"BUFFER MANAGEMENT FOR VARIABLE-LENGTH ENCODING OF SPEECH"

A project submitted by

NG SZE-CHIU

(Stud. No. 7819075)

To: the Department of Electrical Engineering for the partial fulfillment of the requirements for the Degree of Master of Engineering

January, 1981

Faculty of Engineering

McGill University

Montreal, Quebec

Canada

ABSTRACT

This report studies three methods to solve the buffer overflow problem introduced by using variable-length codes in digital processing of speech signals. Two feed-back coders and one feed-forward coder are proposed. The feedback coders perform adaptive quantization, variable-length encoding and buffer management. The feed forward coder performs quantization, variable-length encoding and the search for the right quantization step-size. The first feedback coder exploits the dual function of the expansion-contraction factors associated with an adaptive quantizer. These multipliers are used not only to track the change in input standard deviation, but also to adjust the bit rate into the transmission buffer. The other feedback coder controls the bit rate into the buffer by changing the quantization step-size. The feedforward coder takes a vector of samples and adjust the quantization step-size until the total number of bits required to code this vector is approximately equal to a given value. The design, simulation and performance of the proposed coders for speech signals transmitted over a telephone network are then discussed. Comparisons of the proposed coders and an adaptive non-uniform quantizer with fixed length encoding are also presented. The report concludes with the subjective performance of the coders and the trade-off between complexity and performance.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Peter Kabal for his numerous discussions, helpful ideas and guidance in the realization of this work.

Special thanks are also extended to Betty Ann Woods and France Tessier for their typing of this report.

TABLE OF CONTENTS

Abstract.....i
Acknowledgements.....ii
Table of Contents.....iii
List of Tables.....v
List of Figures.....vi

CHAPTER 1 INTRODUCTION.....1

CHAPTER 2 REVIEW OF WAVEFORM CODERS.....3

2.1 Evolution.....3
2.2 Problems in Variable-Length Encoding.....17

CHAPTER 3 BETTER MANAGEMENT.....21

3.1 Adaptive Expansion-Contraction Factors.....24

3.1.1 Model.....24
3.1.2 Adaptive Scheme.....28

3.2 Adaptive Quantization Step-Size.....36

3.2.1 Model.....36
3.2.2 Adaptive Scheme.....40

3.3 Block quantization.....44

3.3.1 Model.....44
3.3.2 Buffer Management Scheme.....47

CHAPTER 4 MODEL SIMULATION AND RESULTS.....53

4.1 Experimental Set Up.....53
4.2 Simulation Results.....55

4.2.1 Adaptive Expansion-Contraction Factors.....60
4.2.2 Adaptive Quantization Step-Size.....82
4.2.3 Block Quantization.....91

4.3 Comparisons.....101

CHAPTER 5 CONCLUSIONS.....106

TABLE OF CONTENTS cont'd

REFERENCES.....108

APPENDICES PROGRAM LISTINGS.....112

A. Adaptive Expansion-Contraction Factors Module.....112
B. Adaptive Quantization Step-Size Module.....128
C. Block Quantization Module.....144
D. Common Block Module.....170
E. Utility Programs Module.....174

LIST OF TABLES

Table 2.1 Comparison of Objective and Subjective Performance of DPCM-AQ and Log-PCM with μ=100. (After Commiskey, Jayant and Flanagan (11)). [p.13]

Table 4.1 Average STD. [p.55]

Table 4.2 Rescaled STDs of SRFMT, SRFMP, FA3 and MB3 for Coder Simulation. [p.64]

Table 4.3 Optimum Coder Parameters with SRFMT as input Tr=16 kb/s, Sr=6.5 kSample/s and N=19. [p.69]

Table 4.4 T_1,T_2, Buffer Size & Delay with Tr=16kb/s, Sr=6.5 kSample/s and N=19. [p.71]

Table 4.5 Coder Performance with Tr=16kb/s, Sr=6.5 kSample/s, N=19 and INPUT=SRFMT. [p.73]

Table 4.6 Optimum γ & $\delta_{\Delta x1}$ with Tr=16 kb/s, Sr=6.5 kSample/s, N=19, Δ_{xmax} =2, Δ_{xmin} =0.4, $\tilde{\sigma}_{min}$=0.0113 & $\tilde{\sigma}_{max}$ =V_{max} =20. [p.83]

Table 4.7 T_1, T_2, Buffer Size & D_{max} with Tr=16 kb/s, Sr=6.5 kSample/s & N=19. [p.84]

Table 4.8 Coder Performance with Tr=16 kb/s, Sr=6.5 kSample/s, N=19 & Input=SRFMT. [p.85]

Table 4.9 Block Quantization Coder Performance with M=300 samples, N=19, Δ_{xmax} =40, Δ_{xmin} =0.00452, Tr=16 kb/s & Sr=6.5 kSample/s. [p.97]

Table 4.10 Optimum Parameters for ANQ+FLC Coder. [p.102]

Table 4.11 Comparisons of Coder Performance. [p.104]

LIST OF FIGURES

Figure 2.1 PCM System. [p.4]

Figure 2.2 (a) An N-Level Quantizer, (b) The "Q[.]" Operation and (c) The "Q^{-1}[.] Operation. [p.4]

Figure 2.3 DPCM System. [p.5]

Figure 2.4 DPCM System with both Adaptive Predictor and Adaptive Quantizer (a) Coder & (b) Decoder. [p.7]

Figure 2.5 Real Speech and Theoretical Gamma and Laplace Probability Densities (After Paez and Glissor (5)). [p.10]

Figure 2.6 Difference Signal $\mathscr{e}_k$ and Theoretical Gamma and Laplace Probability Densities (After Paez and Glisson (5)). [p.10]

Figure 2.7 N-Level Quantizer with Adaptation Multipliers. [p.11]

Figure 2.8 DPCM System with Adaptive Quantization (DPCM-AQ). [p.12]

Figure 2.9 DPCM with Adaptive Quantization and Entropy Encoding (DPCM-AQ+EC). [p.16]

Figure 2.10 Self-Synchronizing Hoffman codes for a 19-level quantizer. [p.19]

Figure 3.1 Block Diagram of Coder with Buffer Management. [p.27]

Figure 3.2 (a)ΔB verse BFOC for Increasing Buffer Contents. [p.34]
(b)ΔB verse BFOC for Decreasing Buffer Contents. [p.34]

Figure 3.3 Block Diagram of Coder with Buffer Management Scheme Using Adaptive Quantization Step-size. [p.37]

Figure 3.4 (a) $\delta_{\Delta x}$ verse BFOC for Increasing Buffer Contents. [p.41]
(b) $\delta_{\Delta x}$ verse BFOC for Decreasing Buffer Contents. [p.41]

Figure 3.5 Block Diagram of Coder with Buffer Management Scheme Using the Block Quantization Concept. [p.46]

Figure 3.6 B_N verse Δ_x. [p.49]

Figure 3.7 Quantization Step-Size verse Granular Distortion. [p.49]

LIST OF FIGURES (Cont'd)

Figure 4.1 (a), (b) & (c) Waveform of SRFMT. [p.58]

Figure 4.2 SNR verse Input Power Level in dB Relative to Overload Point V with Laplacian Input pdf for 7-bit μ-Law PCM with μ=255. [p.65]

Figure 4.3 Periodic Testing Square Wave. [p.70]

Figure 4.4 (a),(b) & (c) Decoded Output at Receiver verse Time with $M_i=(0.95975+B_k(i-1)^2)^{0.5}$; $k\geq 2$, $B_1= 0.01$. [p.75]

Figure 4.5 (a),(b) & (c) BFOC verse Time with $M_i=0.982+B'_k (i-1)^2$; $k\geq 2$, $B'_1 = 0.0045$. [p.77]

Figure 4.6 (a),(b) & (c) SLISNR verse Time with $M_i= 0.982+B'_k (i-1)^2$; $k\geq 2$, $B'_1=0.0045$. [p.78]

Figure 4.7 (a),(b) & (c) BFOC verse Time with $M_i=[0.095975+B_k (i-1)^2]^{0.5}$; $k\geq 2$, $B_1= 0.01$. [p.80]

Figure 4.8 (a),(b) & (c) SLISNR verse Time with $M_i=[0.95975+B_k (i-1)^2]^{0.5}$; $k\geq 2$, $B_1= 0.01$. [p.81]

Figure 4.9 (a),(b) & (c) Decoded Output at Receiver verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size. [p.87]

Figure 4.10 (a),(b) & (c)BFOC verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size. [p.89]

Figure 4.11 (a),(b) & (c) SLISNR verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size. [p.90]

Figure 4.12 (a) Flow-Chart of Main Program. [p.95]
(b) Flow-Chart of Subroutine SII, Calculate [Δ_{x1}, 2 Δ_{x1}]. [p.95]
(c) Flow-Chart of Subroutine SDELXK, Calculate Δ_{xk}. [p.95]

Figure 4.13 SEGSNR verse Vector Dimension M. [p.96]

Figure 4.14 (a),(b) & (c) Decoded Output at Received verse Time with Buffer management Achieved by Block Quantization. [p.99]

Figure 4.15 (a),(b) & (c) SLISNR verse Time with Buffer Management Achieved by using Block Quantization. [p.100]

CHAPTER 1 INTRODUCTION

The study is motivated by the problems caused by using variable-length encoding of speech signals in differential pulse-coded-modulation. Differential pulse-code-modulation (DPCM)[1] systems and pulse-code-modulation (PCM)[1] systems are widely used in speech transmission. In Canada, U.S.A. and Japan, 7-bit and 8-bit μ-Law PCM systems with $\mu = 255$ is used extensively in telephone speech transmission[2]. Variable-length encoding is a source coding technique. Longer code-words are assigned to less probable source symbols and shorter code-words are assigned to more probable source symbols. In this way, the coding efficiencycan be improved if the source symbols occur with an unequal probability. Variable-length encoding can be done either in a sample by sample basis or block by block basis. That is we can encode the source outputs one by one or encode a sequence of source outputs one at a time. The block by block variable-length encoding can always give a better coding efficiency than the sample by sample scheme. As the block length become larger and larger, we can generate a code such that the coding efficiency are almost 1 if the outputs are ergodic. See Gallager [3] for a treatment of source models and source coding. In our study here, the probability distribution of the source outputs, the quantizer outputs, allows us to generate a code on a sample by sample basis with a coding efficiency better than 0.95. Therefore we do not elaborate to generate a code on a block by block basis. Using variable-length encoding can improve the coding efficienty, however, this can also cause problems with

(i) buffer overflow,

(ii) buffer underflow,

(iii) code-word synchronization problem and

(iv) transmission delay

This study investigates these problems and proposes new methods to solve the buffer overflow problem.

This report is organized in the following way. In the first subsection of Chapter 2, we discuss evolution of wave-form coders. In the second subsection, we discuss

(i) the cause for the problems introduced by using variable-length encoding

(ii) the method to solve buffer underflow and code-word synchronization problem and,

(iii) the relationship between buffer overflow and transmission delay.

In Chapter 3, we propose and describe three methods to solve the buffer overflow problem under the constraint of acceptable transmission delay. In Chapter 4, we demonstrate how the ideas proposed in Chapter 3 can be used to design coders for speech signals transmitted over a telephone network. Comparisons of these coders and an adaptive non-uniform quantizer with fixed length encoding are also presented. In Chapter 5, conclusions based on the results obtained in Chapter 4 are drawn.

CHAPTER 2 REVIEW OF WAVEFORM CODERS

2.1 Evolution

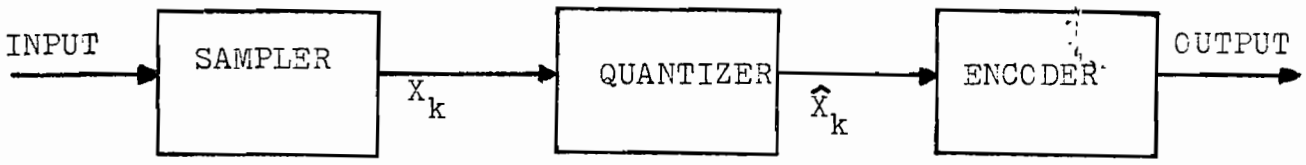
In this study we concentrate on PCM and DPCM systems. Fig 2.1 shows the block diagram of a typical PCM system. The sampler samples the analog input every "T" sampling interval. The quantizer takes the analog sample and replaces it with an approximate value taken from a finite set of allowed values. The encoder takes this approximate value and specifies it with a code-word. The structure of a zero-memory N-level quantizer is shown in Fig 2.2 (a). A N-level zero-memory quantizer can be defined by specifying

(i) a set of N+1 decision levels $[T_o, T_1, ..., T_N]$

(ii) a set of level numbers [1,2,3,...N] and

(iii) a set of output levels $[Y_1, Y_2, ..., Y_N]$.

The step-sizes of the quantizer, being specified by the decision levels, can be uniform or nonuniform according to the application. Max [4] had tabulated the optimum step-sizes, in the sense of giving minimum quantization noise, for inputs with a Gaussian probability density function (pdf). Paez and Glisson [5] have tabulated the optimum step-size for Laplace and Gamma inputs. The operation of a N-level zero-memory quantizer can be viewed as the combination of two operations. First, the "Q[.]" operation, Fig 2.2 (b), which maps the input sample to a level number that indicates into which level it falls. Second the "Q^{-1} [.]" operation, Fig 2.2 (c), which takes the level number and returns the corresponding output level. Whenever the input sample X_k falls in the ith bin, i.e. $T_{i-1} < X_k \leq T_i$, then the combined operation "Q[.] Q^{-1} [.]" will map X_k to its corresponding approximate output $\hat{X}_k$.

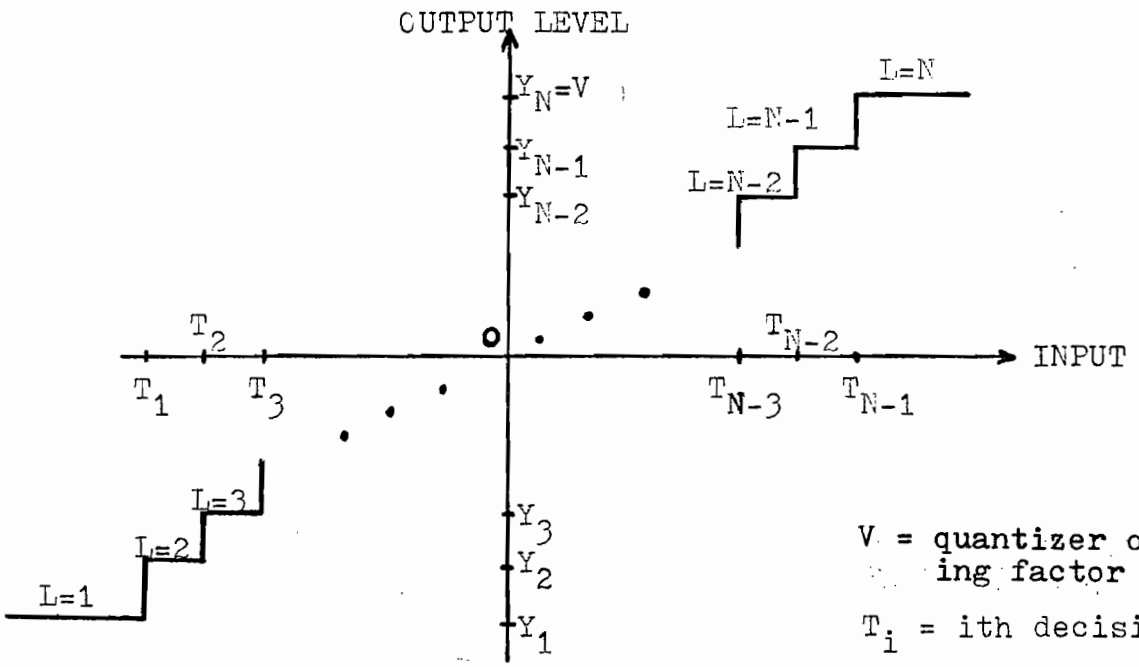


X_k = kth input sample

$\hat{X}_k$ = kth quantizer output
$= X_k + q_k$

q_k = kth quantizer noise

Fig. 2.1 PCM System



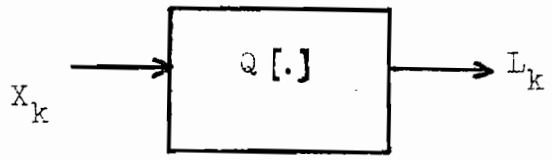
V = quantizer overloading factor

T_i = ith decision level

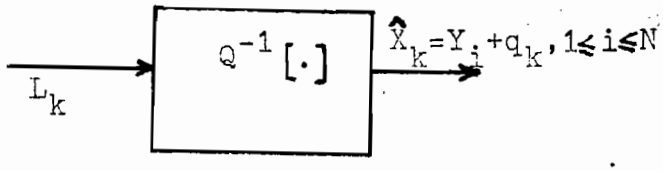
Y_i = ith output level, $1 \leq i \leq N$

L = level number

(a)



(b)

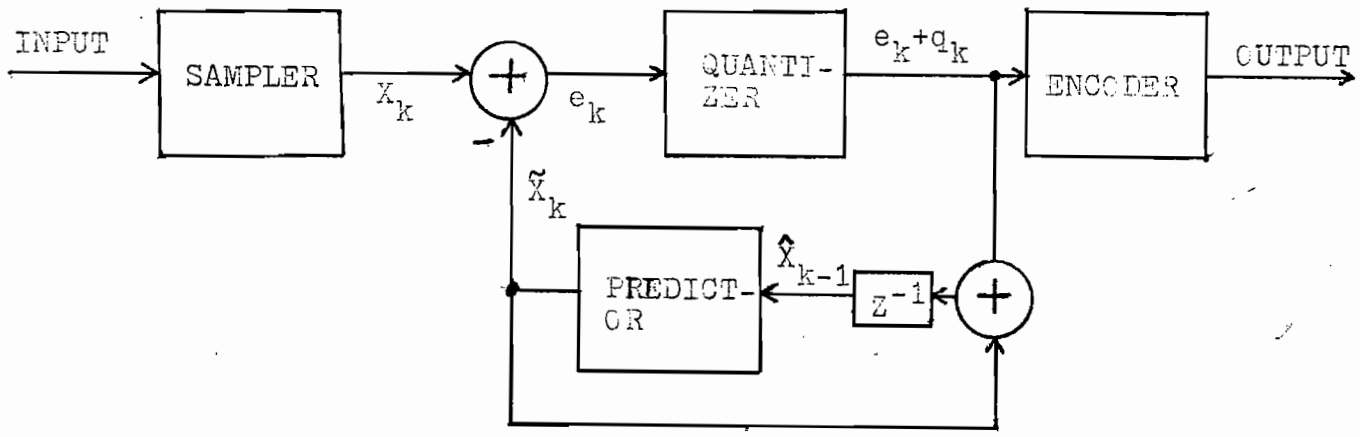


(c)

Fig. 2.2 (a) An N-Level Quantizer
(b) The "Q [.]" Operation
(c) The "Q^{-1} [.]" Operation

For a review of the principles of quantization, see Gersho (6). For nonstationary inputs, adaptive quantization is usually used to deal with the dynamic nature of the inputs.

The block diagram of a typical DPCM system is shown in Fig. 2.3. In DPCM systems, the error signal e_k is quantized rather than the direct input.



q_k = quantization noise

$\hat{X}_k = X_k+q_k$

Fig 2.3 DPCM System

The error signal is formed by substracting a linear predicted value $\tilde{X}_k$ from the input X_k. The linear predicted value is

$$\tilde{X}_k = \alpha_{k1}\hat{X}_{k-1} + \alpha_{k2}\hat{X}_{k-2} + \cdots + \alpha_{kp}\hat{X}_{k-p} \quad , \qquad 2.1$$

where α_{ki} is the predictor coefficient for $1 \leq i \leq p$ and p is the predictor order. The subtraction reduces the correlation present in the input. Both the predictor and the quantizer can be either fixed or adaptive. Adaptive systems can be classified into two categories:

(a) feed-forward adaptive systems and

(b) feed-back adaptive systems

Fig 2.4 shows the feed-forward and feed-back DPCM system with both adaptive predictor and adaptive quantizer. For feed-forward systems, those adaptive parameters, e.g. predictor coefficients and quantization step-size have to be transmitted to the receiver. This kind of information is called side information. For feedback adaptive systems, no side information has to be sent because the receiver can recover this information from the received signals. In Fig 2.4, the quantization step-size for the quantizer and the predictor coefficients α_{ki}'s are transmitted as side information for feed-forward adaptive scheme. If no channel errors have been introduced, then the transmitted information should be equal to the received information and $\hat{X}_k = \hat{X}'_k$.

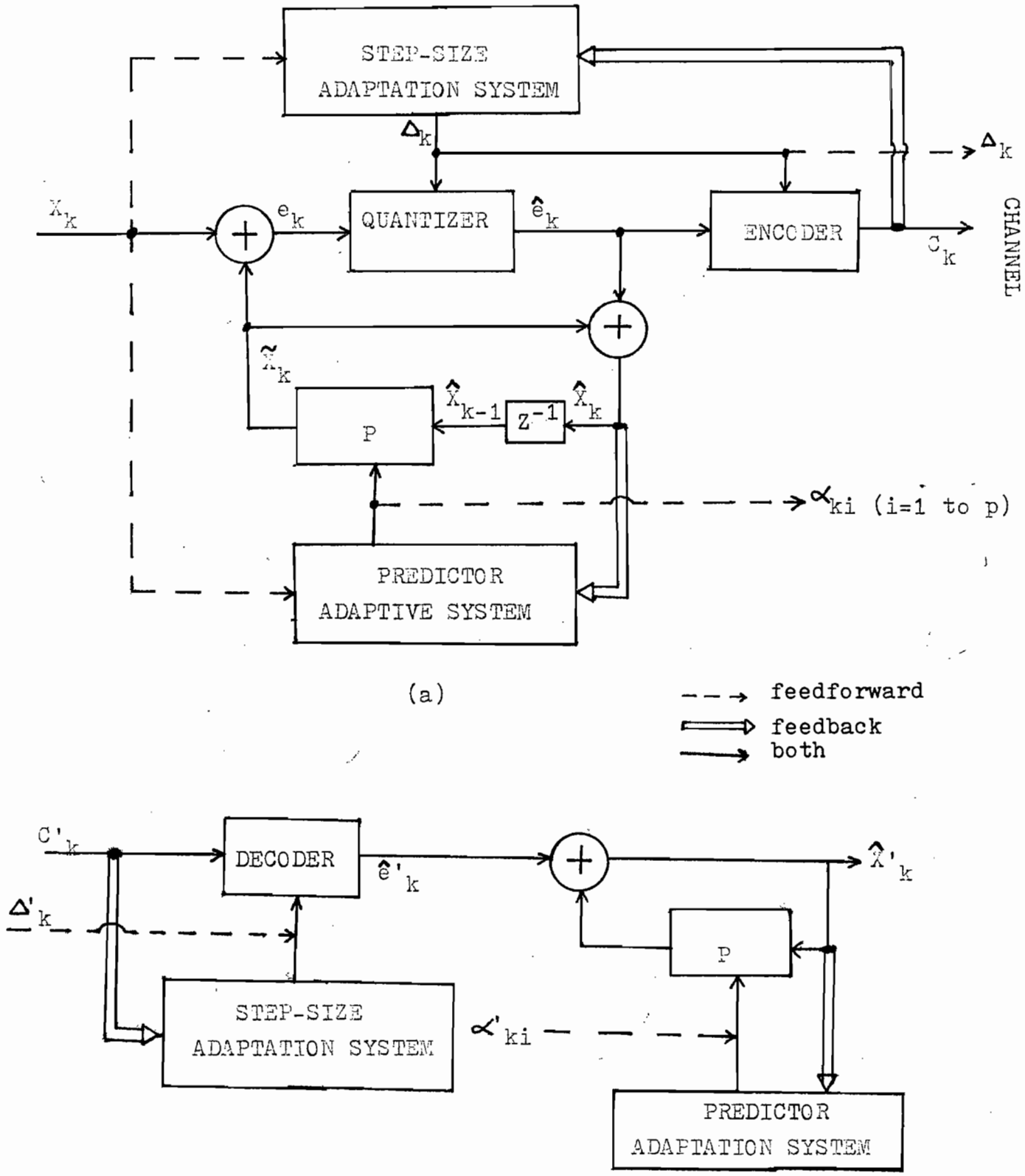


Fig 2.4 DPCM System with both Adaptive Predictor and Adaptive Quantizer (a) Coder & (b) Decoder

The signal to quantization noise ratio (SNR) is defined as

$$SNR = \frac{\text{average energy of the inputs}}{\text{average energy of the quantization noise}} \qquad 2.2$$

From Fig 2.1, the SNR of PCM is

$$SNR = \frac{E_x}{E_q} = \frac{\sum_k x_k^2}{\sum_k q_k^2} \qquad 2.3$$

where E_x, E_q are the input energy and quantization error energy respectively. However, for DPCM systems, the SNR is

$$SNR_{DPCM} = \frac{E_x}{E_q} = \frac{E_x}{E_e} \frac{E_e}{E_q} = G_p \, SNR_Q \qquad 2.4$$

where G_p = predictor pain

$$= \frac{\text{energy of input signal}}{\text{energy of difference signal}}$$

$$= \frac{E_x}{E_q} \qquad 2.5$$

and SNR_Q = quantizer SNR

$$= \frac{\text{energy of difference signal}}{\text{energy of quantization noise}} \qquad 2.6$$

With speech sampled at 8 kHz as input, G_p is about 6 dB for fixed predictor and 7 dB for adaptive predictor for predictor order p equals 1[7]. As p increases, G_p also increases. However, as p increases beyond 9, G_p is more or less reached its maximum. Noll [7] concluded from his work that reasonable maximum G_p for fixed and adaptive prediction are 10.5 dB and 14 dB respectively. Besides larger prediction gain, adaptive prediction scheme is also not very sensitve to speaker and speech material while fixed prediction scheme is inherently very sensitive.

