windsor, Windsor, Ontario, Canada.