Fig 2.5 shows the probability density functions of real speech, theoretical Gamma and Laplace densities. It is clear that speech can be approximated by either Gamma or Laplace density. Fig 2.6 shows that the probability density function of the difference signal with p=1 and predictor coefficient as determined by Stroh [8] can also be approximated by Gamma or Laplace density if the quantization is "fine". Therefore, the SNR of the quantizer alone will not change much whether differential scheme is used or not. As a result, with differential configuration, the SNR will be 5 dB to 10 dB greater than the SNR of the quantizer with the same quantization scheme and same number of levels which acts directly on the input speech signals. The differential scheme would behave in much the same manner as the direct PCM scheme, i.e. the SNR would increase by 6 dB for each bit added to the code-words, and the SNR would show the same dependence upon signal level. Similarly, the SNR of a μ-law PCM

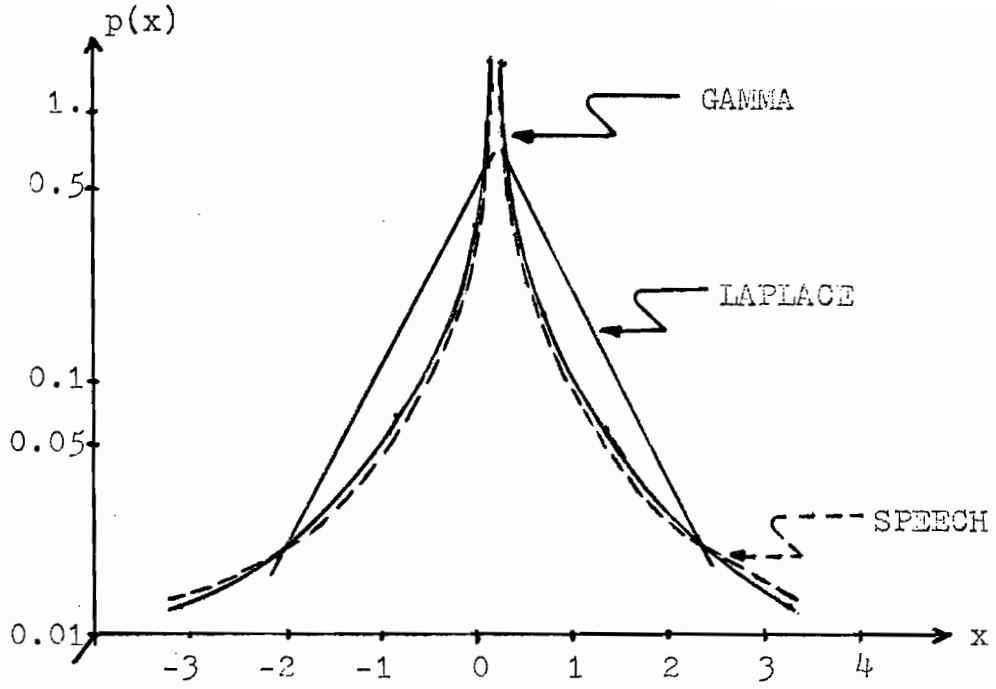


Fig. 2.5 Real Speech and Theoretical Gamma and Laplace Probability Densities (After Paez and Glisson[2])

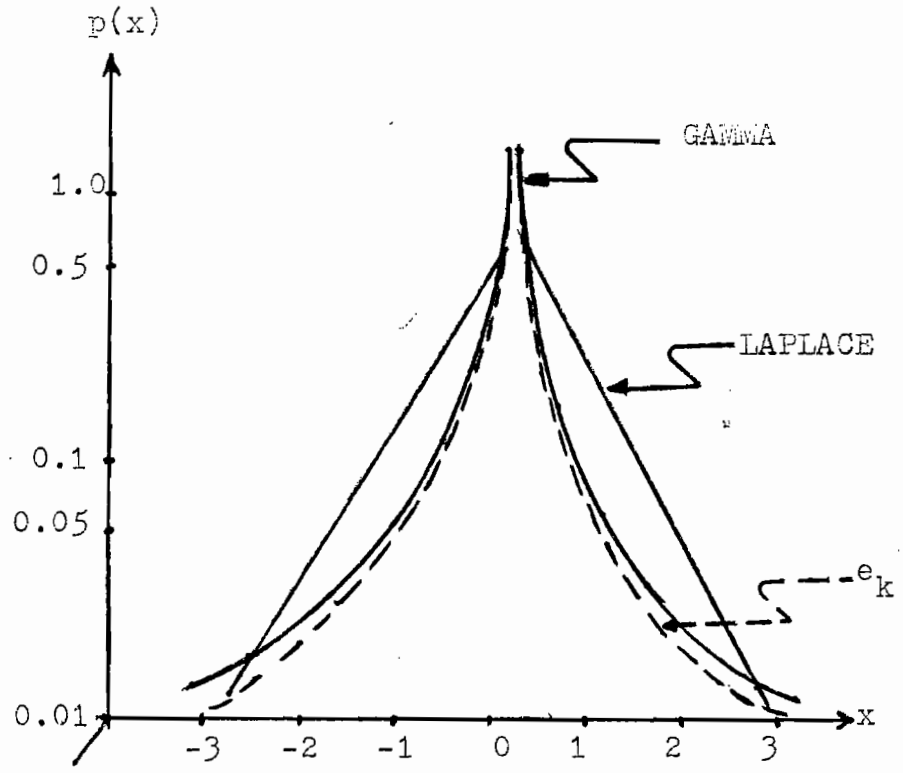


Fig. 2.6 Difference Signal e_k and Theoretical Gamma and Laplace Probability Densities (After Paez and Glisson[2])

quantizer would be improved by about 5 dB to 10 dB if differential configuration is used and at the same time its characteristic insensitivity to input signal level would be maintained.

Speech is a quasi-stationary process. Noll [9] and Jayant [10] [11] had proposed adaptive quantization schemes to track the variance of the input speech. Jayant proposed that a multiplier be associated with each level of the quantizer, see Fig 2.7. The instantaneous variance of the

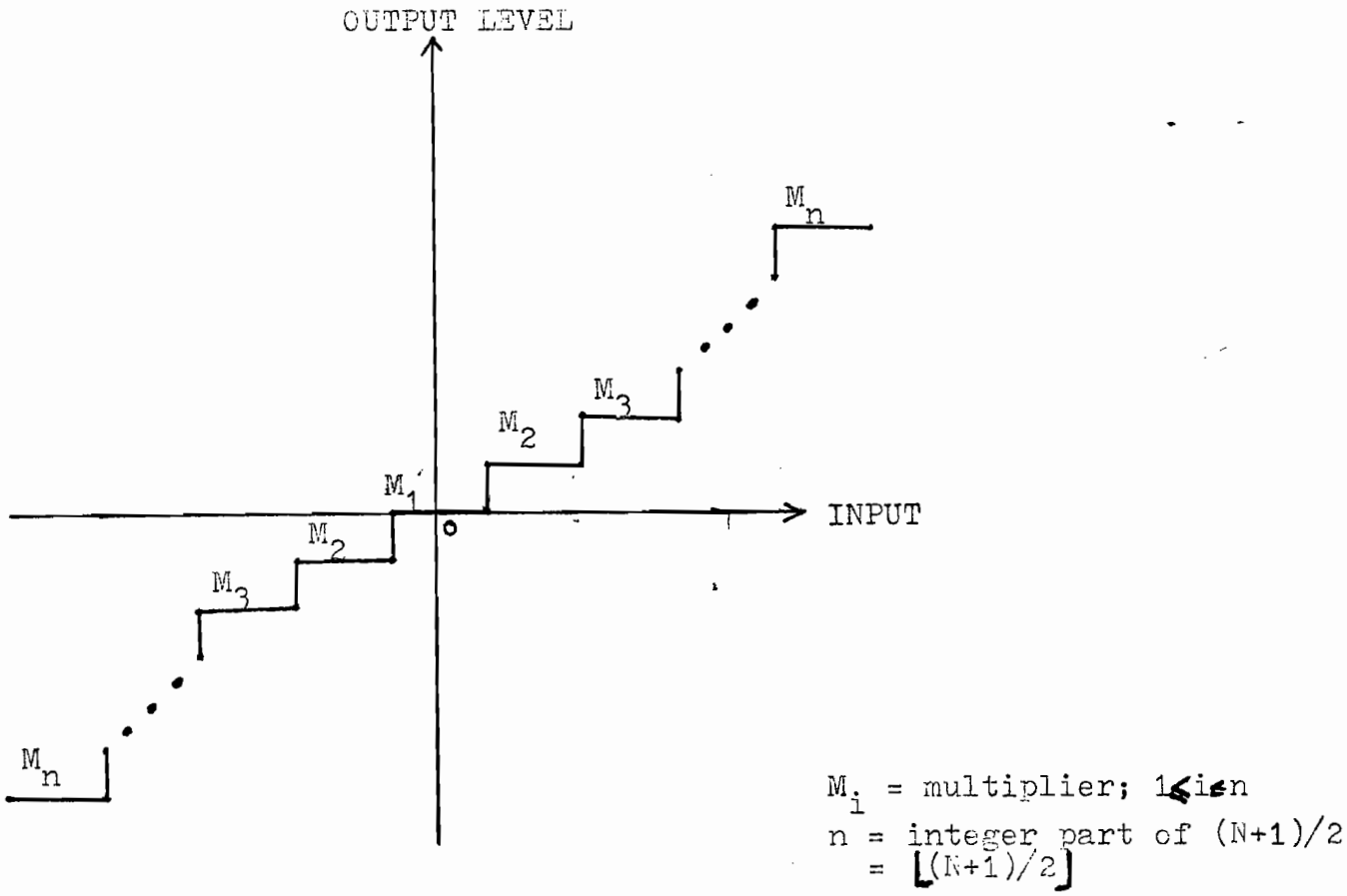


Fig. 2.7 N-Level Quantizer with Adaptation Multipliers

difference signal sample e_k is estimated to be $\tilde{\sigma}_k$ which is equal to

$$\tilde{\sigma}_k = M_i \tilde{\sigma}_{k-1} \quad , \qquad 2.7$$

see Fig 2.8. The difference input is normalized by $\tilde{\sigma}_k$ before being quantized. Note, multiplying the step-size by $\tilde{\sigma}_k$ is equivalent to dividing the input by $\tilde{\sigma}_k$. For a treatment of the theory of adaptive quantizers, see Goodman (12) and Mitra (13). Jayant

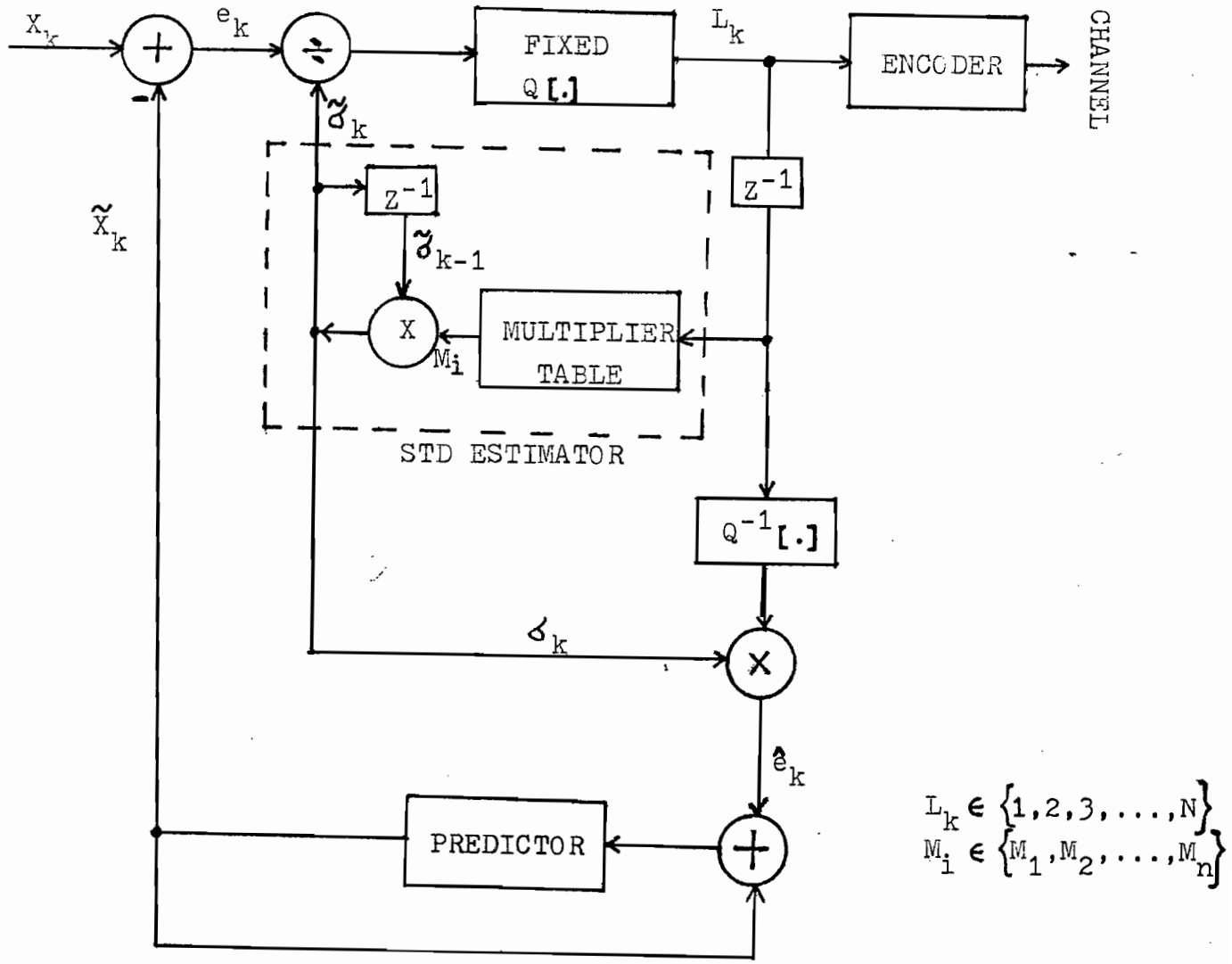


Fig. 2.8 DPCM System with Adaptive Quantizer (DPCM-AQ)

found that the adaptation scheme for speech should be fast increasing and slow decreasing of step-size. Jayant had also found that adaptive quantization outperforms μ-Law PCM with μ= 100 by about 5dB for speech input. Table 2.1 shows the comparison of a μ-Law PCM system and a DPCM system with first order fixed prediction and a feed-back adaptive quantizer designed for a Gaussian pdf quantizer. This table shows that 4-bit DPCM-AQ is preferred subjectively to a 6 bit μ-Law PCM. The SNR improvement for DPCM-AQ with fixed prediction is expected to be 10-12 dB, or roughtly 2-bits. The 4 bit DPCM-AQ was preferred to the 6-bit μ-Law PCM even though the SNR of the 4-bit DPCM-AQ is somewhat lower. This is because the SNR shows an 8 dB bias for PCM coding, i.e the SNR of the PCM has to exceed the SNR of DPCM-AQ by 8 dB before the two are judged to be equally preferable subjectively.

TABLE 2.1 Comparison of Objective and Subjective Performance of DPCM-AQ and Log-PCM with μ=100, $V_x = 8\sigma_x$; where V_x = maximum value the input X_k can assume without overload distortion, σ_x = STD of the input speech.
(After Commiskey, Jayant and Flanagan (11))

Objective Rating (SNR)	Subjective Rating (Preference)
7-bit PCM	7-bit PCM (High Pref)
6-bit PCM	4-bit DPCM-AQ
4-bit DPCM-AQ	6-bit PCM
5-bit PCM	3-bit DPCM-AQ
3-bit DPCM-AQ	5-bit PCM
4-bit PCM	4-bit PCM (Low Pref)

The above discussion shows that for a fixed transmission rate, adaptive differential quantization has a higher SNR than PCM. In other words, adaptive differential quantization allows a reduction in bit rate while keeping the SNR the same. The price paid, of course, is increased complexity in the quantization system. One goal of digital speech signal processing research is to generate coded speech with quality more or less equal to the original band limited, 300-3300 Hz, speech signal at low transmission rates, 16 kb/s or below. Differential quantization tends to remove the correlation in the signal. For speech inputs, Fig 2.6 shows that the pdf of the residual signal e_k is not uniform. This introduces another redundancy. This redundancy can be removed by variable-length encoding. The purpose of using variable-length encoding is to make the entropy of a code-word, the average value of self- information over all posible code-words, as close as possible to the entropy of the source. In our study, this is achieved by assigning longer code-words to less probable symbols and shorter code-words to more probable symbols. In addition, the sample is encoded one by one. When all the symbols to be transmitted are uncorrelated, it is possible to generate codes such that the average code-word length of these codes is almost equal to the entropy of the source symbols. Hence, the name entropy encoding is sometimes used instead of variable-length encoding. H(x) of a discrete memoryless source is defined as:-

$$H(x) \triangleq -\sum_{i \geq 1} p_i \log_2(1/p_i) \quad , \qquad 2.8$$

where p_i is the probability of occurance of symbol Y_i and the summation is sum over all the symbols of the source. The entropy of the output of an N-level quantizer in our study here is

$$H^* = - \sum_{i=1}^{N} p_i \log_2(1/p_i) \quad , \quad 2.9$$

where p_i is the probability of occurance of level i. For a stationary Gaussian source with independent samples, a uniform quantizer with entropy coding achieves an SNR only 1.5 dB less than the very best attainable performance with block quantization. If the source samples are correlated, a higher SNR can always be achieved, e.g. by block quantization. Block quantization is the scheme which first takes a block of samples then de-correlates them through a transformation. Then, the transformed outputs are quantized individually because the variance of the transformed outputs are different from one another. See Berger [14] for a treatment of Rate Distortion Theory.

Hoffman [15] proposed a procedure to generate optimum decodable codes, i.e. codes with minimum redudancy. However, the codes generated in this way are neither self-starting nor self-synchronizing. In this study, other self-synchronizing codes are used. Virupaksha [16], Makhoul [18] and Cohn [19] had studied the performance of DPCM

systems with adaptive quantizer and adaptive predictor plus entropy coding (DPCM-AQ-AP + EC) for speech digitization. Fig 2.9 shows the block diagram of a typical DPCM-AQ + EC system. A buffer is needed before the code-words are transmitted through the channel. This is because the output of the entropy encoder is of variable rate,

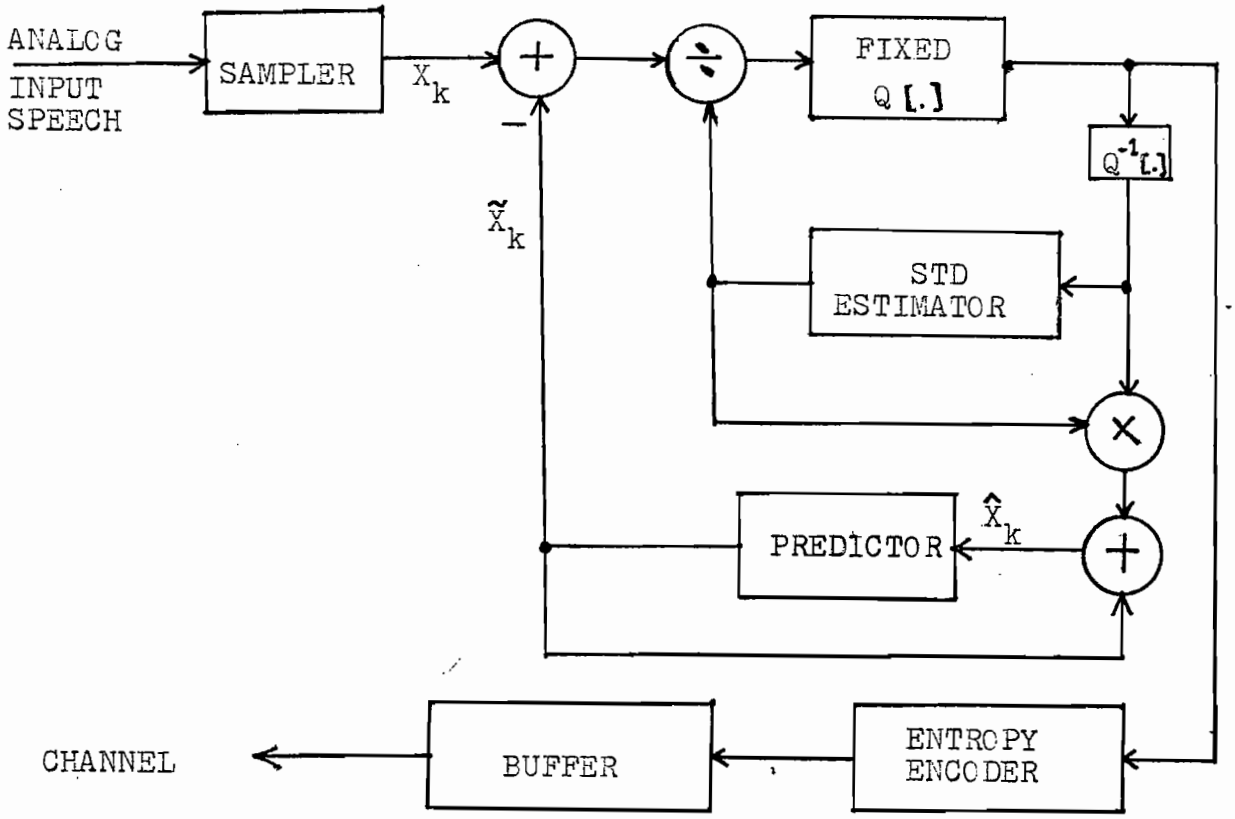


Fig. 2.9 DPCM with Adaptive Quantizer and Entropy Encoding (DPCM-AQ + EC)

but the channel transmission rate is constant. A buffer is also required in the receiver to store up received code-words before any decoding process is being performed. The results of Virupaksha's [16] study indicated that a simple DPCM-AQ + EC system will perform about 2 dB better than one without entropy coding at a channel rate of 16 kb/s. Increasing the channel rate will increase the gain in SNR acheived by entropy coding. Further objective improvement can be achieved by using an adaptive predictor and a better adaptive quantizer [17][19]. Subjective improvement can be accomplished by adaptive shaping of quantization noise spectrum [18].

2.2 Problems in Variable-Length Encoding

The discussion in section 2.1 shows that DPCM systems with variable-length encoding are better than those without. However, the requirement of the buffer between the entropy coder and the channel can lead to

(i) buffer overflow,

(ii) buffer underflow,

(iii) code-word synchronization problems and

(iv) transmission delay.

Buffer overflow and underflow are caused by the quasi-stationary nature of speech signals and the changes of the speaking level from speaker to speaker and from time to time during the conversation. For speech signals, the variance of the voice segments are much larger then unvoiced and silent segments. In telephone transmission, the speech variance is allowed to have a 40 dB dynamic range while maintaining a minimum acceptable SNR of 25 dB (6). When the variance of the input to the quantizer changes, the bit rate into the buffer also changes. This is because it takes time for the quantizer to adapt itself to the input power level. Therefore, the buffer input bit rate can be higher or lower than the channel transmission rate. Buffer overflow will occur if the input bit rate is larger than the transmission rate for a long time. On the other hand, buffer underflow will occur if the buffer input bit rate is smaller than the transmission rate. The buffer underflow problem can be solved by sending dummy signals to the receiver to mark time until the buffer has enough information to transmit.

A set of modified Hoffman codes, the so called self-synchronizing Hoffman codes, is used throughout this project. Fig 2.10 shows the structure of the codes for a 19-level quantizer. The receiver can start parsing the bit stream after it decodes a "0". The self-synchronizing codes for an N-point quantizer, with $N \neq 19$, can be constructed in the same way. Makhoul (18) found that, with channel rate at 16 kb/s, the codes in Fig 2.10 which is optimal for an exponential distribution is almost

optimum for difference speech signals because the inner most seven bins is identical to Hoffman codes and these bins account for 99% of the data.

Figure 2.10 Self-Synchronizing Hoffman codes for a 19-level quantizer.

Level Number	Code-Word Length (bits)	Code-Word
1	18	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
2	16	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
3	14	1 1 1 1 1 1 1 1 1 1 1 1 1 0
4	12	1 1 1 1 1 1 1 1 1 1 1 0
5	10	1 1 1 1 1 1 1 1 1 0
6	8	1 1 1 1 1 1 1 0
7	6	1 1 1 1 1 0
8	4	1 1 1 0
9	2	1 0
10	1	0
11	3	1 1 0
12	5	1 1 1 1 0
13	7	1 1 1 1 1 1 0
14	9	1 1 1 1 1 1 1 1 0
15	11	1 1 1 1 1 1 1 1 1 1 0
16	13	1 1 1 1 1 1 1 1 1 1 1 1 0
17	15	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
18	17	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
19	18	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Therefore, the initial synchronization problem is solved by using this set of codes. In addition, the receiver can synchronous itself when a "0" is decoded. This eliminates the problem of generating a string of incorrect code-words immediately after a wrong bit is received due to channel noise.

Telephone transmission delay of 45 msec or more will cause known echo problems. These problems can be solved by putting echo cancellors or echo suppressors at appropriate locations throughout the network (3). Echo free delay had little effect on speaking level. However, round trip echo free delay of 600 msec is large enough to cause double talking, i.e. simultaneous speech from both speakers, mutual silent and confusion between the speakers (20). Transmission delay can be reduced by using short buffers, but this will increase the probability of buffer overflow and loss of information.

In this project, we will develop buffer management schemes that will eliminate buffer overflow problems for low bit rate speech transmission with acceptable transmission delay. Channel errors are not considered in this study.

Buffer overflow caused by using variable-length encoding had been studied by Goyal [21], Cohn [19] and Qureshi [17]. Goyal shifted the buffer between "normal" and "full" regions. Whenever the buffer is in the full region, i.e. buffer overflow will probably occur, a quantizer with fewer levels is used. The set of codes used with this quantizer has the property that the longest code word is equal to or less than the channel bit rate. The buffer contents, therefore, decreases gradually and returns to the normal region. It is obvious that the SNR is low in the buffer full region. The decision levels of the quantizer in the normal region have to be carefully chosen to ensure that the long-term average bit rate into the buffer is less than the transmission rate by a reasonable amount. This is to ensure the probability that the buffer is in the full mode is small. However, with the quantizer output entropy less than the channel rate most of the time, the system is not running optimally. This is because the maximum SNR a DPCM-AQ + EC system can achieve is directly proportional to the average quantizer output bit rate. Cohn also used a similar technique to deal with the buffer overflow problem [19].

In our study, we will concentrate on the quantizer, the entropy encoder and the buffer of the DPCM-AQ + EC system. Quantization errors can be

classified into two kinds

(i) slope overload (clipping errors) and

(ii) rounding errors (granular noise)

Clipping errors occur whenever the absolute value of the input samples is larger than V in Fig 2.2 (a). Clipping errors cause undesirable degradation in the output speech in the form of "pops" or "clicks". Clipping errors can be reduced by using a quantizer with a large number of levels. Simulation results show that the coders in this study with quantizers of 19-levels have a probability of overload less than 0.003. Gish and Pierce [22] had shown that for quantizers with moderate to large numbers of quantization levels, uniform quantization minimizes the entropy for Laplace input, i.e. the transmission rate is minimized, if variable-length encoding is used, for a given mean-square quantization error. Fig. 2.5 and Fig. 2.6 show that both direct and difference speech samples can be approximated by a Laplace pdf. In this project, a 19-level uniform quantizer with decoded values taken at the center of each bin is used. If the decoded values are taken as the centroid of the quantization bins, the output mean square error will decrease slightly only because of the large number of quantization levels used. Odd number of quantization levels are chosen because this will reduce channel noise during silent segments.

Transmission rate is fixed at 16 kb/s in this project. At this rate, the codes shown in Fig 2.10 are nearly optimal. If the transmission rate changes, other entropy codes have to be derived.

The transmission delay introduced by the presence of a transmission buffer and a recieving buffer in Fig 2.9 is directly proportional to the buffer size.

The total number of bits in the two buffers is not constant. However, the total number of samples in the two buffers is constant. When the transmitter buffer is full, the receiver buffer is empty and vice versa. Therefore, the transmission delay D is given by

$$D = \frac{\text{buffer size}}{\text{channel rate}} = \frac{BS}{Tr} \quad . \qquad 3.1$$

The acceptable delays are different for different applications. For telephone speech transmission, the round trip delay for a satellite link is about 500 msec. Therefore the maximum allowable delay due to the introduction of transmission and receiving buffer is

$$600 \text{ msec} - 500 \text{ msec} = 100 \text{ msec} \quad . \qquad 3.2$$

The maximum tolerable delay in any application in turn limits the buffer size. Increase the buffer beyond this limit may result in an unacceptable delay.

As stated before, speech is a quasi-stationary process. Speech waveforms can be classified into voiced, unvoiced and silent segments. To deal with buffer overflow problem, we can merge unvoiced, silent and other low energy segments together and consider speech waveforms are composed only of high energy and low energy segments.

In the rest of this chapter, three methods to deal with buffer overflow are proposed. The first two methods use feedback while the last is a feed-forward scheme.

3.1 Adaptive Expansion-Contraction Factor

3.1.1 Model

As stated in Chapter 2, changes in input standard deviation (STD) can be tracked by assigning a multiplier to each quantizer level, see Fig 2.7. If all the multipliers are fixed, and if M_1 to M_i are less than 1 and M_{i+1} to M_n are greater than 1, then the multipliers can be thought of as trying to keep the input to the quantizer to a power level σ_c between the quantization levels with multipliers M_i and M_{i+1}. When the input changes state, from high energy to low energy or vice versa, the magnitudes of the multipliers, especially multipliers of inner bins and outer bins determine the adaption speed. Inner bins are quantization levels with code-words that are shorter than or more or less the same as the channel rate in bits per sampling period. In our case, these levels are closer to the most inner bin. Outer bins are quantization levels with code-words that are considerable longer than the channel rate in bits per sampling period. In our case, these level are further away from the most inner bin. The adaption speed in turn determines the buffer size. For speech signals, fast expansion and slow contraction is recommended. In either one of the two states, high energy or low energy, the magnitudes of M_i and M_{i+1} have dominant effect on the instantaneous bit rate into the buffer if variable-length codes are used. For codes shown in Fig. 2.10, as M_{i+1} and M_i increase, the bit rate into the buffer decreases because the input signals have a higher

probability of falling into the inner bins. Conversely, as M_{i+1} and M_i decrease, the bit rate increases.

For fixed multipliers, the response rate is inversely proportional to the steady state granular noise (12). Software simulation techniques are usually used in searching for multipliers which minimize the quantization noise. The set of multipliers do not lead to a unique minimum. Motivated by the exponentially increasing nature of the multipliers, we set

$$M_i = A+B(i-1)^2 \quad ; \quad i=1,2,\ldots,n \qquad 3.3a$$

$$\text{or} \quad M_i = (A+B(i-1)^2)^{0.5} \quad ; \quad i=1,2,\ldots,n \qquad 3.3b$$

where A and B are constants. With multipliers satisfy equation 3.3, the maximum segmental SNR (23) (SEGSNR) obtained is almost optimum. See subsection 4.2 for the definition of SEGSNR. The maximum SEGSNR obtained with multipliers not subject to equation 3.3 is less than 0.1 dB better than that with multipliers satisfy equation 3.3. However, this greatly reduces the computer time in searching optimum multipliers and highly simplify the buffer management scheme which is going to be discussed in the next subsection.

For adaptive multipliers, it should be the rate of change of the multipliers magnitudes which is inversely proportional to the steady state granular noise. All in all, changing the multiplier magnitudes can change the adaptation speed and can adjust the bit rate into the buffer when the speech is in either of the two states. A block diagram of the coder which has control over the bit rate into the buffer is shown in Fig 3.1. The quantizer is fixed. The multiplier of each quantization level is updated with feed-back knowledge from both the quantizer and the buffer.

Mathematically, the output of the multiplier logic unit is

$$MT_k = MT_{k-1}(A_k + B_k(L_{k-1} - \lfloor \frac{N+1}{2} \rfloor)^2) \quad , k \geq 2 \qquad 3.4a$$

$$\text{or} \quad MT_k = MT_{k-1}(A_k + B_k(L_{k-1} - \lfloor \frac{N+1}{2} \rfloor)^2)^{0.5} \quad , k \geq 2 \qquad 3.4b$$

where $\lfloor . \rfloor$ means "integer part of". A_k and B_k are fixed by the buffer management scheme. The determination of A_k and B_k is discussed in the next subsection. MT_k is limited within the range of MT_{min} and MT_{max}. The initial multiplier logic output, MT_1, can assume any value between MT_{min} and MT_{max}. A clipper with V_{max} and $-V_{max}$ as clipping levels is included in the coder. V_{max} is the overloading level of the adaptive quantizer. Subsection 4.2.1 gives an example of how to determine V_{max}. Therefore the inclusion of the clipper will not degrade the coder performance when compared with the DPCM-AQ + EC coder without the clipper because the input samples with amplitude greater than V_{max} are going to be clipped off in the quantizer. However, the clipper is needed in order to deal with buffer overflow problem. This is because it takes time for the coder to track the change in input STD. If the maximum allowable input amplitude is not specified, we cannot determine the minimum buffer size that will not be overflown.

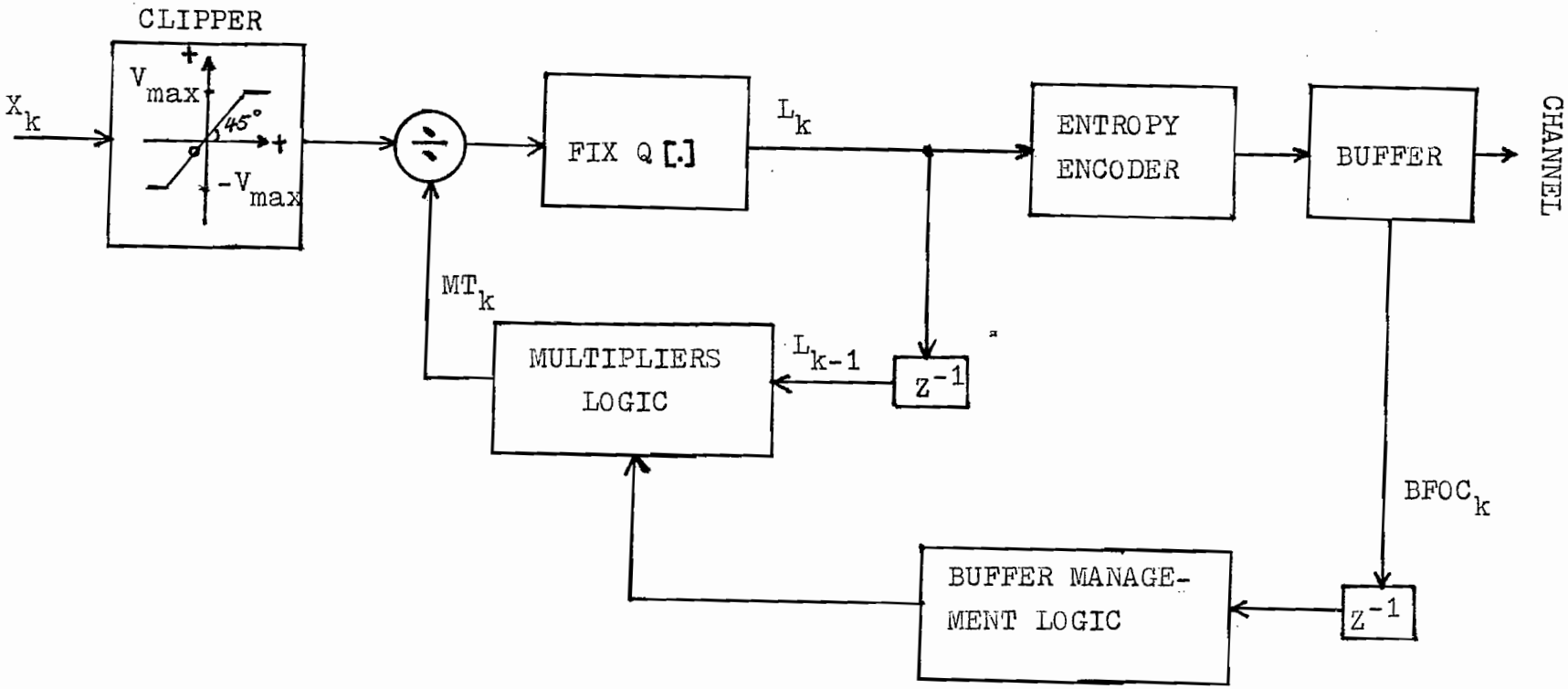


Fig. 3.1 Block Diagram of Coder with Buffer Management

3.1.2 Buffer Management Scheme

In this method, the quantizer is fixed but the multiplier magnitudes are changing through changing "A" and/or "B" in equation 3.3. However, if "A" and "B" can assume any value, they will destroy the primary function of the multipliers, namely to keep the STD of the input to the quantizer to a fixed value σ_c. σ_c will be between levels with multipliers M_i and M_{i+1} if M_1 to M_i are less than 1 and M_{i+1} to M_n are greater than 1. When A or B decreases, σ_c will increase because multipliers M_{i+1}, M_{i+2} and etc. may be less than 1. If the code length corresponding to each multiplier is a monotonic increasing function of multiplier magnitude, which is true in our case, then the bit rate into the buffer will increase when σ_c increase. With MT_k limited to be greater than MT_{min}, σ_c may shift up to a much much higher value in low lenergy states. On the other hand, when A or B increases, σ_c will decrease because multipliers M_i, M_{i-1} and etc. will become greater than 1. Hence, the bit rate into the buffer will decrease. However, any change in σ_c to a fixed quantizer will degrade its performance. Therefore, we have to limit the ranges of change of "A" and "B" so that the change in σ_c will not degrade the coder performance significantly. With channel rate equal to 16 kb/s and sampling rate equal to 6.5 kb/s, the channel removes about 2.5 bits from the transmission buffer every sampling period. For coders using the variable-length codes in Fig 2.10, in order to gain control over the bit rate into the buffer and to eliminate buffer overflow, the minimum range of change of σ_c due to changes in "A" and/or "B" is the step-size of the quantizer itself. To be more specific, the lower limit of the power level σ_c is around $\Delta_x/2$ and the upper limit is around $3\,\Delta_x/2$.

In most application, Δ_x is more or less the minimum range of change of σ_c. The changes in σ_c is the price we have to pay in order to deal with the buffer overflow problem. However, for quantizers with ranges up to 7 - 8 σ_c, this little changes in σ_c will not degrade the quantizer performance.

To simplify the buffer management scheme, only "B" will be changed and "A" is kept fixed. The reasons are:

(i) it will give a larger range of variation for $M_2, M_3 \ldots, M_n$;

(ii) M_2 and M_3 more effective in controlling the bit rate into the buffer when the speech input is in one of the two states and the coder using codes in Fig 2.10, is running at 16 kb/s or above;

(iii) it will limit the maximum range of change of σ_c due to changes in "B" close the limit as stated in the previous paragraph if B is allowed to change with certain prefixed limits "B_{max}" and "B_{min}".

"B_{max}" and "B_{min}" depend on the code-word structure, the channel rate and the sampling rate. The rule to determine B_{max} is to make sure that the buffer contents will decrease even under the worse case once "B_k" and "MT_k" reaches their upper limits "B_{max}" and "MT_{max}" respectively. The worst case will occur if the input signals remain at the maximum allowable level V_{max} for more than one sampling period. The following shows how "B_{max}" and "B_{min}" are obtained for coders using codes in Fig 2.10 with channel rate equal to 16 kb/s and sampling rate equal to 6.5 kSample/s. For "B_{max}, set

$$A + B_{max}\,(2-1)^2 = 1/A^{0.5} \qquad 3.5a$$

or $$A + B_{max}\,(2-1)^2 = 1/A \qquad 3.5b$$

or $$A + B_{max}\,(2-1)^2 = 1/A^2 \qquad 3.5c$$

.

.

.

etc.

Therefore

$$B_{max} = 1/A^{0.5} - A \qquad 3.6a$$

or $$B_{max} = 1/A - A \qquad 3.6b$$

or $$B_{max} = 1/A^2 - A \qquad 3.6c$$

.

.

.

etc.

With MT_{max}, being fixed by the designer, large enough to make the samples with amplitude equal to V_{max} fall in the most inner level, eq. 3.5a guarantees that, under the worst case, the input samples will fall in the 10th and 11st levels in the ratio of 1 : 2. As the code-words of these two levels are 3 bits and 1 bit long, and the channel removes 2.5 bits from the buffer every sampling period, therefore the buffer contents reduce at a rate of 0.5 bits every 3 sampling periods. Eq. 3.5b guarantees that the input samples will fall in the 10th and 11th levels in the ratio of 1 : 1 under the worst case. Therefore the buffer contents reduce at a rate of 1 bit every 2 sampling periods. Similarly, eq. 3.5c guarantees the input samples to fall in the 10th and 11st bins in the ratio of 2 : 1 under the worst condition. As a result, the buffer contents reduces at a rate of 2.5 bits every 3 sampling periods. In this study , eq. 3.6c is adopted because it brings the buffer contents back to the "0" reference point faster than eq. 3.6a and eq. 3.6b. The reader should notice that if, instead of adaptive multipliers, fixed multipliers M_is satisfy eq. 3.3 with "B" given by eq. 3.6 are used, there will be no buffer overflow. However, experimental results show that the coder with these multipliers performs worse than an adaptive non-uniform quantizer with fixed length code-words. If another set of fixed

multipliers is used in order to improve the SNR, then buffer overflow may occur. This is because, depending on the speaker and the speech content, the bit rate into the buffer may tend to increase when the speech is in the high energy state. This may fill up the buffer before the transition comes.

The rule to determine "B_{min}" is that when "B_k" reaches "B_{min}" and "MT_k" reaches "MT_{min}", the buffer contents should increase if "MT_{min}" is small enough to map the input samples to levels with code-words longer than the channel rate in bit per sampling period. Again, channel rate is equal to 16 kb/s and sampling rate is equal to 6.5 kSample/s . Set

$$A + B_{min}\,(3-1)^2 = 1 \qquad 3.7$$

$$B_{min} = (1 - A)\,/4 \qquad 3.8$$

As the code lengths corresponding to M_3 is larger than 2.5 bits, see Fig. 2.10, therefore the buffer contents will increase.

There is more than one way to change "B" in order to deal with the buffer overflow problem. A very simple way is adopted here, see Fig 3.2. Only two pieces of information are required to implement this. First, is the buffer occupancy BFOC, number of bits in the buffer, increasing or decreasing? Second, is the buffer occupancy greater than or less than the "0" reference point? The "0" reference point can be set at anywhere within the buffer. In this study, it is set at the middle of the buffer.

If the buffer occupancy is increasing, i.e. the k-1st code-word length is greater than the channel rate in bits per sample, then

(i) BFOC $\geq$ 0

$$B_k = B_{k-1} + \Delta B_1 \quad ; \quad B_{min} \leq B_k \leq B_{max} \;\&\; k \geq 2 \qquad 3.9$$

(ii) BFOC $<$ 0

$$B_k = B_{k-1} + \Delta B_2 \quad ; \quad B_{min} \leq B_k \leq B_{max} \;\&\; k \geq 2 \qquad 3.10$$

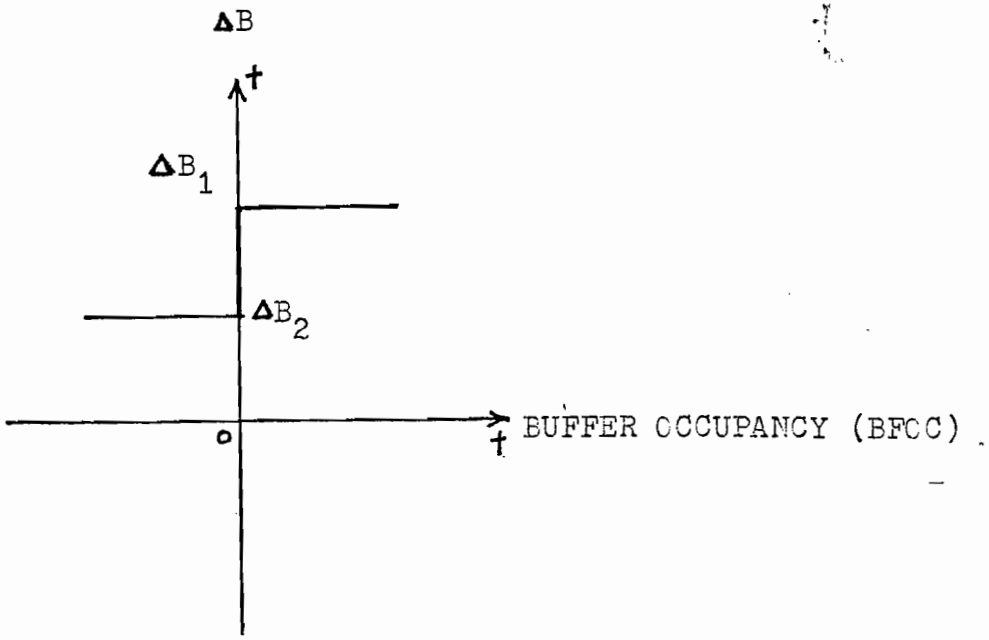
For decreasing buffer contents, i.e. the k-1st code-word length is less than the channel rate in bits per sample, then

(iii) BFOC $\geq$ 0

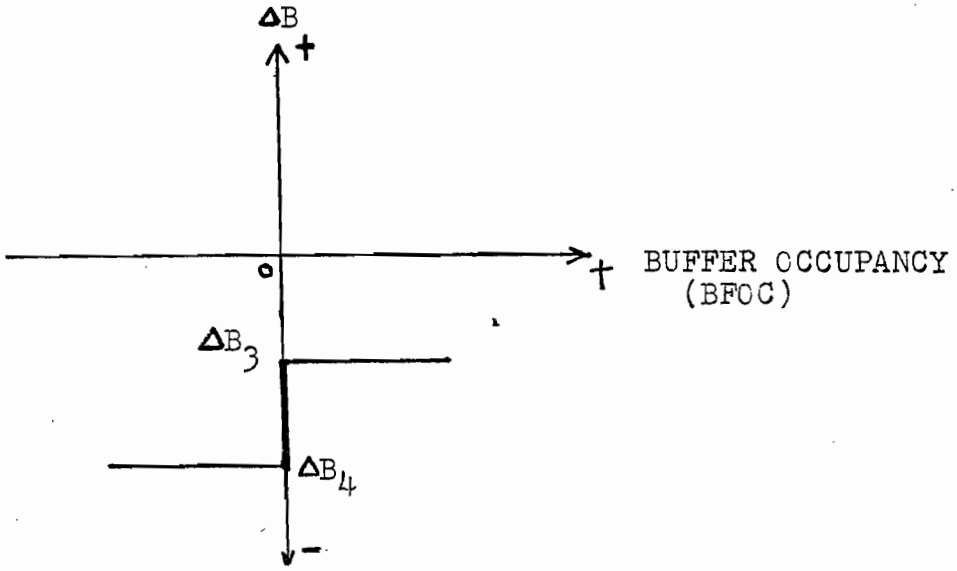
$$B_k = B_{k-1} + \Delta B_3 \quad ; \quad B_{min} \leq B_k \leq B_{max} \;\&\; k \geq 2 \qquad 3.11$$

(iv) BFOC $<$ 0

$$B_k = B_{k-1} + \Delta B_4 \quad ; \quad B_{min} \leq B_k \leq B_{max} \;\&\; k \geq 2 \qquad 3.12$$



(a)



(b)

Figure 3.2 (a) ΔB verse BFOC for Increasing Buffer Contents
(b) ΔB verse BFOC for Decreasing Buffer Contents

B_{max} and B_{min} are obtained from equations 3.6 and 3.8 respectively. The intial value of B, B_1, can assume any value between B_{max} and B_{min}.

We end this subsection by listing the steps which should be followed to design a coder which can eliminate buffer overflow by the method proposed above.

Steps

(i) Fix MT_{max} and MT_{min} such that
(a) they will provide the required dynamic range for speech signals, and
(b) MT_{max} is large enough to bring the maximum allowable input V_{max} down to a magnitude that will fall into a bin with code-word length less than the number of bits taken out by the channel per sampling period.

(ii) Optimize A, ΔB_1, ΔB_2, ΔB_3 and ΔB_4 for speech signals.

(iii) Test the coder with a square wave under the worst condition and note the maximum buffer occupancy. This maximum buffer occupancy in turn is used to calculate the required buffer size; an example is given in subsection 4.1.

(iv) Calculate the maximum possible delay D_{max}.

(v) If D_{max} obtained from (iv) is too large, increase ΔB_1 and return to step (iii) until D_{max} is acceptable.

3.2 Adaptive quantization step-size

3.2.1 Model

In this subsection, we describe a method in which the buffer management scheme has very little influence on the STD estimation of the input signals. The block diagram is shown in Fig 3.3. As we want to separate STD estimation from buffer overflow strategy, we cannot use L_k to derive $\tilde{\sigma}_k$, the estimated kth sample STD, which is equivalent to MT_k in section 3.1. This is because L_k does not reflect X_k's amplitude for a non-fixed quantizer. Therefore, the output of $Q^{-1}[\cdot]$, Y_is, must be used for STD estimation. Stroh[8] suggested a block average approximation. The estimated signal variance is

$$\tilde{\sigma}_k^2 = \frac{1}{M} \sum_{m=0}^{M-1} (\hat{X}_{k-1-m})^2 \qquad 3.13$$

where M = block length

$$\hat{X}_{k-1} = \tilde{\sigma}_{k-1} Y_{L_{k-1}} \qquad 3.14$$

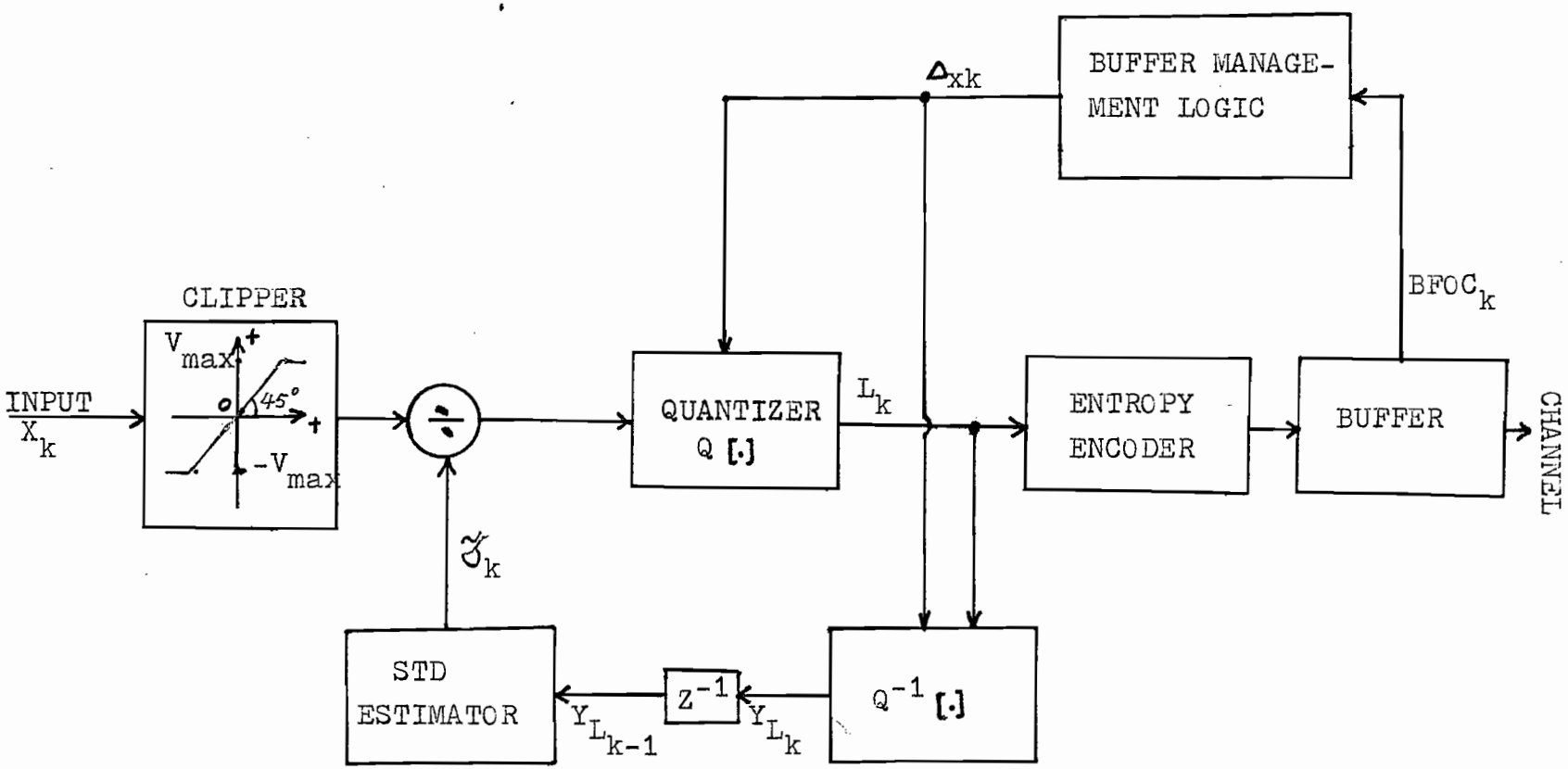


Fig. 3.3 Block Diagram of Coder with Buffer Management Scheme Using Adaptive Quantization Step-size

Castellino[24] suggests an exponential average

$$\tilde{\sigma}_k^2 = \frac{1-\gamma}{\gamma} \sum_{m=1}^{k-1} \gamma^m (\hat{X}_{k-m})^2 + \gamma^{k-1} \tilde{\sigma}_1^2 \quad , \qquad 3.15$$

$$0 < \gamma < 1 \quad \& \quad k \geq 2 \; .$$

This can be rewritten as

$$\tilde{\sigma}_k^2 = (1-\gamma)(\hat{X}_{k-1})^2 + \gamma\left(\frac{1-\gamma}{\gamma} \sum_{m=1}^{k-2} \gamma^m (\hat{X}_{k-m-1})^2 + \gamma^{k-2} \tilde{\sigma}_1^2\right)$$

$$= (1-\gamma)(\hat{X}_{k-1})^2 + \gamma \tilde{\sigma}_{k-1}^2 \qquad 3.16$$

Substitute 3.14 into 3.16, therefore

$$\tilde{\sigma}_k = \alpha_k \tilde{\sigma}_{k-1} \quad ; \quad k \geq 2 \qquad 3.17$$

where

$$\alpha_k = ((1-\gamma)(Y_{L_{k-1}})^2 + \gamma)^{0.5} \; . \qquad 3.18$$

For the detailed relationship between an adaptive quantizer and a STD estimator, see Cohn[25]. As a matter of fact, equation 3.3(b) is analog to equation 3.18. The effect on the STD estimator due to changes in quantization step-size is to degrade the accuracy of the estimated STD when the step-size is large. Equation 3.18 shows that

$$\alpha_k \leq 1 \qquad \text{if} \quad Y_{L_{k-1}} \leq 1 \tag{3.19}$$

and

$$\alpha_k > 1 \qquad \text{if} \quad Y_{L_{k-1}} > 1 \tag{3.20}$$

independent of Y. Therefore, for an odd-level quantizer with $\Delta_{x\max}$ = 2, the STD estimator which performs 3.17 and 3.18 will always track the input STD and adapt the input to the quantizer to unit standard deviation. Whenever the input speech signals changes from low energy to high energy state, the bit rate into the buffer increases because it

takes time for the STD estimator to track the change and so the input signals have a higher probability of falling in the outer bins during this transition. Buffer overflow may or may not occur. This depends on whether (i) the coder can adjust the bit rate into the buffer when the input is in one of the two states,

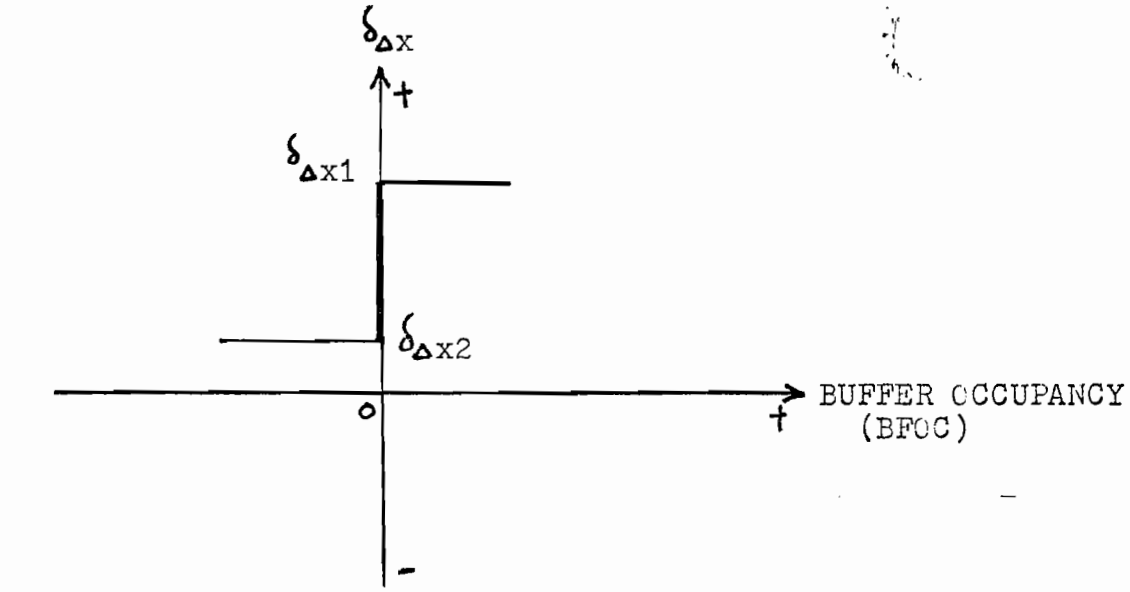
(ii) the STD estimator can adapt fast enough to the transition from low energy state to high energy state.

3.2.2 Buffer Management Scheme

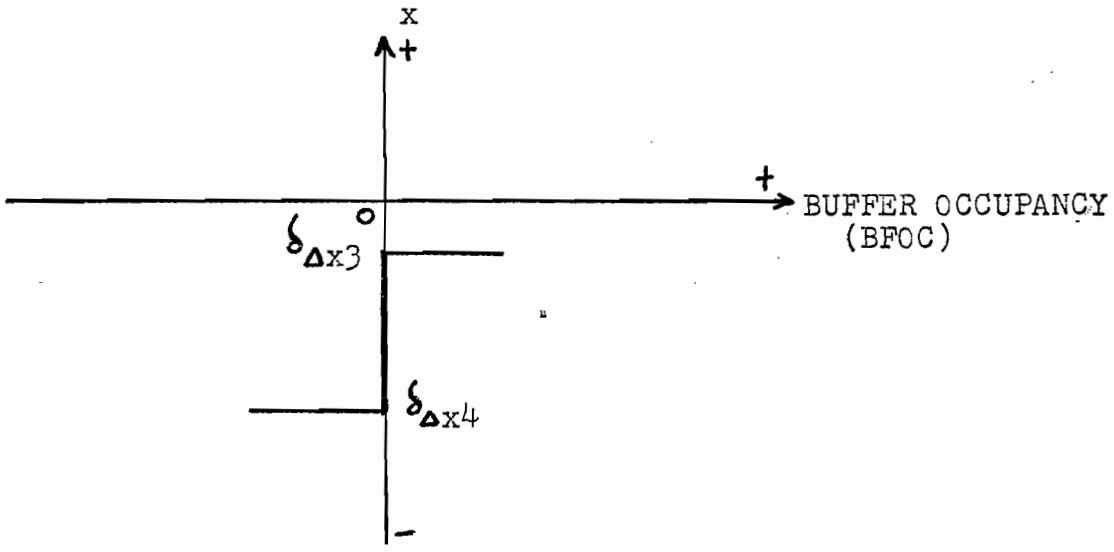
The width of the bins (quantization step-size) has an influence on which bin an input sample is going to fall. When the quantization step-size is large, the sample has a higher probability of falling in the inner bins. When the step-size is small, the sample has a higher probability of falling in the outer bins. Therefore, we enlarge the step-size when the buffer occupency increases and decrease the step-size when the buffer occupency decreases. A simple way of changing the step-size, similar to that of changing "B" in subsection 3.1.2, is shown in Fig 3.4. When the length of the k-1st code-word is greater than the number of bits out of the buffer by the channel per sampling period, then

(i) BFOC $\geq$ 0

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta x} \quad ; \quad \Delta_{xmin} \leq \Delta_x \leq \Delta_{xmax} \quad \& \; k \geq 2 \qquad 3.21$$



(a)



(b)

Fig. 3.4 (a) $\delta_{\Delta x}$ verse BFOC for Increasing Buffer Contents
(b) $\delta_{\Delta x}$ verse BFOC for Decreasing Buffer Contents

(ii) BFOC $\prec$ 0

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta x2} \quad ; \quad \Delta_{xmin} \leq \Delta_x \leq \Delta_{xmax} \quad \& \quad k \geq 2 \qquad 3.22$$

When the length of the k-1th code word is less than the channel rate in bits per sample, then

(iii) BFOC $\geq$ 0

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta x3} \quad ; \quad \Delta_{xmin} \leq \Delta_x \leq \Delta_{xmax} \quad \& \quad k \geq 2 \qquad 3.23$$

(iv) BFOC $\prec$ 0

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta x4} \quad ; \quad \Delta_{xmin} \leq \Delta_x \leq \Delta_{xmax} \quad \& \quad k \geq 2 \qquad 3.24$$

The initial bin width, Δ_{x1}, can be of any value between Δ_{xmax} and Δ_{xmin} The value of Δ_{xmax} is:

$$\Delta_{xmax} = \begin{cases} 2 \text{ for odd-level quantizer} \\ 1 \text{ for even level quantizer} \end{cases} \qquad 3.25$$

This limit on the quantization step-size eliminates the possibility that the buffer management will destroy the function of the STD estimator. The value of Δ_{xmin} depends on the channel rate, the code-word structure and the sampling rate. The rule to determine Δ_{xmin} is that when "Δ_{xk}" reaches "Δ_{xmin}" and "γ_k" reaches "γ_{min}", the buffer contents should increase if "γ_{min}" is small enough to map the input samples to levels with code-words longer than the channel rate in bits per sampling period.

The values chosen for the parameters $\delta_{\Delta xi}$ have strong effect on SNR and delay. $\delta_{\Delta x1}$ and $\delta_{\Delta x2}$ will speed up the standard deviation adaptation in the transient period of transition from low energy to high energy state and reduce the probability of occurance of outer bins. Therefore the minimum buffer size that will not result in buffer overflow is directly proportional to the sizes of $\delta_{\Delta x1}$ and $\delta_{\Delta x2}$. However, if $\delta_{\Delta x1}$ is too large, SNR will decrease because of large fluctuation in the quantizer's decoded outputs when the input is in one of the two states. As $\delta_{\Delta x1}$ becomes larger and larger, the reduction in delay will be less significant and finally the step-size will increase faster than $\tilde{\sigma}$. Obviously, SNR will degrade a lot at this stage. $\delta_{\Delta x3}$ and $\delta_{\Delta x4}$ will reduce the bin width if the output bit rate of the entropy encoder is less than the channel rate. $\delta_{\Delta x4}$ will also improve the SNR of the silent state. All in all, the STD estimator tracks the changes in input speech signals while the buffer management strategy maintains the buffer contents to the "0" reference position of the buffer. The following lists the steps to determine values for parameters of the proposed coder.

Steps

(i) Select $\tilde{\sigma}_{min}$ and $\tilde{\sigma}_{max}$, the minimum and maximum of the STD estimator output, such that this will provide the required

dynamic range for speech signals. In addition, $\tilde{\sigma}_{max}$ should be large enough to bring the maximum allowable input Vmax to less than unity.

(ii) Determine Δ_{xmin}.

(iii) Optimize $\delta_{\Delta x1}$, $\delta_{\Delta x2}$, $\delta_{\Delta x3}$ and $\delta_{\Delta x4}$ with a test speech utterance.

(iv) Test the coder with a square wave under the worst condition and note the maximum buffer occupancy.

(v) If the maximum possible delay obtained in (iv) is not acceptable, increase $\delta_{\Delta x1}$ until the delay is acceptable.

3.3 Block Quantization

3.3.1 Model

The concept of block quantizaion is used to solve the buffer overflow problem. Fig 2.5 shows the block diagram of the coding process. The samples are first stored up in a buffer. Every M sampling periods, a

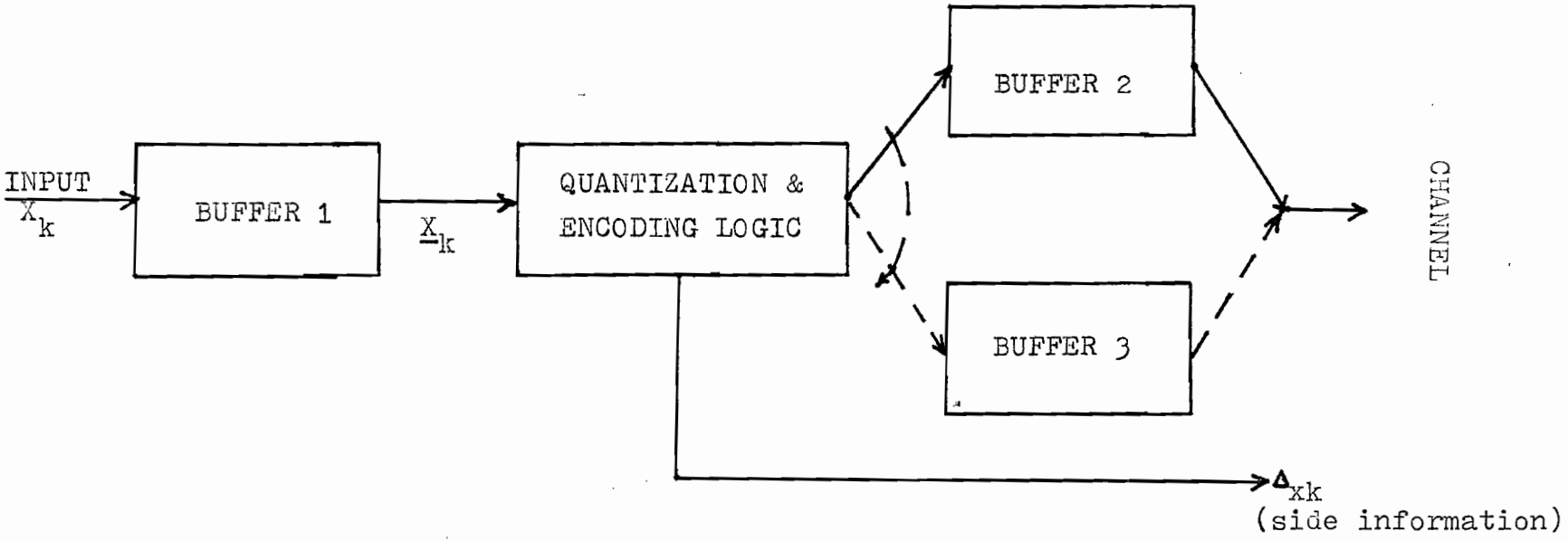
vector of samples of dimension M is fed to the quantization and encoding logic unit. Unlike conventional block quantization, the vector does not undergo any transformation for the purpose of decorrelation. All samples of the vector are quantized with quantizers of equal step-size and equal number of quantization levels. Each quantizer output is encoded with codes shown in Fig 2.10. Then the total number of bits B_N required to code the whole vector is checked. If

$$\left.\begin{array}{l} B_T \text{ (threshold bit)} \quad B_N \quad M \times \frac{Tr}{Sr} - B_s = B_{max} \text{ ,} \\ \text{where Tr is the channel rate in kb/s and} \\ \text{Sr is the sampling rate in kSample/s} \text{ ,} \end{array}\right\} \quad 3.26$$

then the code-words are dumped into the idle buffer which is not transmitting. B_T in equation 3.29 is fixed by the designer. One reasonable choice for codes in Fig 2.10 is

$$B_T = B_{max} - (N-1)/2 \quad . \qquad 3.27$$

B_s is the number of bits used to code the step-size. If B_N does not satisfy 3.26, then the quantization and encoding logic unit will adjust the step-size, quantize and encode the whole vector again. This process will be repeated until B_N satisfies 3.26. The strategy for adjusting B_N will be discussed in the next subsection.



$\underline{X}_k$ = kth input vector of dimension M

Δ_{xk} = quantization step-size for the kth input vector

Fig. 3.5 Block Diagram of Coder with Buffer Management Scheme Using the Block Quantization Concept

In a practical realization, another constraint has to be met. The time required for the quantization and encoding logic unit to process the vector and dump the code-words into the idle buffer (either buffer 2 or buffer 3) has to be less than or equal to (M/Sr) sec.

3.3.2 Buffer Management Scheme

Once again, the idea of changing the step-size is used to change the probability of occurence of each bin of the quantizer. For any given input vector, the probability of occurence of the inner bins will increase as step-size increases and will decrease as step-size decreases. B_N, therefore, will decrease as step-size increases and will increase as step-size decreases if entropy codes shown in Fig 2.10 are used. We can conclude that B_N is a monotonic decreasing function of quantization step-size, Fig 3.6 shows this function with an assumption that when the quantization step-size equals to 0, all elements of the vector fall into the two outer most bins. As the step-size increases indefinitely, the whole vector will finally fall into the most inner bin.

For stationary input and constant transmission rate, the quantization distortion decreases as the dimension of the vector M increases. In the limit, as M increases to infinite, the minimum distortion approaches a limiting value depending on the transmission rate[13]. However, this is not true

for non-stationary processes like speech in our application here. If M is large, the vector may include both high energy and low energy segments. The distortion in the silent periods will be large. This is because each element of the vector is being quantized by quantizers with equal step size and equal number of quantization levels subjected to equation 3.26. This may result in a step size which is fairly large for the silent period. The other factor which limits the size of the vector is the transmission delay D which is approximately equal 2M/Sr sec. Therfore, we have to find an optimum M such that it will give best SEGSNR and will introduce a delay which is less than the maximum allowable delay.

There are 3 conditions which govern the strategy for adjusting the step-size. They are

(i) B_N must be less than B_{max},

(ii) The final step-size should give best SNR for the corresponding vector and

(iii) The final step-size must be determined within M/Sr sec.

Obviously, conditions (i) and (iii) have to be met at any cost. A fast algorithm to determine the optimum step-size has to be developed. Now let us take a close look at condition (ii). Overload distortion increases with decreasing step-size. The relationship between the step-size and granular distortion is not as simple. For a given sample X_k, the granular distortion is not a monotonic function of the step-size. The error $|\hat{X}_k - X_k|$ lies in the shaded area of Fig 3.7. That is, reducing the step-size does not guarantee the reduction in granular noise. However, if the step-size is adjusted according to equation 3.28,

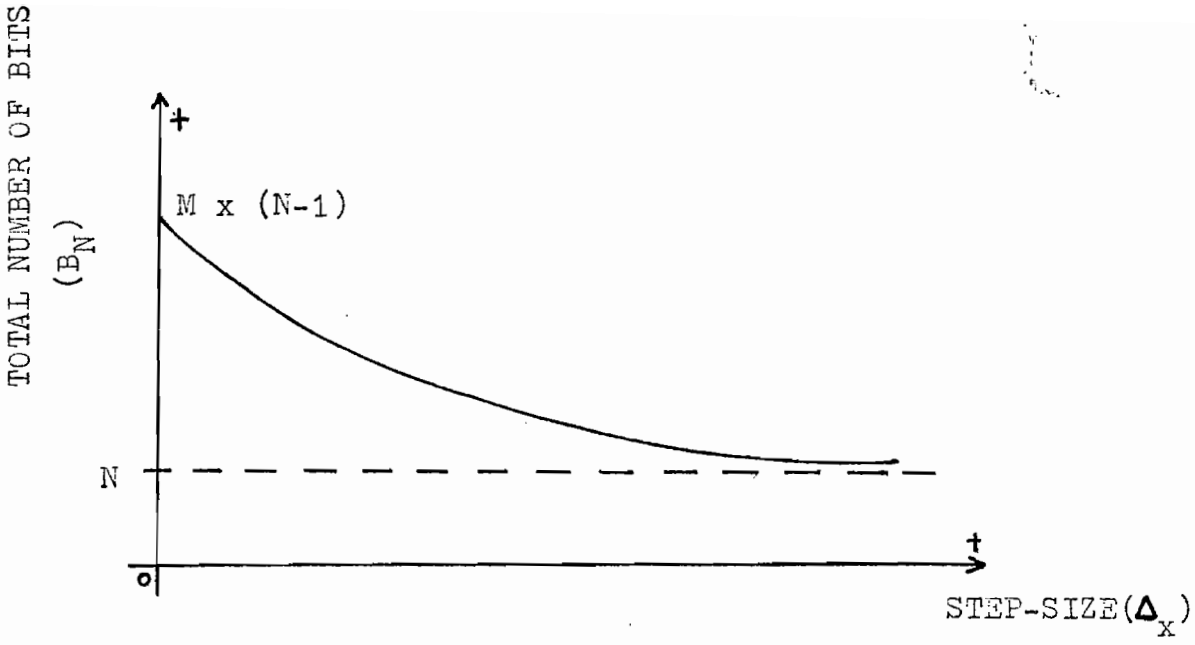


Fig. 3.6 B_N verse Δ_x

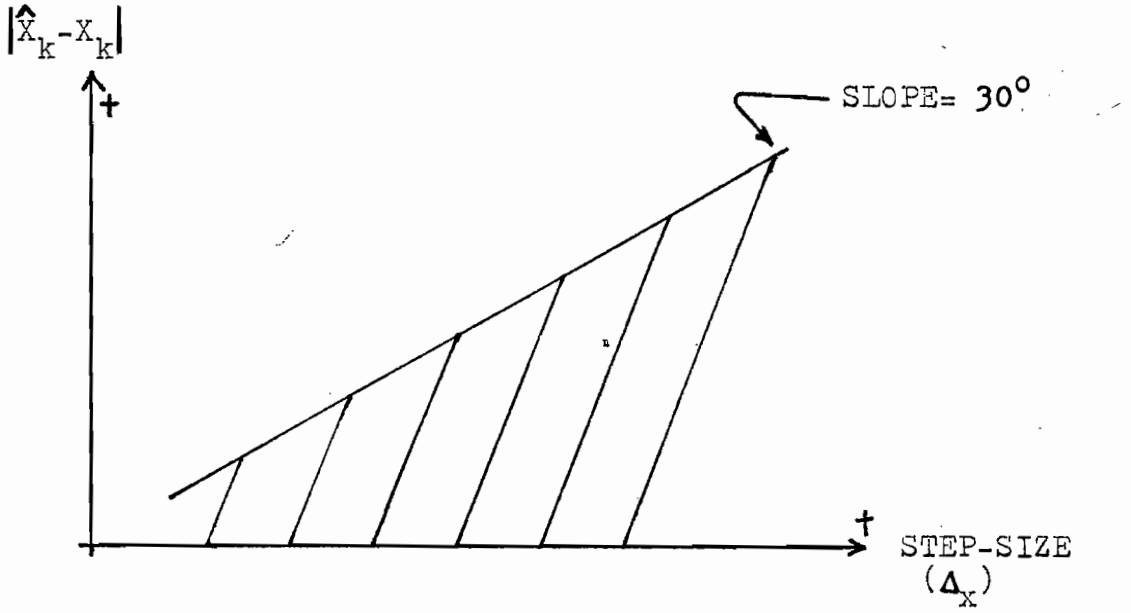


Fig. 3.7 Quantization Step-Size verse Granular Distortion

then granular noise will not change in the opposite way the step-size is changing. This is also true for a finite dimension vector

$$\Delta_{xk} = 2^{i} \Delta_{xk-1} \quad ; \quad i \in \{\text{Integer}\} \qquad 3.28$$

$$k=2,3,4,\ldots \quad \& \quad \Delta_{x1} = \text{initial guess}$$

of samples. For M tends to infinite and with continuous input, we reduce to the well known result that granular noise is approximately $\Delta_x^2/12$ for uniform quantizer. Adjust the step-size according to equation 3.28 has the defect that the resolution is not good enough. Let B_1 and B_2, with corresponding step-sizes Δ_{x1} and 2 Δ_{x1}, satisfy 3.29.

$$B_2 < B_{max} < B_1 \qquad 3.29$$

In between Δ_{x1} and 2 Δ_{x1}, Fig 3.7 shows that there will exist other step-sizes which may give substantial improvement in SNR over 2 Δ_{x1}. Statistically, with M reasonable large, a smaller step-size will give smaller granular noise. Therefore, we adopt the following strategy for changing the step-size.

(i) Sort out Δ_{x1} and $2\Delta_{x1}$ with corresponding B_1 and B_2 satisfy equation 3.29

(ii) Let $\Delta_{xL} = 2 \times \Delta_{x1}$, $\Delta_{xs} = \Delta_{x1}$

(iii) Let $\Delta_{xk} = \Delta_{xL} - (B_{max} - B_2)(\Delta_{x1} - \Delta_{xs})/(B_1 - B_2)$

and calculate the corresponding B_N.

(iv) Let $\Delta_{xs} = \Delta_{xk}$ and $B_1 = B_N$ if $B_N > B_{max}$

Let $\Delta_{xL} = \Delta_{xk}$ and $B_2 = B_N$ if $B_N < B_T$

(v) Repeat (iii) and (iv) until $B_T < B_N < B_{max}$. Then Δ_{xk} is the step size for the corresponding input vector.

A detail flow-chart for the quantization and encoding process is shown in Fig 4.12. The above method of adjusting the step-size without regard of SNR favourite granular noise at the expenses of overload noise. If the input signals changes abruptly, the vector may accidently contain only samples of the two extremes, i.e., samples with amplitude very close to either "0" or "V_{max}". If this happens and the number of samples with

amplitude close to V_{max} is smaller than ((Tr/Sr)-1) x M/(N-2), assuming codes in Fig 2.10 are used, then the above strategy to sort out the step-size will end up with a step-size which will result in a very large overload distortion. Fortunately the energy rises gradually before the pitch pulses come and decays gradually from voice to silent period, see Fig 4.1. Another way to eliminate the possibility of large overload distortion is to use as many quantization levels as possible to totally eliminate clipping error. Besides the above proposed method to sort out the right step-size, there are other ways to do this too. For example, one can calculate the STD of the vector and adjust the step-size according to the calculated STD until equation 3.26 is met.

We end this section by adding a few more comments on this method to solve buffer overflow. In order to satisfy the speed requirement, the number of iterations to determine the step-size has to be limited. In hardware implementation, quantized step-size should be used in the transmitter as well as the receiver, and, the maximum allowable step-size should be large enough to make input samples with amplitude equal to V_{max} or $-V_{max}$ fall in quantization levels with code-words shorter than the channel rate in bits per sampling period.

CHAPTER 4 MODELS SIMULATION AND RESULTS

In this chapter, we describe the computer simulation for the buffer management schemes discussed in chapter 3, experimental set up, input signal characteristics and the comparisons of the proposed coders with an adaptive non-uniform quantizer with fixed length codes.

4.1 Experimental Set up

Computer simulation is used to investigate the performance of the discussed buffer management schemes. Software implementation has some inherent advantages over hardware implementation. Software implementation permits

(i) experimental optimization and

(ii) easy modification of system parameters.

The steps for the simulation procedure adopted in this study are listed below:

(i) The analog signal is sampled at 6.5 kHz, digitized by a 15 bit A/D converter and band-pass filtered by an FIR (finite impulse response)

digital filter with a pass-band equal to 200 Hz to 3.2 kHz.

(ii) Carry out any necessary pre-processing operations. In this study, linear prediction technique [1] is used to remove the format energy and pitch prediction technique [1] is used to remove the pitches.

(iii) The software than simulates the adaptive quantizer, entropy encoder and buffer management in non-real time with input generated in (ii).

The outputs produced above can be converted back to analog signals by a 15 bit D/A coverter and playback under computer control. The simulation discussed above is performed on a VAX-11 mini-computer in FORTRAN. Appendix A gives a listing of the FORTRAN programs and subroutines relevant to the simulation.

The utterances used in this study are:

(i) A: "Sorry the number you have dialed is not in service." and

(ii) B: "It's easy to tell the depth of a well."

The speech utterances used as input for the simulation of the coders are:

(i) SRFMT; utterance A with formant structure removed.

(ii) SRFMP; utterance A with both formant structure and pitch removed.

(iii) FA3; utterance B spoken by a female speaker.

(iv) MB3; utterance B spoken by a male speaker.

The average STD of these four utterances are tabulated in Table 4.1. The waveform of SRFMT is shown in Fig 4.1. Speech utterance SRFMT is used to obtain optimum parameters for the quantizer and the buffer management logic unit.

TABLE 4.1 Average STD

SRFMT	SRFMP	FA3	MB3
315.53	258.14	2816.4	1575.7

4.2 Simulation results

The simulation results are presented in four subsections. The first deals with adaptive quantizer (AQ) + entropy encoder (EC) + buffer management

(BM) using adaptive expansion-contraction factors (AECF). The second deals with AQ + EC + BM using adaptive quantization step-size (AQSS). The third one deals with AQ + EC + BM using the concept of block quantization. For all three coders, the number of quantization levels, N, is set equal to 19 and the entropy codes in Fig 2.10 are used. The fourth one presents the comparisons among the above three simulations and an adaptive non-uniform quantizer + fixed length encoder (ANQ + FLC).

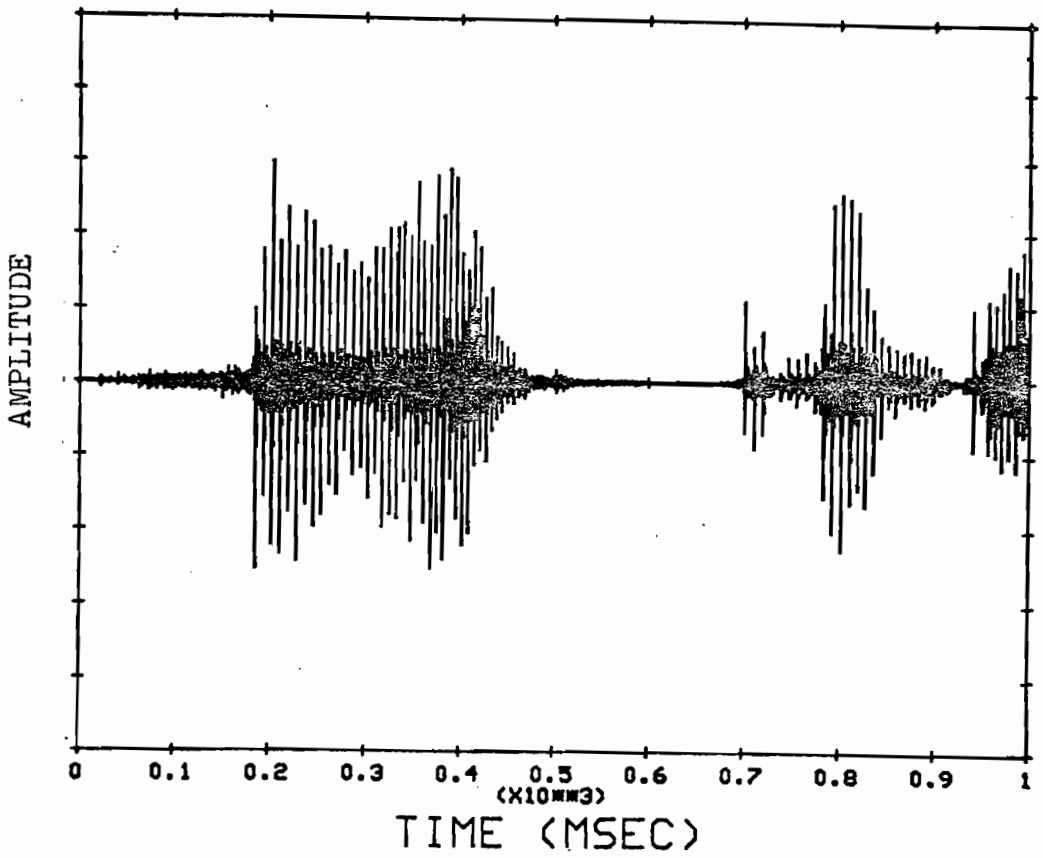
The performance of the three proposed coders are examined by the following two methods.

(i) Plots of input waveforms to the coders, decoded quantizer output waveforms at the receiver under the assumption of error-free transmission, waveforms of buffer occupancy and waveforms of sliding SNR[15]. The SNR for sliding SNR waveforms is computed by equation 4.1 over 100-sample (15.4 msec) segments of the decoded quantizer output.

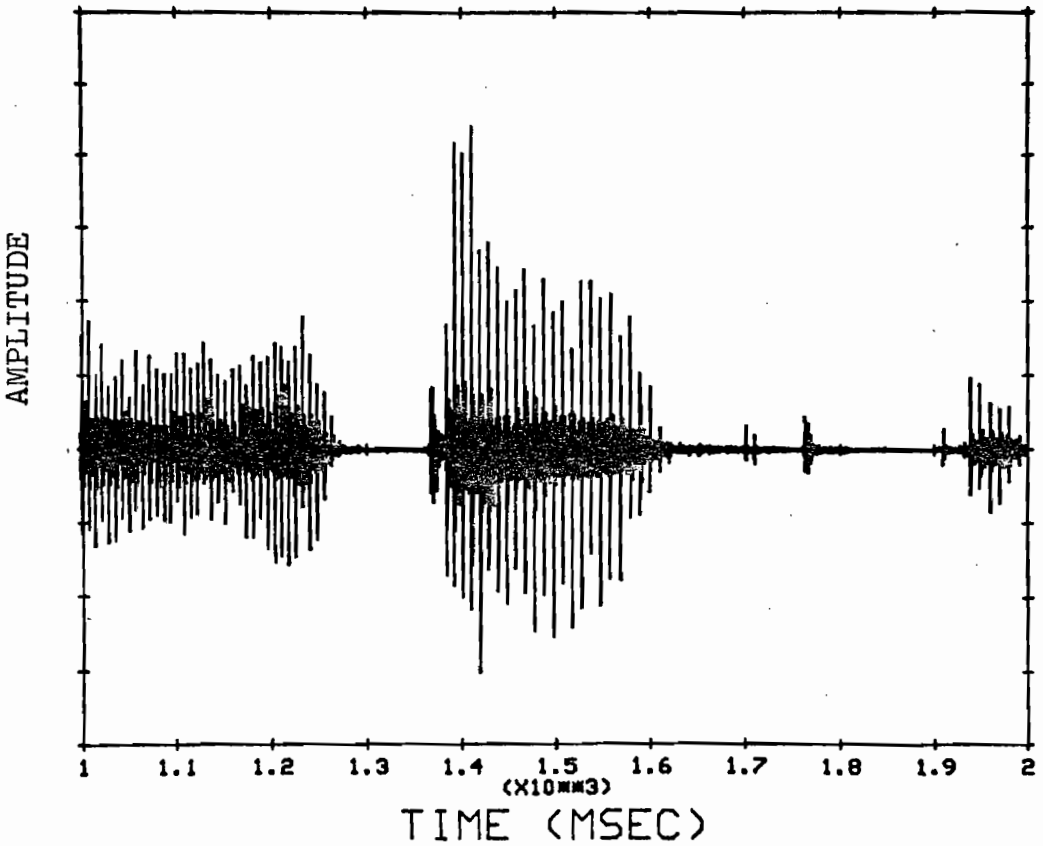
$$\text{SNR (Sliding)} = 10\text{Log}_{10}\left(1 + \frac{\sum_{k=1}^{100} (X_k)^2}{\sum_{k=1}^{100} (X_k - \hat{X}_k)^2}\right) \quad 4.1$$

$$= \text{SLISNR}$$

The "1" in equation 4.1 restricts the SNR of individual segment to positive values.


AMPLITUDE
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)


AMPLITUDE
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

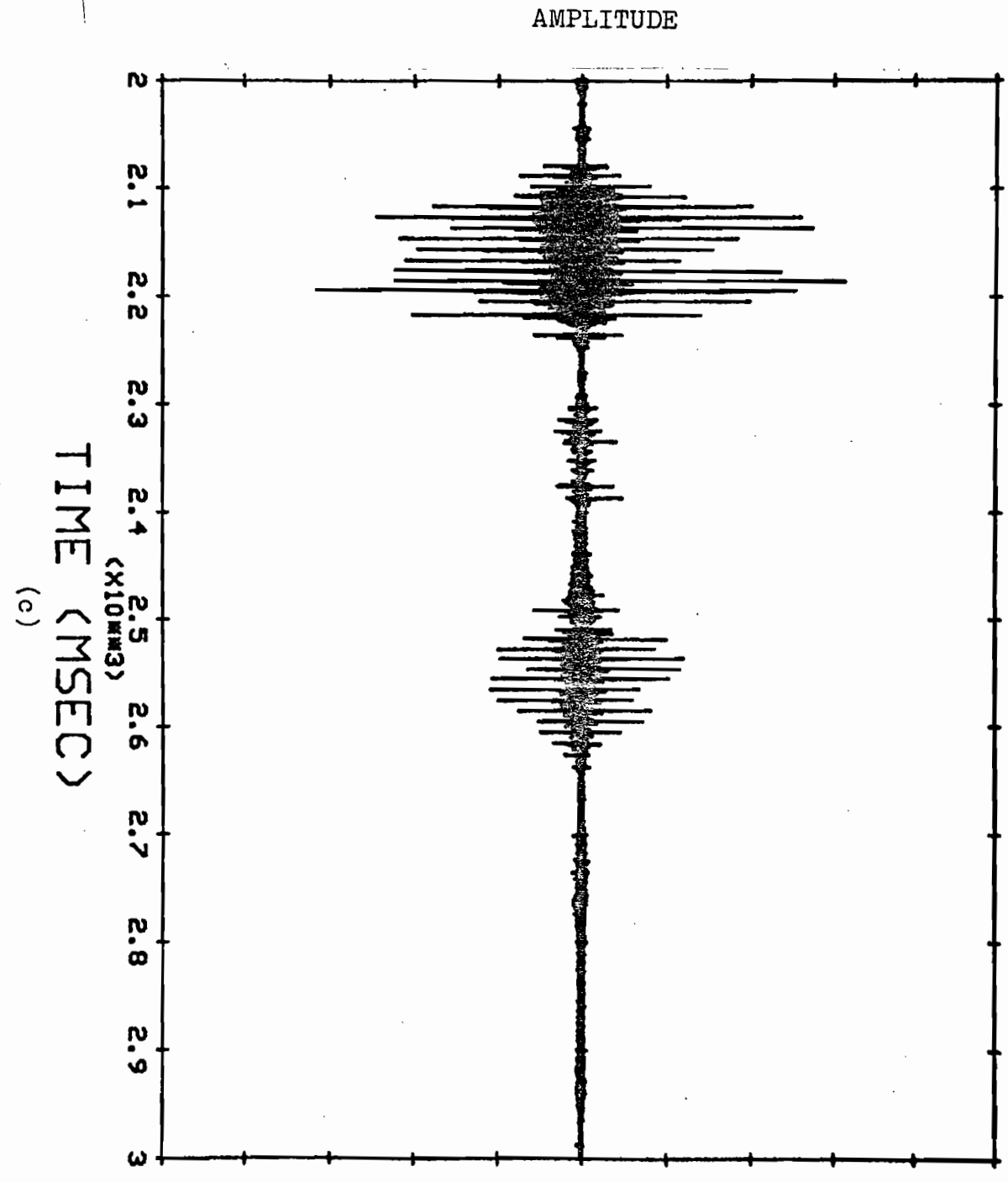


Fig. 4.1 (a),(b) & (c) Waveform of SRFMT

(ii) Segmental SNR (SEGSNR) as an objective distortion measurement.

Conventional SNR tends to give more weight to high amplitude segments of speech. Segmental SNR eliminates this bias by averaging SNR over short intervals (10 - 30 msec). In our simulation, all the intervals are of the same length.

Let:-

N_S = number of samples each interval, and

N_{SEG} = number of intervals of the input speech utterance

Then

$$\text{SNR (convention)} \triangleq 10 \times \text{Log}_{10}\left(\frac{\sum_{k=1}^{N_S \times N_{SEG}} X_k^2}{\sum_{k=1}^{N_S \times N_{SEG}} (X_k - \hat{X}_k)^2}\right) , \quad 4.2$$

but

$$\text{SEGSNR} \triangleq 10 \ \text{Log}_{10} q^2 \quad 4.3$$

with

$$\text{Log}_{10}\ (1+q^2) = \frac{1}{N_{SEG}} \times \sum_{j=1}^{N_{SEG}} \text{Log}_{10}\left(1+\frac{\sum_{k=(j-1)N_S+1}^{N_S} X_k^2}{\sum_{k=(j-1)N_S+1}^{N_S} (X_k - \hat{X}_k)^2}\right)$$

$$= Z \quad , \quad 4.4$$

$$\therefore q^2 = 10^Z - 1 \qquad 4.5$$

$$\therefore \text{SEGSNR} = 10 \times \log_{10} (10^Z - 1) \qquad 4.6$$

Again, the "1" added to q^2 in equation 4.4 restricts the contributions from the individual segments to positive values. This allows one to take into account the effect of noise both in the present and absense of speech. In our simulations, N_S is set equal to 100 samples, i.e. 15.4 msec, for coders proposed in subsection 3.1 and 3.2. N_{SEG} depends on the length of the utterance. For SRFMT and SRFMP, N_{SEG} equals 212 segments. Optimum parameters for the proposed coders are obtained by maximizing segmental SNR with SRFMT as input.

4.2.1 Adaptive Expansion - Contraction Factors

In this subsection, the coder shown in Fig 3.1 with buffer control parameters satisfying the delay requirement for speech transmitted over the telephone network is simulated. For convenience, the equation of MT_k and those related equations are repeated here.

$$\left.\begin{aligned} MT_k &= MT_{k-1}(A'+B'_k(L_{k-1}-\lfloor\tfrac{N+1}{2}\rfloor)^2) \;;\quad k \geqslant 2, \\ MT_{min} &\leqslant MT_k \leqslant MT_{max} \quad \& \; B'_{min} \leqslant B'_k, B'_1 \leqslant B'_{max} \end{aligned}\right\} \qquad 4.7(a)$$

or

$$\left.\begin{aligned} MT_k &= MT_{k-1} \times (A + B_k \times (L_{k-1} - \lfloor \tfrac{N+1}{2} \rfloor)^2)^{0.5} \quad ; \\ & k \geq 2,\ MT_{min} \leq MT_k,\ MT_1 \leq MT_{max} \\ & \&\ B_{min} \leq B_k,\ B_1 \leq B_{max} \quad . \end{aligned}\right\} \quad 4.7(b)$$

$$B'_{min} = (1-A')/4 \quad ; \quad B'_{max} = 1/(A')^2 - A' \qquad 4.8(a)$$

$$B_{min} = (1-A)/4 \quad ; \quad B_{max} = 1/A^2 - A \qquad 4.8(b)$$

and

$$\left.\begin{aligned} B'_k(B_k) &= B'_{k-1}(B_{k-1}) + \Delta B_i \quad ; \\ & i=1,2,3\ \&\ 4 \quad ; \quad k \geq 2 \end{aligned}\right\} \quad . \qquad 4.9$$

To simplify the search for optimum ΔB_is,

set

$$\left.\begin{aligned} \Delta B_2 &= \Delta B_1/2 \\ \Delta B_3 &= \Delta B_1/2 \\ \Delta B_4 &= \Delta B_1/2 \end{aligned}\right\} \qquad 4.10$$

Now, follow the steps listed in subsection 3.1.2. We first have to fix MT_{min} and MT_{max}. The minimum dynamic range for telephone transmission is 40 dB (6). As 7-bit and 8-bit μ-law PCM with μ=255 are used is telephone networks, therefore Fig. 4.2 is used to determine MT_{max} and MT_{min}. In this simulation, a dynamic range of 45 dB which is larager than the minimum 40 dB is used. With reference to Fig. 4.2, for minimum requirement of 25 dB,

$$20 \log_{10} V/P_{max} = 12 \text{ dB} \qquad 4.11$$

where V = overload factor of the μ-Law PCM coder

P_{max} = maximum input level to the μ-Law PCM system for minimum performance of 25 dB.

With 45 dB dynamic range,

$$20 \log_{10} (P_{max}/P_{min}) = 45 \text{ dB} \qquad 4.12$$

where P_{min} = minimum input level to the μ-Law PCM system for minimum performance of 25 dB.

Therefore,

$$V = 10^{2.25} \times 10^{0.6} \times P_{min} \qquad 4.13$$

In telephone networks, V is set equal to $8\sigma_x$, where σ_x is the operating signal level of the network. In our study, we set $\sigma_x = 10$. Therefore

$$V = 80 \qquad 4.14$$

From eq. 4.13 and eq. 12

$$P_{min} = 0.113 \quad \text{and} \qquad 4.15$$

$$P_{max} = 20.1 \qquad 4.16$$

With reference to Table 4.1, it is noticed that the signal levels of the difference signals are about 10 times less the direct speech signals. As SRFMT is used to determine optimum parameters for the coder. Therefore MT_{min} and $\widetilde{MT}_{max}$ of the coder are

$$MT_{min} = P_{min}/10 = 0.0113 \qquad 4.17$$

$$\widetilde{MT}_{max} = P_{max}/10 = 2.01 \qquad 4.18$$

Therefore the coder will give a roughly flat response for differential inputs with levels lie in between MT_{min} and $\widetilde{MT}_{max}$. We cannot obtain V_{max} of the coder by just scaling V with a factor of 0.1. This is because the predictor in DPCM coders do not remove the pitches in speech utterances. These pitches can have amplitude 15 to 20 times the power level of the difference signals. As σ_x for the μ-Law PCM is 10, so the corresponding level for our coder is 1. This is obtained by scaling down the σ_x of the μ-Law PCM by 10. Therefore, the maximum allowable amplitude to the coder is

$$V_{max} = 20 \qquad 4.19$$

With N = 19, therefore the quantization step-size is

$$\Delta_x = (2 \times V_{max}/\widetilde{MT}_{max})\ (1/(N-2)) \qquad 4.20$$

$$= (40/2.01)\ (1/17)$$

$$= 1.1706 \qquad 4.21$$

Taking into account the requirement of buffer management in our coder, MT_{max} of the coder is

$$MT_{max} = (2V_{max}/\Delta_x) = 34.2 \qquad 4.22$$

With reference to Table 4.1, we see that all the input speech utterances stored in digital form have to be rescaled to a more appropriate range between MT_{max} and MT_{min} such that the limitations of the coder parameters MT_{max}, MT_{min} and V_{max} will not degrade the coder performance for any of the testing utterances. As there are two types of inputs, direct and difference, therefore we rescale different type with different factors. SRFMT is scaled down to a level of 1 and other difference inputs are rescaled with SRFMT as reference. For direct inputs, the STD of FA3 is saled down to 3 and the rest direct inputs are scaled with respect to FA3. Table 4.2 shows the new overall STDs of the utterances.

Table 4.2 Rescaled STDs of SRFMT, SRFMP, FA3 and MB3 for Coder Simulation

SRFMT	SRFMP	FA3	MB3
1	0.8026	3	1.6785

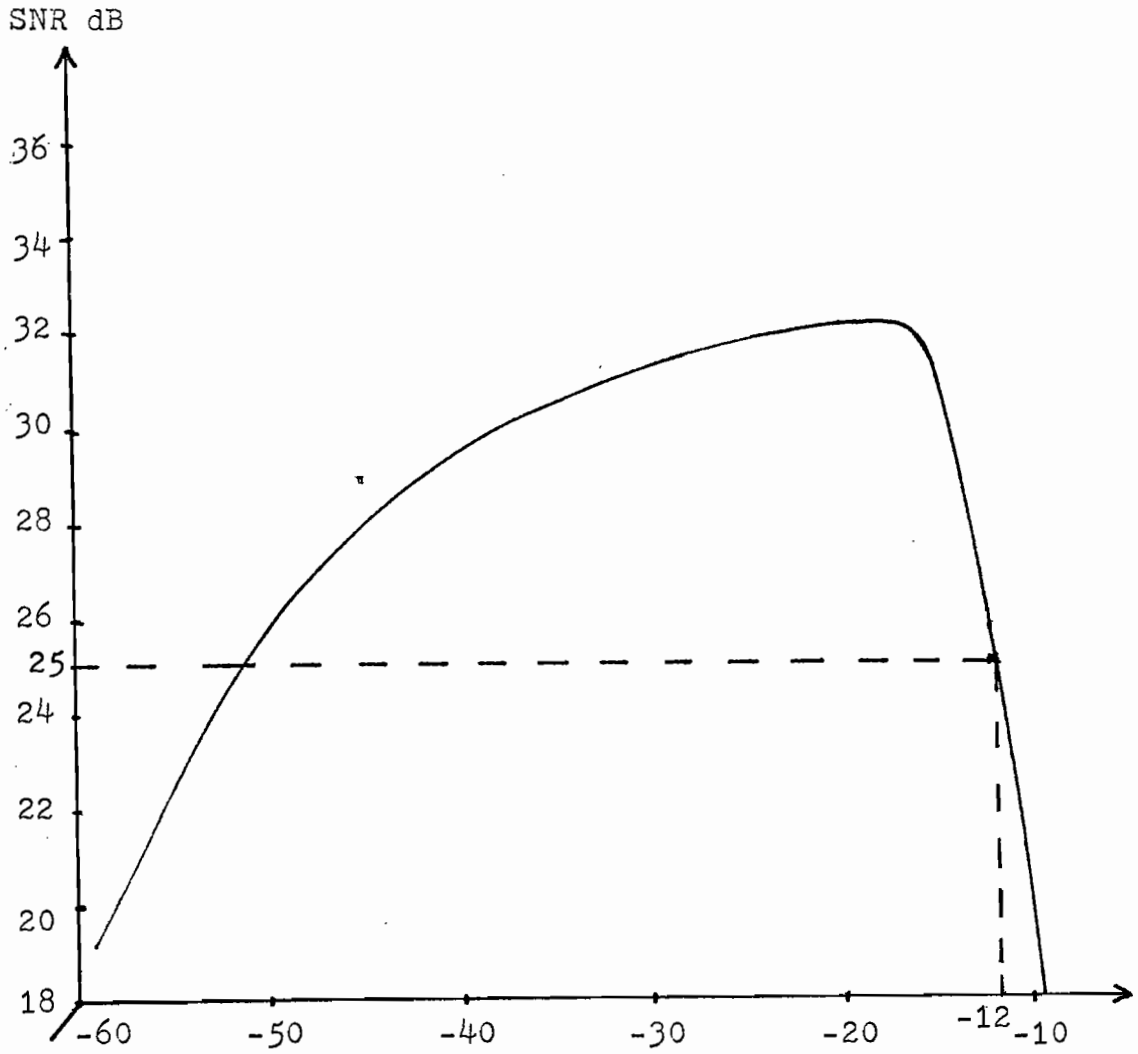


Fig.4.2 SNR verse Input Power Level in dB Relative to Overload Point V, with Laplacian Input pdf for 7-bit μ-Law PCM with μ=255

The second step requires the calculating of optimum, A'(A) and ΔB_1. This can be done in two methods.

Method 1:

(i) Use a pattern search method[26] to maximize the penalty function F in equation 4.23. The penalty function tries to maximize SEGSNR while keeping $\bar{L}$ near R_c (with K = 500).

$$F = SEGSNR - K\left|\bar{L}-R_c\right| \quad , \qquad 4.23$$

with

$$\left.\begin{aligned} &K = 500 \\ &R_c = Tr/Sr = 16/6.5 \text{ b/sample} = 2.46 \text{ b/sample} \\ &\bar{L} = \text{average number of bits/sample} \\ &\quad = \sum_{k=1}^{N_T} B_k/N_T \quad \text{b/sample;} \\ &B_k = \text{code-word length of the kth sample,} \\ &N_T = \text{total number of samples} \\ &\quad = 21248 \text{ for SRFMT} \end{aligned}\right\} \quad 4.24$$

and

$$MT_k = MT_{k-1} \times (A' + B' \times (L_{k-1} - \lfloor \frac{N+1}{2} \rfloor)^2) \; ; \qquad k \geqslant 2, \quad MT_{min} \leqslant MT_k, \; MT_1 \leqslant MT_{max} \qquad \text{4.25a}$$

or

$$MT_k = MT_{k-1} \times (A + B \times (L_{k-1} - \lfloor \frac{N+1}{2} \rfloor)^2)^{0.5}; \qquad k \geqslant 2, \quad MT_{min} \leqslant MT_k, \; MT_1 \leqslant MT_{max} \; . \qquad \text{4.25b}$$

Note, there is no buffer management in the coder because both A'(A) and B'(B) are kept fixed thoughout the whole utterance.

(ii) Substitute A'_{opt} (A_{opt}) obtained in (i) into equations 4.7(a), 4.7(b), 4.8(a), 4.8(b).

With MT_k = 4.7(a) or 4.7(b)

$$\Delta_x = 1.1706$$

$$B'_k(B_k) = B'_{k-1}(B_{k-1}) + \Delta B_i \; ; \quad i=1,2,3\&4 \qquad \text{4.26}$$

and

$B'_{min}(B_{min})$ and $B'_{max}(B_{max})$ = 4.8(a) & 4.8(b),

maximize SEGSNR with SRFMT as input. This can be done by trying different values for ΔB_1.

(iii) Readjust A'opt (Aopt) obtained in (i) to see if there is any increase in SEGSNR.

Method 2: -

(i) Maximize F in equation 4.27

$$F = SEGSNR - K\left|\bar{L}-R_c\right| \qquad 4.27$$

with K, $\bar{L}$, R_c given by 4.24 and

$$\left.\begin{aligned} &MT_k = \text{eq. } 4.7a \text{ or } 4.7b \quad ; \\ &k \geqslant 2 \quad , \quad MT_1 = MT_{min} \quad , \\ &B'_k(B_k) = B'_{k-1}(B_{k-1}) + \Delta B_i \quad ; \\ &k \geqslant 2 \quad , \quad i=1,2,3,4 \quad , \\ &B'_1(B_1) = B'_{min}(B_{min}) \quad , \\ &B'_{min}(B_{min}) \leqslant B'_k(B_k) \leqslant B'_{max}(B_{max}) \quad , \end{aligned}\right\} \qquad 4.28$$

MT_{min}, MT_{max}, B'_{min}, B'_{max}, B_{min} and B_{max} given by equations 4.17, 4.22, 4.8(a) and 4.8(b) respectively.

The optimum A'(A), andΔB_1 obtained by the two methods match with each other. A'_{opt} (A_{opt}), ΔB_{1opt},

Δ_x, MT_{min}, MT_{max}, B'_{min} (B_{min}) and B'_{max} (B_{max}) are tabulated in Table 4.3.

Table 4.3 Optimum Coder Parameters with SRFMT as input, Tr=16 kb/s, Sr=6.5 kSample/s & N=19.

A'_{opt}	A_{opt}	ΔB_{1opt}	Δ_x	MT_{min}	MT_{max}	B'_{min}	B'_{max}	B_{min}	B_{max}
.982	.959	.0002	1.171	.0113	34.17	.0045	.055	.01	.126

Now we proceed to the third step listed in section 3.1.2. A'(A), ΔB_1, Δ_x, MT_{min}, MT_{max}, B'_{min} (B_{min}) and B'_{max} (B_{max}) are set equal to the optimum values listed in Table 4.3. Set

$$\left.\begin{aligned} MT_1 &= MT_{min} \\ B'_1\ (B_1) &= B'_{min}\ (B_{min}) \\ BFOC_1 &= \text{middle of buffer} = 0 \end{aligned}\right\} \qquad 4.29$$

This is the worst condition for the coder. Now, simulate the coder with a periodic square wave Fig 4.3 as input. T_1 is just long enough to bring the buffer occupancy to its maximum. Buffer occupancy reaches the maximum value one sampling period before MT_k reaches its maximum limit MT_{max}. T_2 is the time required by the multiplier MT_k

to reduce back to MT_{min}. With coder parameters as fixed, buffer occupancy reduces back to 0, middle of buffer, faster than MT_k back to MT_{min} and B'_k (B_k) back to B'_{min} (B_{min}).

Therefore,

$$\begin{aligned} BS &= \text{Buffer size} \\ &= \text{2 times maximum buffer occupancy} \\ &\quad \text{reached by the testing square wave} \\ &= 2 \times BFOC_{max} \end{aligned} \qquad 4.30$$

It is not a must to set the "0" reference point at the middle of the buffer. However, the positive buffer range must be equal to or greater than $BFOC_{max}$. With buffer size given by equation 4.30, the results are tabulated in Table 4.4 with maximum delay D_{max} equals BS/Tr. As the maximum delay D_{max} for coder with MT_k equals to either 4.7(a) or 4.7(b) are well below the maximum allowable delay 100 msec, therefore the optimum parameters listed in Table 4.3 can be adopted.

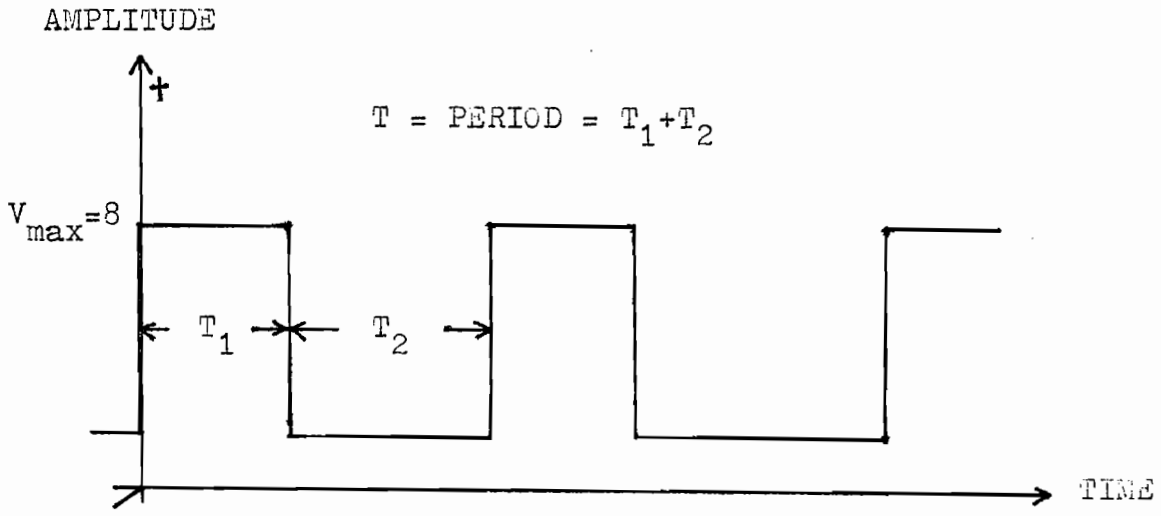


Fig. 4.3 Periodic Testing Square Wave

Table 4.4 T_1, T_2, Buffer Size & Delay with Tr=16 kb/s

Sr= 6.5 kSample/s & N=19

Coder with	Coder with
$MT_k=MT_{k-1}(.982+B'_k(L_{k-1}-\lfloor\frac{N+1}{2}\rfloor)^2)$;	$MT_k=MT_{k-1}(.959+B_k(L_{k-1}-\lfloor\frac{N+1}{2}\rfloor)^2)\cdot 5$
$k \geq 2$, $MT_1=.0113$	$k \geq 2$, $MT_1=.0113$
$.0113 \leq MT_k \leq 34.17$	$.0113 \leq MT_k \leq 34.17$
$B'_k=B'_{k-1}+\Delta B_i$; $i=1,2,3,4$	$B_k=B_{k-1}+\Delta B_i$; $i=1,2,3,4$
$B_1=.0045$	$B_1=.01$
$.0045 \leq B_k \leq .055$	$.01 \quad B_k \quad .126$
$B_1=0.0002=2\Delta B_2=-2\Delta B_3=-2\Delta B_4$	$B_1=.0002=2\Delta B_2=-2\Delta B_3=-2\Delta B_4$
$\Delta_x = 1.1706$	$\Delta_x = 1.1706$

T_1 msec	T_2 msec	$BFOC_{max}$ Bits	BS Bits	D_{max} msec	T_1 msec	T_2 msec	$BFOC_{max}$ Bits	BS Bits	D_{max} msec
26.6	42.2	401	802	51	45.7	59.1	560	1120	70

The objective coder performance is tabulated in Table 4.5. $\bar{L}$ is given by equation 4.24 and

$$\text{Efficiency} = \text{Entropy}/(\bar{L} \times \text{Log}_2 2) = \text{Entropy}/\bar{L} \qquad 4.31$$

The decoded output at the reciever for coder with M_i given by 3.3(b) is shown in Fig 4.4(a), (b) & (c). The decoded output for coder with M_i given by 3.3(a) is almost the same as Fig 4.4 and therefore is not appended. A close examination of Fig. 4.1 & Fig. 4.4 shows that they are very much identical with each other. The buffer occupancy waveform and the sliding SNR for M_i given by 3.3(a) and 3.3(b) are shown in Fig 4.5, 4.6, 4.7 and 4.8 respectively. These figures together with Table 4.5 show that coder M_i equals to 3.3(a) is almost the same as coder with M_i equals to 3.3(b). With M_i given by 3.3(b), a longer buffer is required.

Table 4.5 Coder Performance with Tr=16 kb/s, Sr=6.5 kSample/s, N=19 and INPUT=SRFMT

Coder with	Coder with
$MT_k=MT_{k-1}(.982+B'_k(L_{k-1}-\lfloor\frac{N+1}{2}\rfloor)^2)$;	$MT_k=MT_{k-1}(.959+B_k(L_{k-1}-\lfloor\frac{N+1}{2}\rfloor)^2)$
$k \geq 2$, $MT_1 = .0113$	$k \geq 2$, $MT_1 = .0113$
$0.0113 \leq MT_k \leq 34.17$	$0.0113 \leq MT_k \leq 34.17$
$B'_k=B'_{k-1}+\Delta B_i$; $i=1,2,3,4$,	$B_k=B_{k-1}+\Delta B_i$; $i=1,2,3,4$,
$k \geq 2$, $B'_1=.0045$	$k \geq 2$, $B_1=.01$
$.0045 \leq B'_k \leq .055$	$.01 \leq B_k \leq .126$
$\Delta B_1=.0002=2\Delta B_2=-2\Delta B_3=-2\Delta B_4$	$\Delta B_1=.0002=2\Delta B_2=-2\Delta B_3=-2\Delta B_4$
$\Delta_x=1.1706$	$\Delta_x=1.1706$
BS = 802 Bits	BS = 1120 Bits

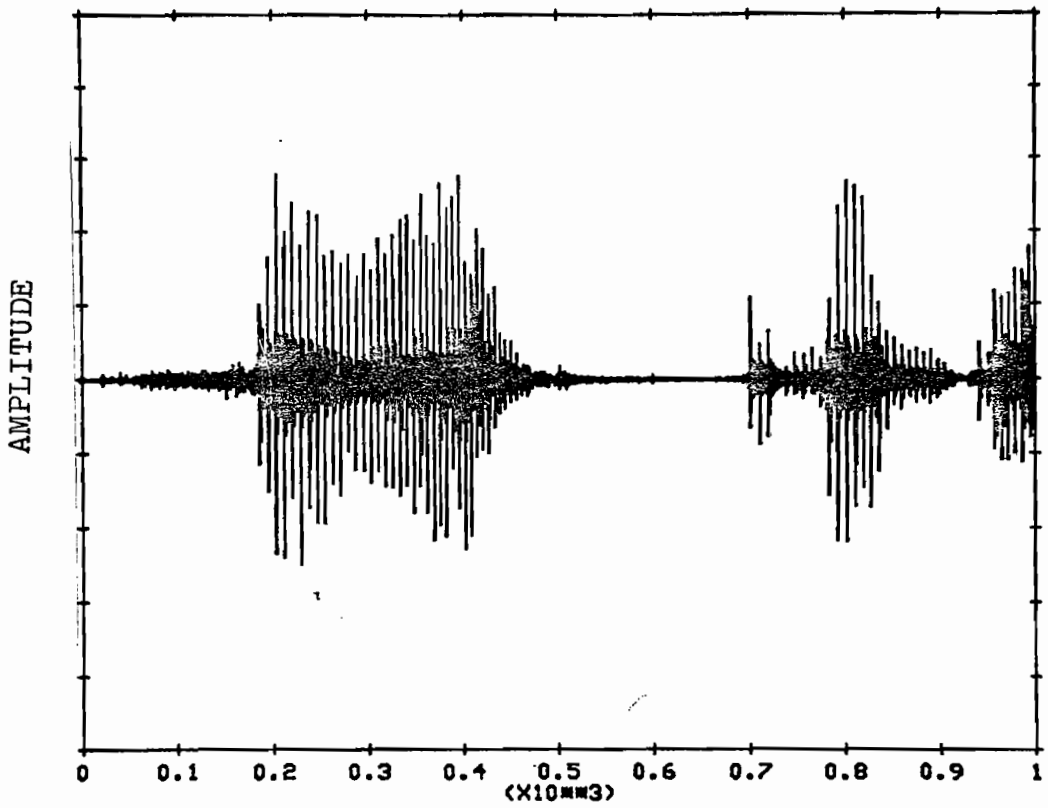
SEGSNR dB	ENTROPY bits	$\bar{L}$ b/sample	EFFICIENCY	SEGSNR dB	ENTROPY bits	$\bar{L}$ b/sample	EFFICIENCY
12.47	2.34	2.44	.959	12.49	2.34	2.44	.959

Amplitude Distribution of Quantized Output

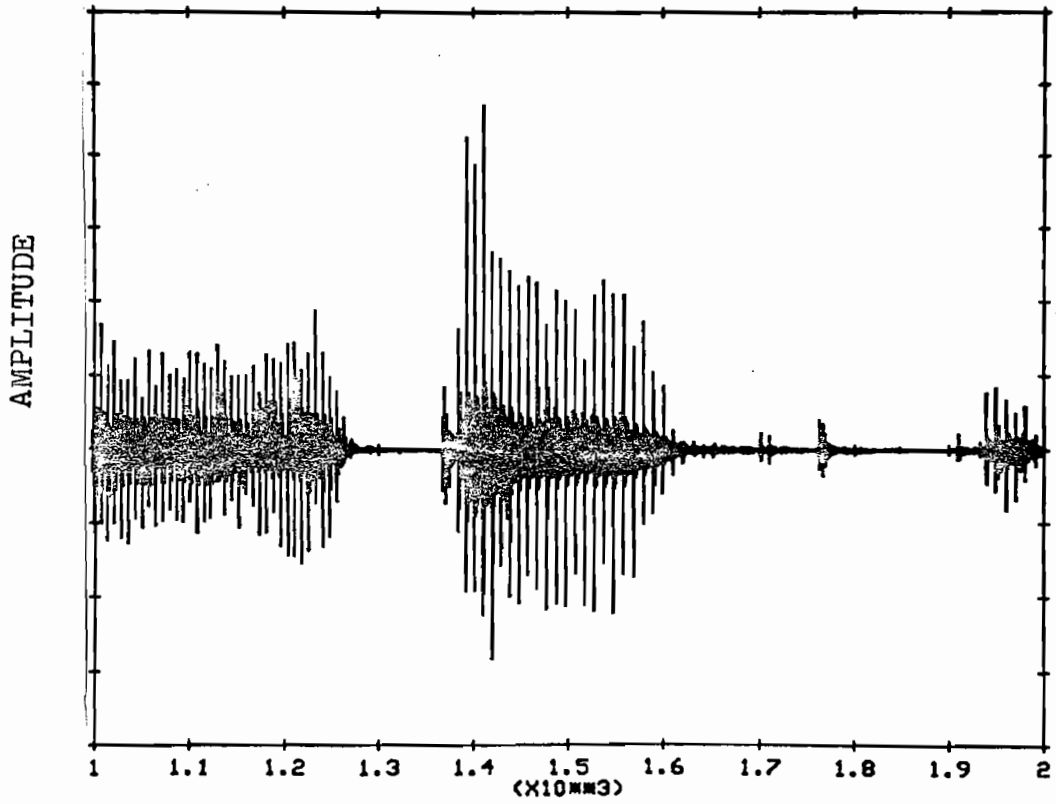
Level Number	Dist.	Level Number	Dist.
1	0.23%	19	0.099%
2	0.094%	18	0.033%
3	0.13%	17	0.16%
4	0.21%	16	0.19%
5	0.26%	15	0.25%
6	0.44%	14	0.46%
7	1.6%	13	0.97%
8	6.7%	12	4.5%
9	24%	11	20%
10	39.5%		

Amplitude Distribution of Quantized Output

Level Number	Dist.	Level Number	Dist.
1	0.188%	19	0.066%
2	0.094%	18	0.051%
3	0.21%	17	0.12%
4	0.16%	16	0.22%
5	0.28%	15	0.28%
6	0.43%	14	0.43%
7	1.6%	13	1.0%
8	6.8%	12	4.4%
9	23.9%	11	20%
10	39.5%		


AMPLITUDE
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)


AMPLITUDE
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

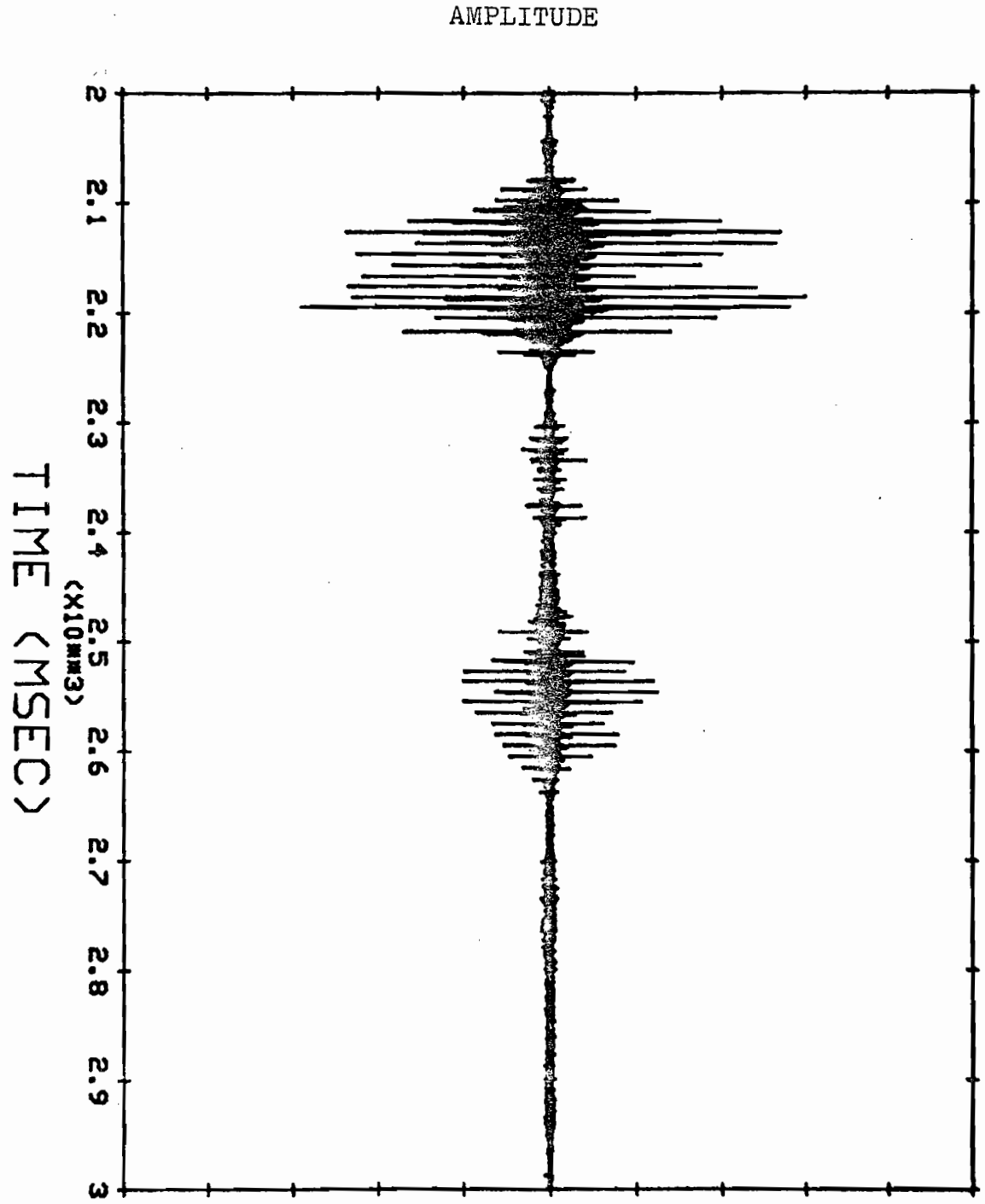
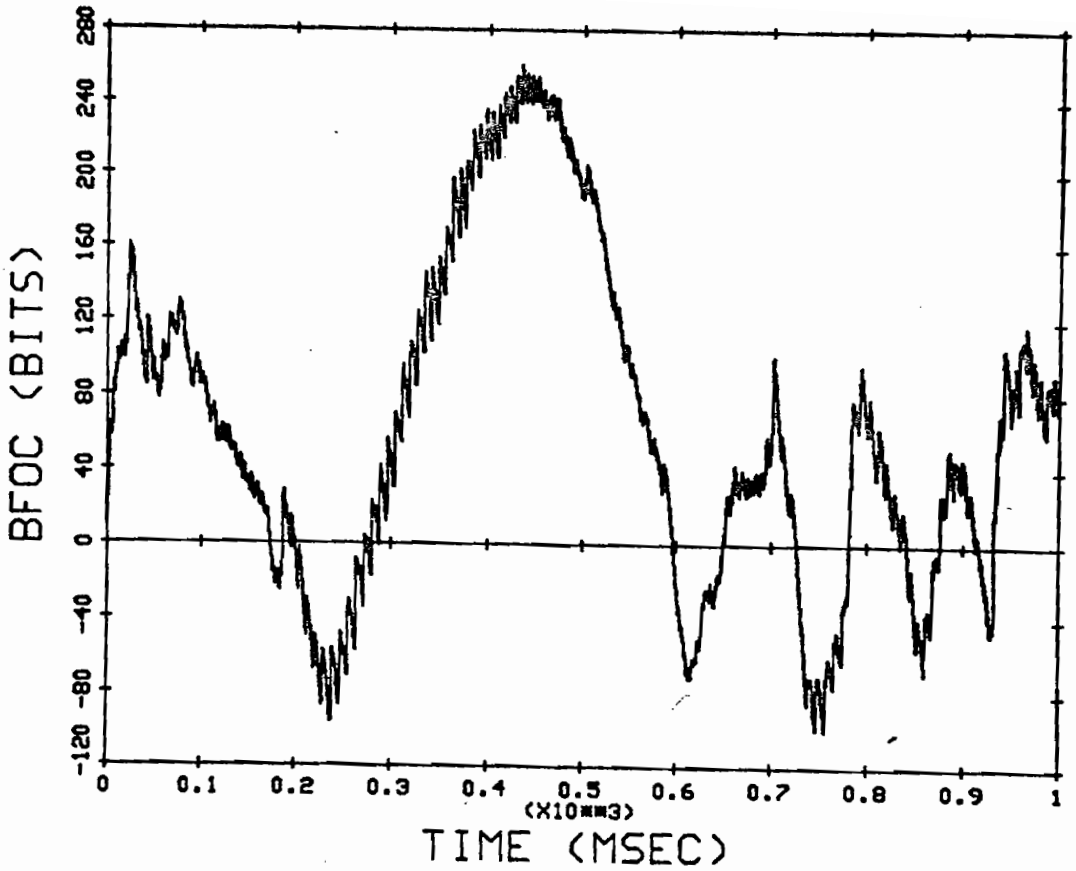
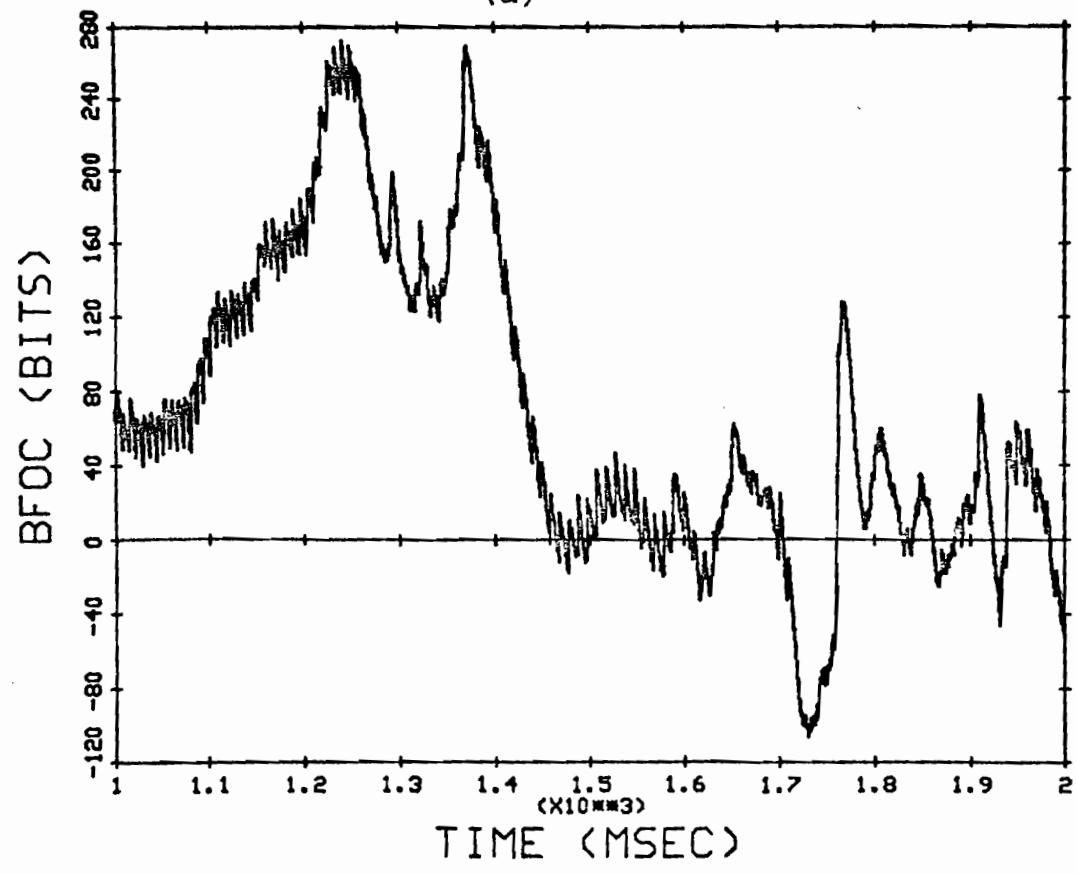


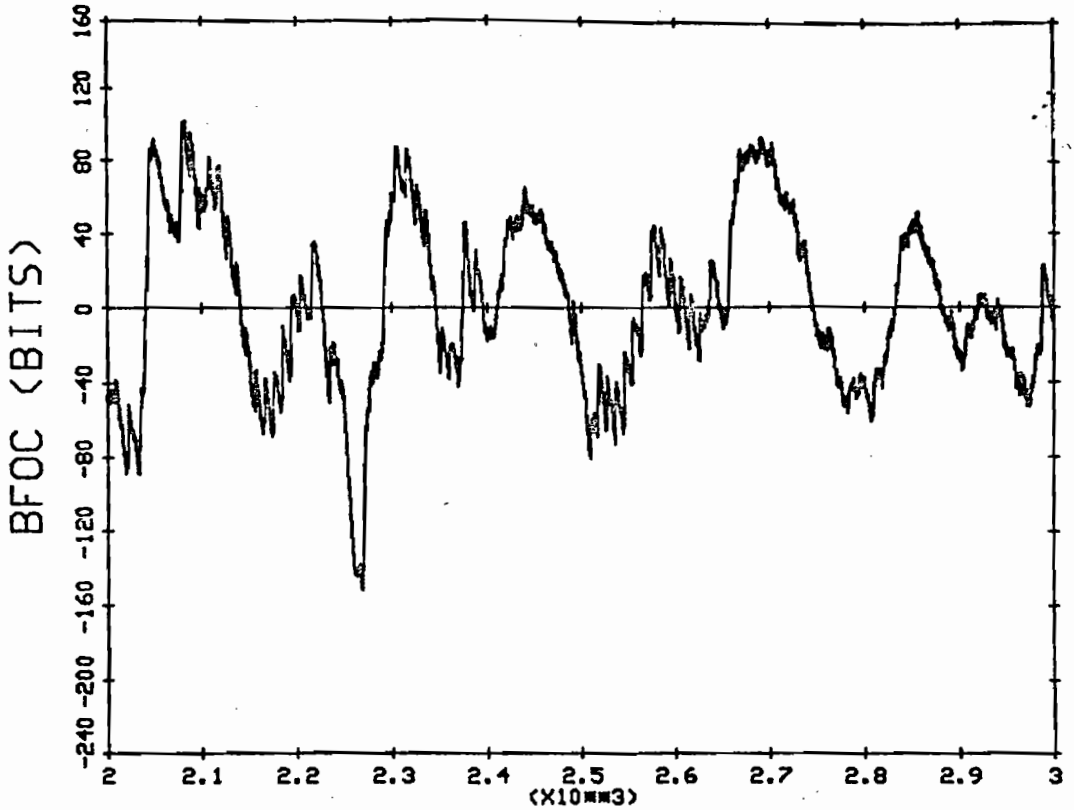
Fig. 4.4 (a), (b) & (c) Decoded Output at Receiver verse Time with $M_i=(.95975 + B_k x(i-1)^2)^{0.5}$; $k \geq 2$, $B_1=0.01$


BFOC (BITS)
280
240
200
160
120
80
40
0
-40
-80
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)

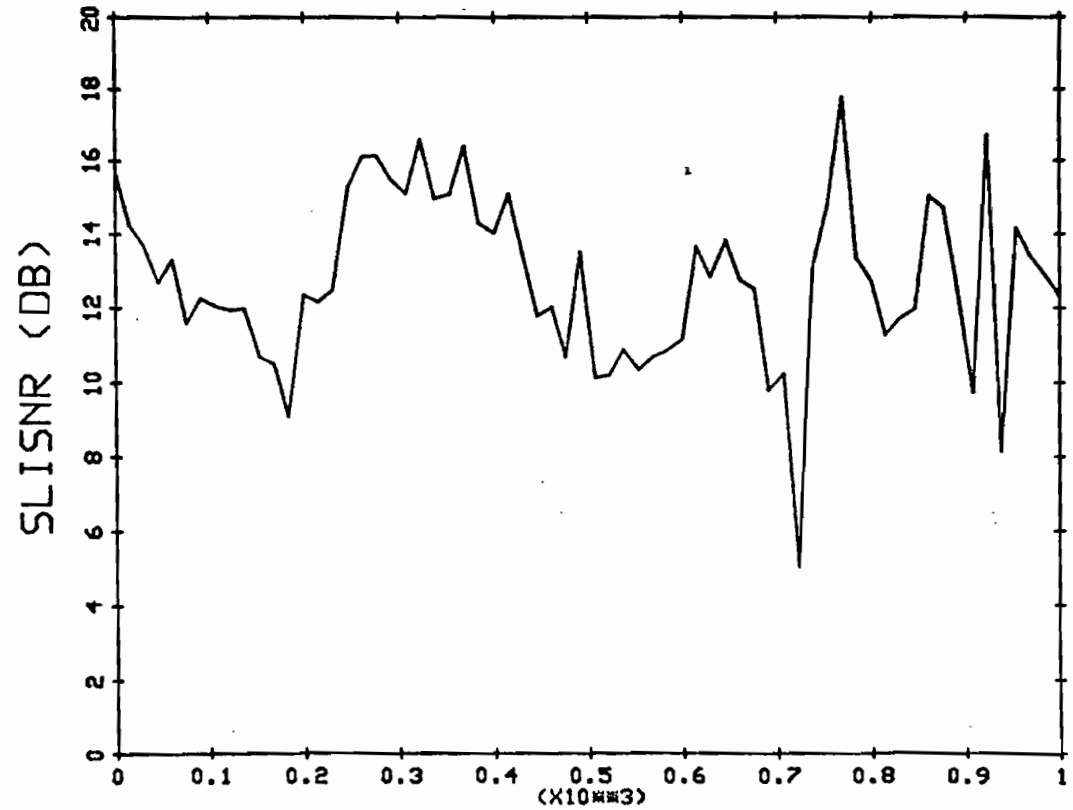

BFOC (BITS)
280
240
200
160
120
80
40
0
-40
-80
-120
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

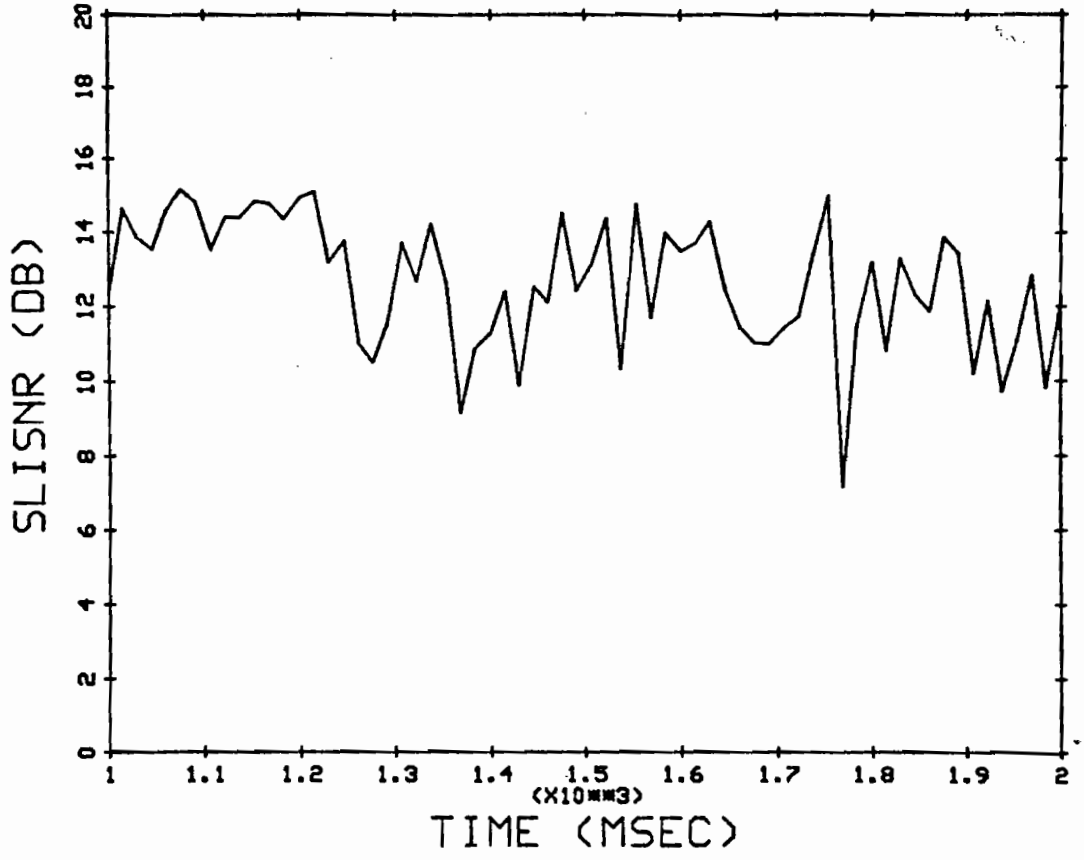


(c)

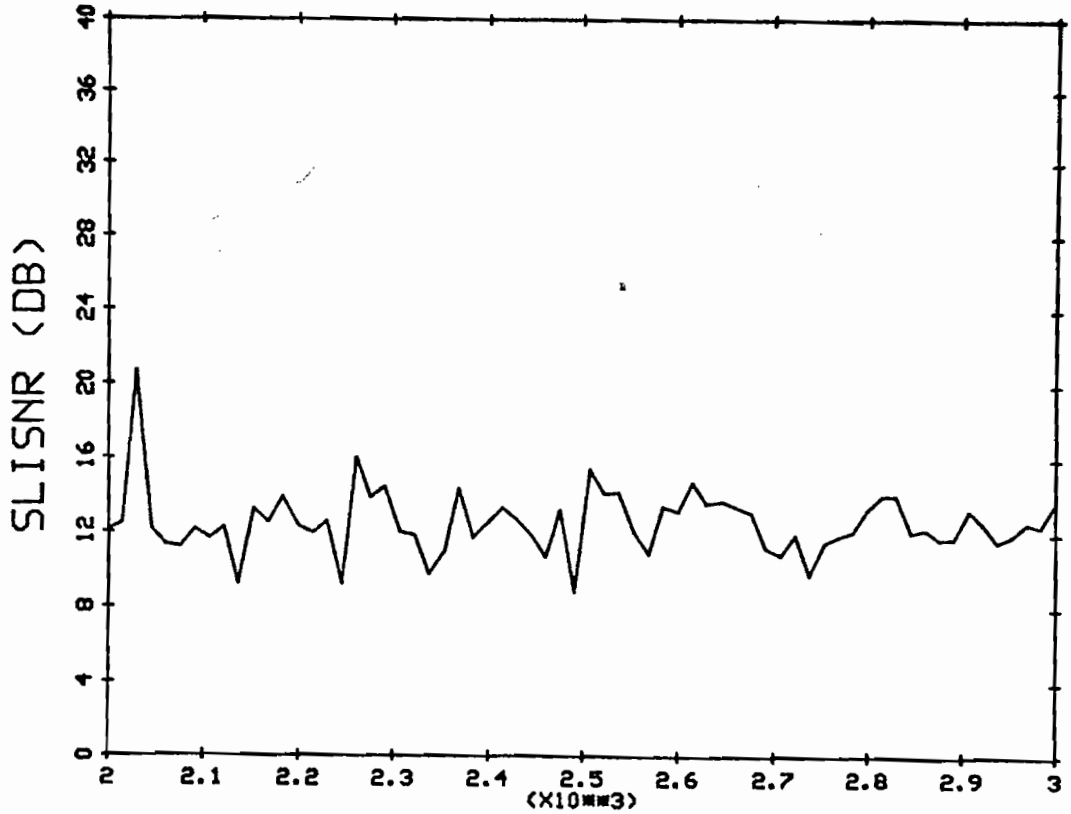
Fig. 4.5 (a), (b) & (c) BFOC verse Time with $M_i = 0.982 + B'_k (i-1)^2$; $k \geq 2$, $B'_1 = 0.0045$



(a)

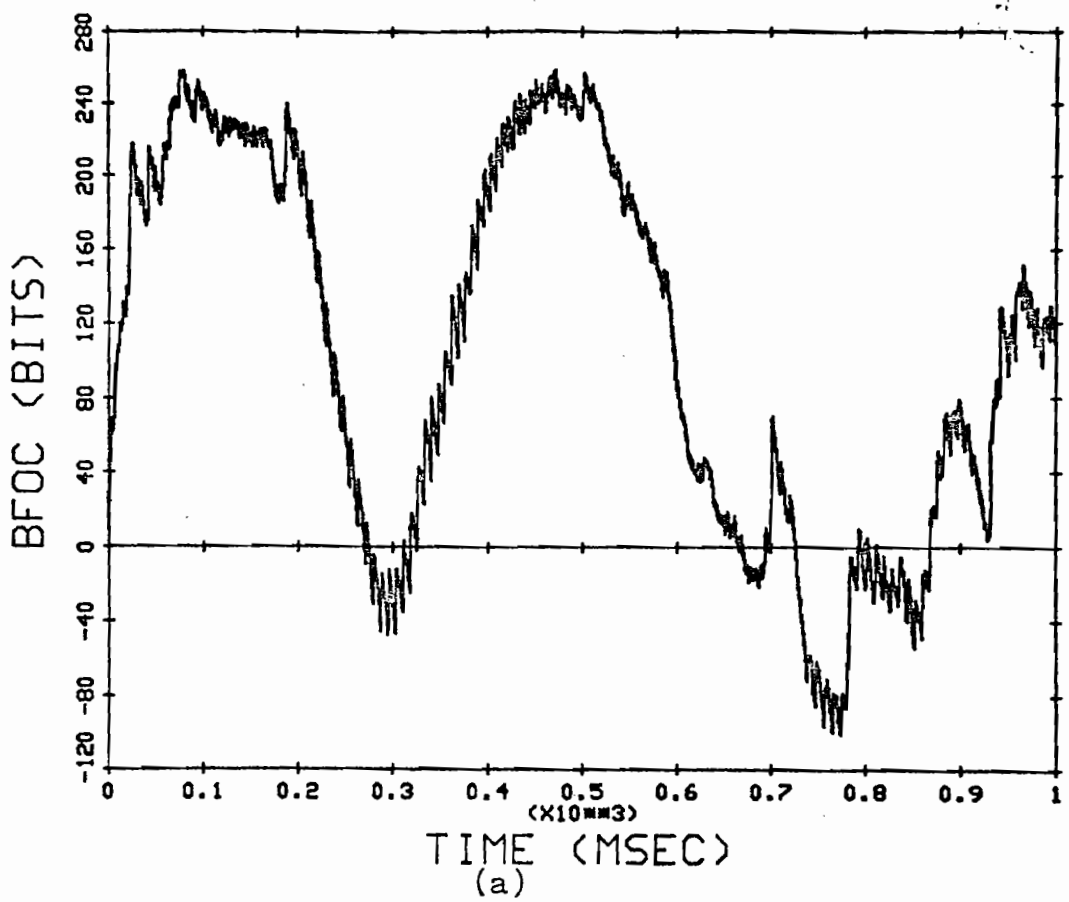


(b)

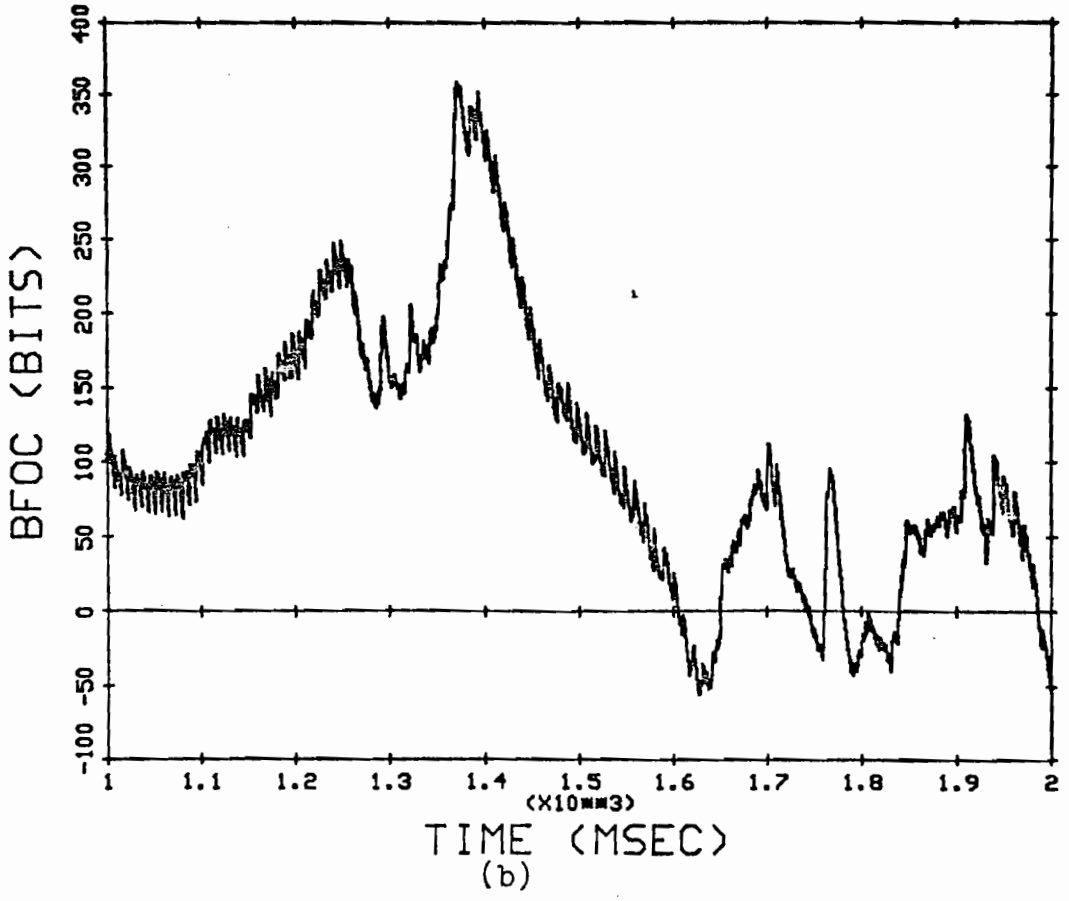


(c)

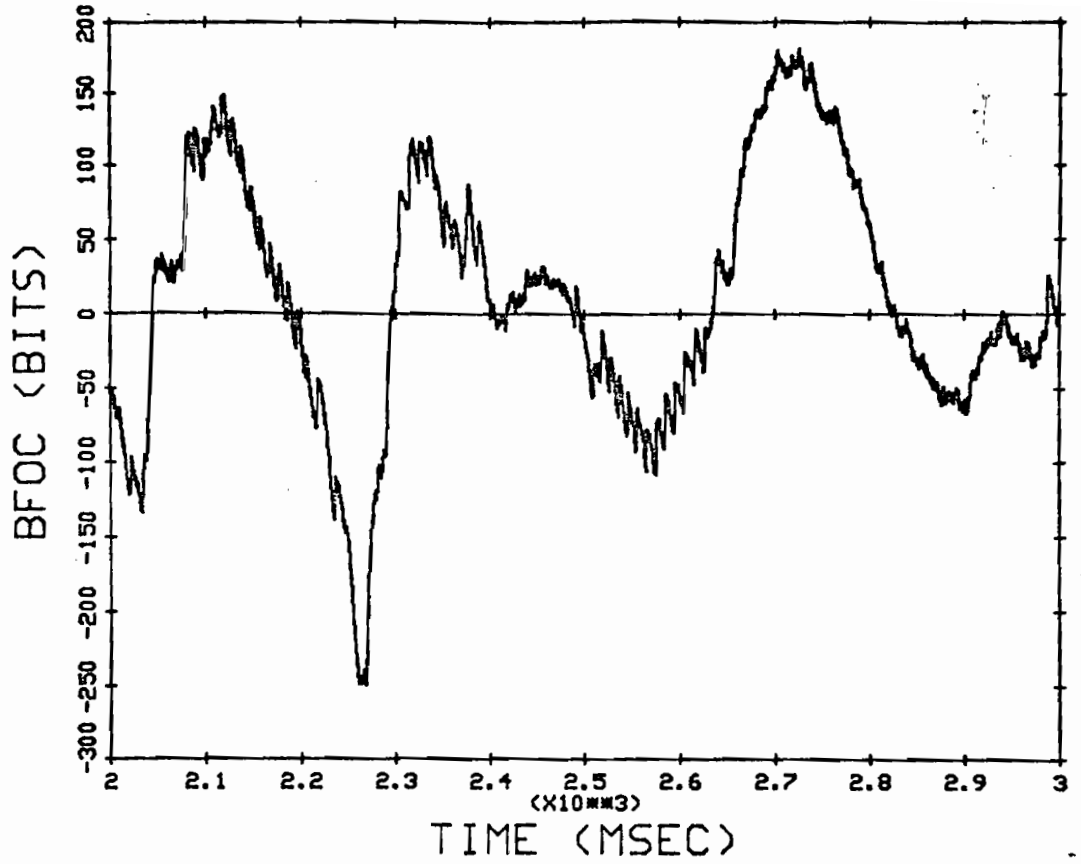
Fig. 4.6 (a),(b)&(c) SLISNR verse Time with $M_i = .982 + B_k'(i-1)^2$; $k \geq 2$, $B_1' = .0045$


BFOC (BITS)
280
240
200
160
120
80
40
0
-40
-80
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)

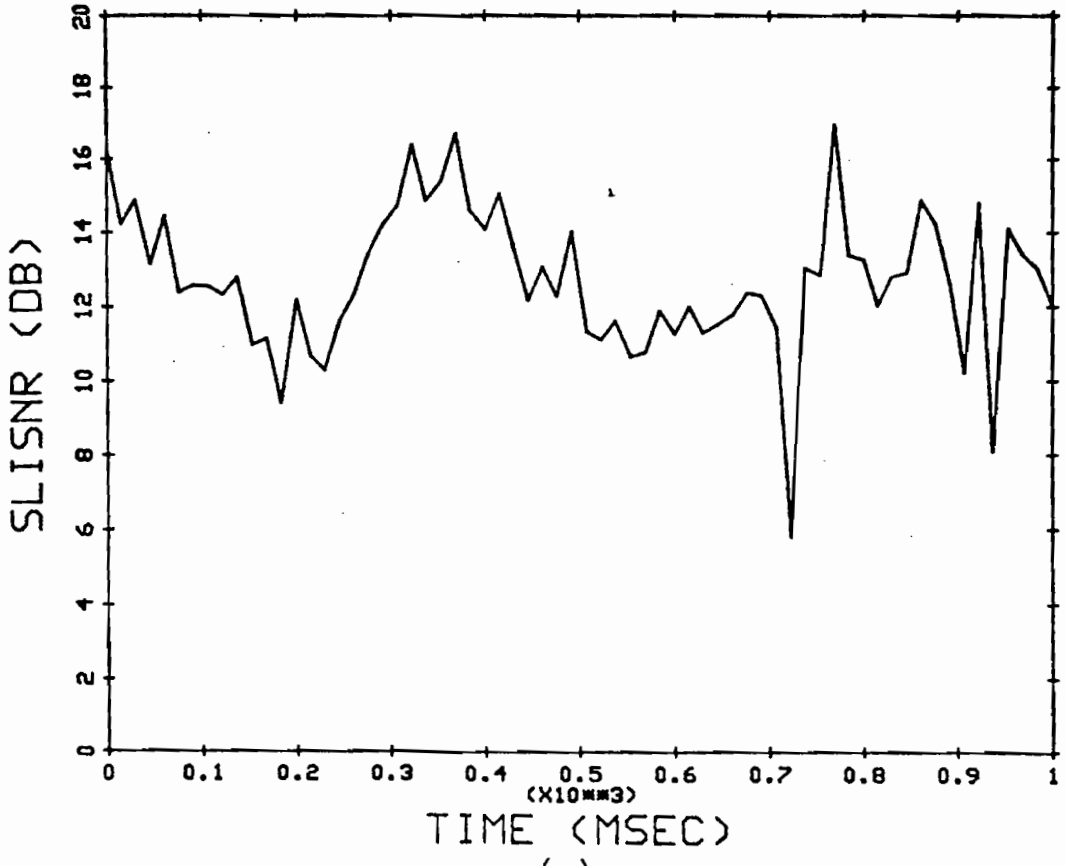

BFOC (BITS)
400
350
300
250
200
150
100
50
0
-50
-100
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

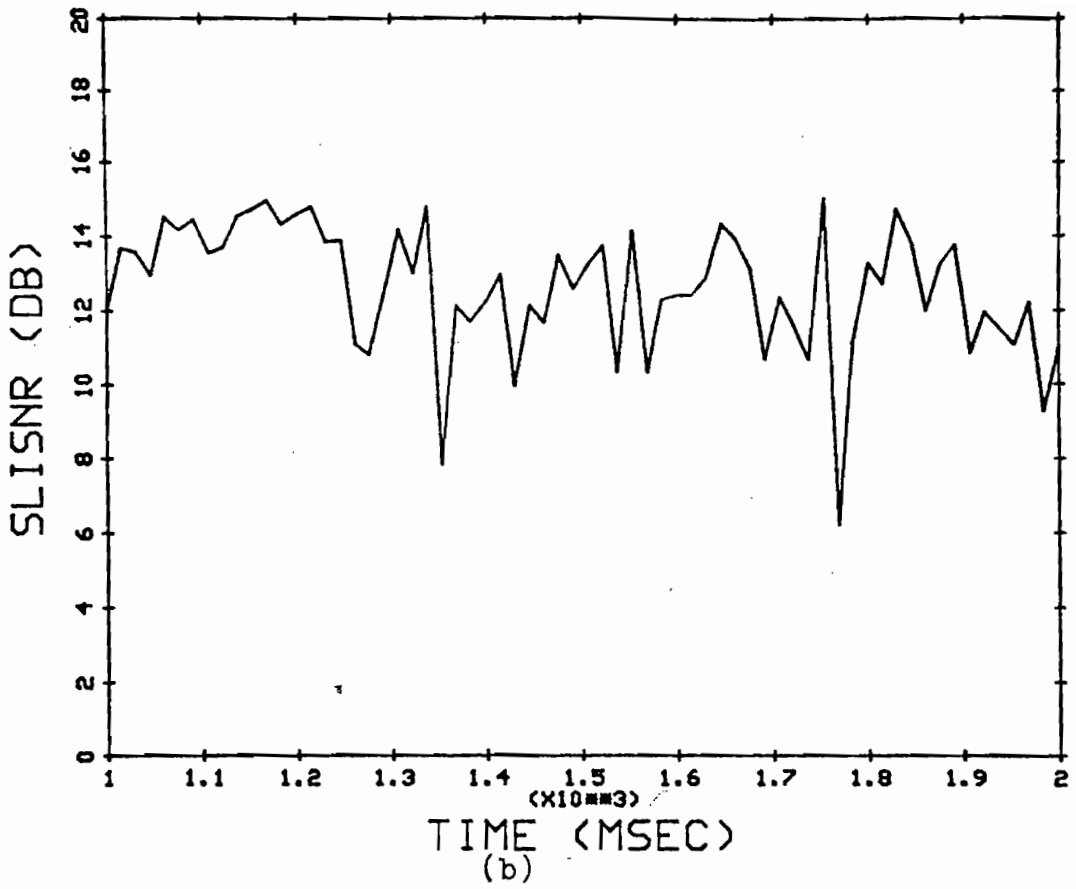


(c)

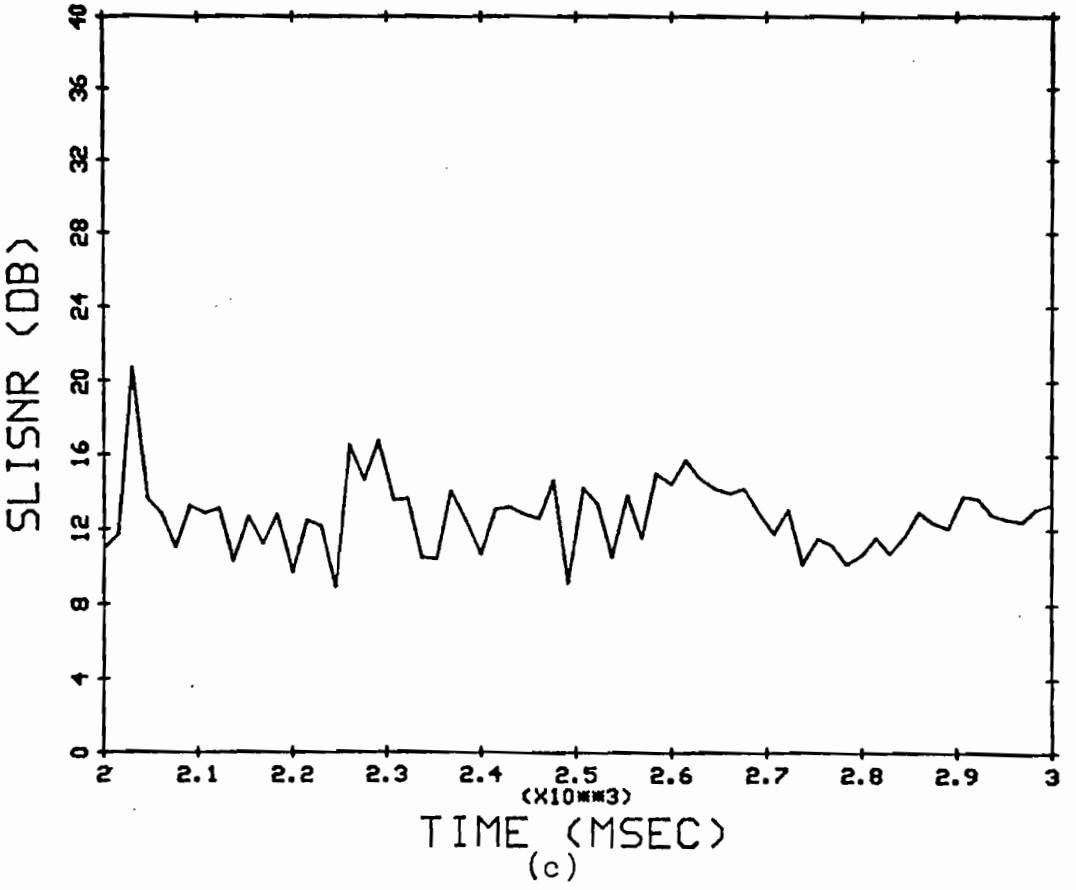
Fig. 4.7 (a), (b) & (c) BFOC verse Time with $M_i=((.95975 + B_k(i-1)^2)^{0.5})$; $k \geq 2$, $B_1 = 0.01$



(a)



(b)



(c)

Fig. 4.8 (a), (b) & (c) SLISNR verse Time with $M_i=(.95975 + B_k(i-1)^2)^{0.5}$; $k \geq 2$, $B_1=0.01$

4.2.2 Adaptive Quantization step-size

The coder shown in Fig 3.3 is simulated in this subsection. For convenience, we repeat here the equations which govern the coder operation.

$$\tilde{\sigma}_k(\text{estimated input STD}) = \alpha_k \tilde{\sigma}_{k-1} \quad ; \quad k \geq 2 \tag{4.32}$$

$$\text{where } \alpha_k = ((1-\gamma)Y_{L_{k-1}}{}^2 + \gamma)^{.5} \quad ; \quad 0 < \gamma < 1 \tag{4.33}$$

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta xi} \quad ; \quad k \geq 2 \ , \quad i=1,2,3,4 \tag{4.34}$$

$$\text{and} \quad \Delta_{xk} \leq \Delta_{xmax}=2 \quad \text{for N=odd=19 levels} \tag{4.35}$$

To simplify the search for $\delta_{\Delta xi}$ s, set

$$\left.\begin{aligned} \delta_{\Delta x2} &= \delta_{\Delta x1}/2 \\ \delta_{\Delta x3} &= -\delta_{\Delta x1}/4 \\ \delta_{\Delta x4} &= -\delta_{\Delta x1}/3 \end{aligned}\right\} \tag{4.36}$$

Now we follow the steps listed in subsection 3.2.2. Parallel the same reasons in subsection 4.2.1, we have

$$V_{max} = 20 \tag{4.37}$$

$$\tilde{\sigma}_{min} = 0.0113 \tag{4.38}$$

$$\tilde{\sigma}_{max} = 2V_{max}/\Delta_{xmax} = V_{max} \tag{4.39}$$

With channel rate equal to 2.5 bits per sampling period and the code-words of level 8 and level 12 are 4 bits and 5 bits long respectively, therefore

$$\Delta_{xmin} = 1/2.5 = 0.4 \qquad 4.40$$

Step (iii) requires the optimization of γ and $\delta_{\Delta x1}$. Now we only have two parameters to optimize. This was done by changing one parameter while keeping the other fixed alternatively to maximize the segmental SNR. The optimum results are tabulated in Table 4.6. Now we proceed to the forth

Table 4.6 Optimum γ & $\delta_{\Delta x1}$ with Tr = 16 kb/s, Sr = 6.5 kSample/s, N = 19, $\Delta_{xmax} = 2$, $\Delta_{xmin} = 0.4$, $\tilde{\sigma}_{min}$ = 0.0113 & $\tilde{\sigma}_{max} = V_{max} = 20$

γ_{opt}	$\delta_{\Delta x1}$
0.96	0.02

step listed in subsection 3.2.2. γ & $\delta_{\Delta x1}$ are set to the optimum values listed in Table 4.6. For the worst condition, set

$$\left.\begin{aligned} &\Delta_{x1} = \Delta_{xmin} = 0.4 \\ &\tilde{\sigma}_1 = \tilde{\sigma}_{min} = 0.0113 \\ &BFOC_1 = 0 = \text{middle of buffer} \end{aligned}\right\} \qquad 4.41$$

The coder is then simulated with a square wave as shown in Fig 4.3. Again T_1 is just long enough to bring the buffer occupancy to its maximum. Here, it is noted that $\tilde{\sigma}_k$ reaches its maximum long before the buffer occupancy reaches its maximum. T_2 is the time required for $\tilde{\sigma}_k$ to reduce back to $\tilde{\sigma}_{min}$. In this case, $\tilde{\sigma}_k$ reduces back to $\tilde{\sigma}_{min}$ much faster than the buffer occupancy. But, Δ_{xk} shrinks to Δ_{xmin} slower than the buffer occupancy. Therefore the required buffer size is still given by equation 4.30 The results are tabulated

Table 4.7 T_1, T_2, Buffer Size & D_{max} with Tr=16 kb/s, Sr=6.5 kSample/s & N=19

Coder with

$$\tilde{\sigma}_k = \tilde{\sigma}_{k-1} \times (.96 + (1-.96) \times (Y_{L_{k-1}})^2)^{0.5} \quad ; \quad k \geq 2, \quad \tilde{\sigma}_1 = .0113$$

$$\tilde{\sigma}_{min} = .0113 \leq \tilde{\sigma}_k \leq 20 = \tilde{\sigma}_{max}$$

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta xi} \quad ; \quad i=1,2,3,4, \quad k \geq 2$$

$$\delta_{\Delta x1} = .02 = 2\,\delta_{\Delta x2} = -4\,\delta_{\Delta x3} = -3\,\delta_{\Delta x4}$$

$$0.4 = \Delta_{xmin} \leq \Delta_{xk} \leq \Delta_{xmax} = 2$$

$$\Delta_{x1} = \Delta_{xmin}$$

T_1 msec	T_2 msec	$BFOC_{max}$ Bits	BS Bits	D_{max} msec
12.3	36.9	351	702	43.9

in Table 4.7. As D_{max} in Table 4.7 is well below 100 msec, therefore the optimum values listed in Table 4.6 can be adopted. The coder performance is listed in Table 4.8. The waveform of the decoded output at the receiver, the buffer occupancy and the sliding SNR are shown in Fig 4.9, 4.10 and 4.11 respectively.

Table 4.8 Coder Performance with Tr=16 kb/s, Sr=6.5 kSample/s, N=19 & Input=SRFMT

Coder with

$$\gamma_k = \gamma_{k-1}(0.96 + (1-.96)(Y_{L_{k-1}})^2)^{0.5} \quad ; \quad k \geq 2,$$

$$.0113 = \gamma_{min} \leq \gamma_k \leq \gamma_{max} = 20; \qquad \gamma_1 = \gamma_{min}$$

$$\Delta_{xk} = \Delta_{xk-1} + \delta_{\Delta xi} \quad ; \quad k \geq 2, \quad i=1,2,3,4 \quad , \Delta_{x1} = \Delta_{xmin} = 0.4$$

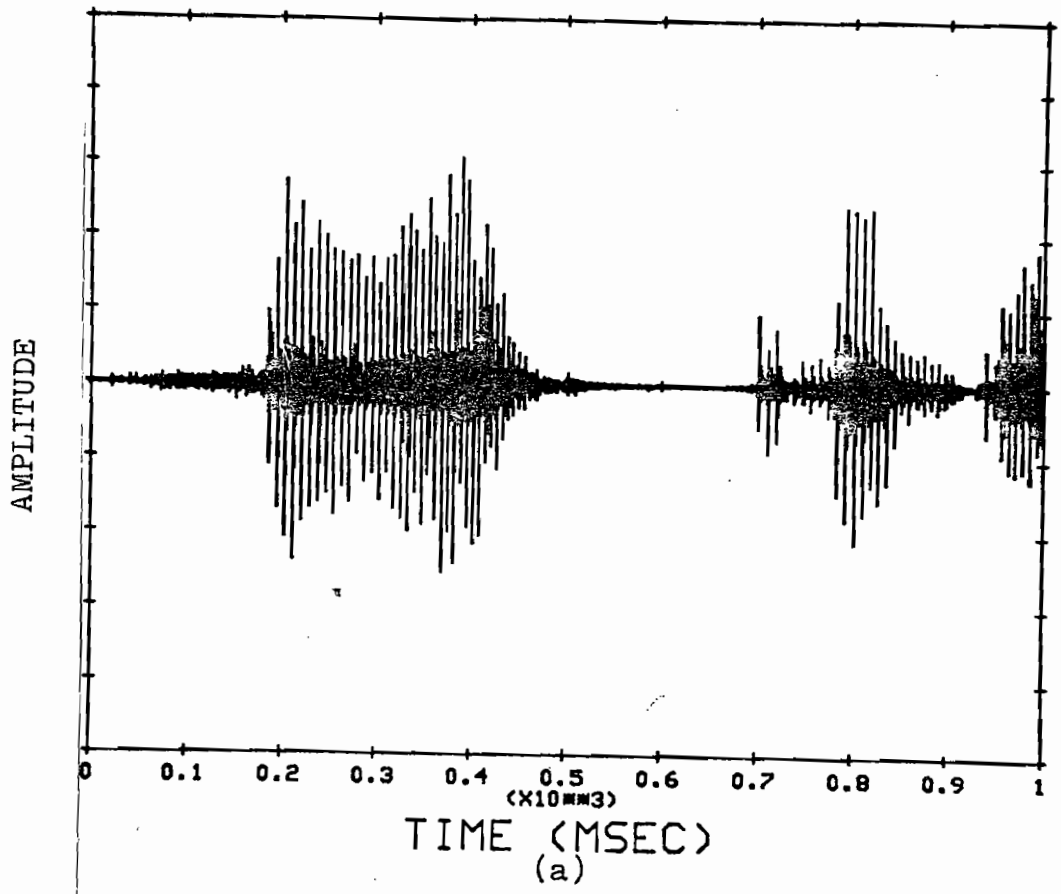
$$\delta_{\Delta x1} = 0.02 = 2\delta_{\Delta x2} = -4\delta_{\Delta x3} = -3\delta_{\Delta x4}$$

Buffer Size = 702 Bits

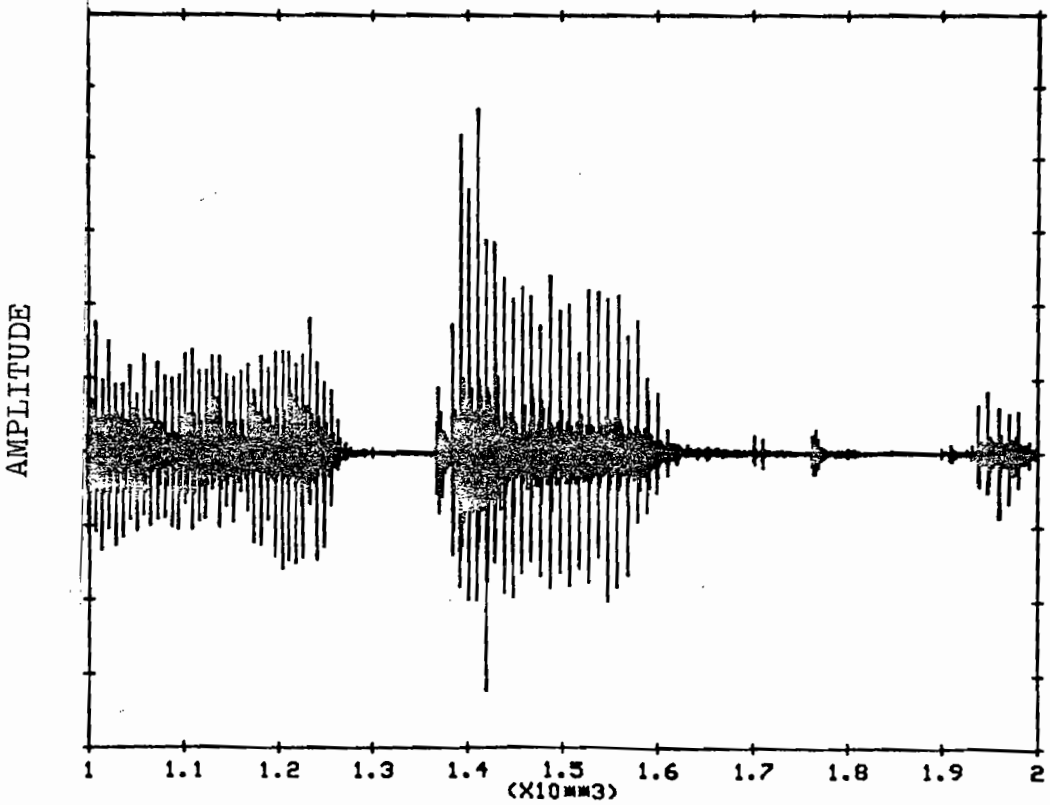
AMPLITUDE DISTRIBUTION OF QUANTIZED OUTPUT

Level Number	Distribution	Level Number	Distribution
1	0.216%	19	0.07%
2	0.155%	18	0.05%
3	0.136%	17	0.12%
4	0.16 %	16	0.26%
5	0.25 %	15	0.22%
6	0.43 %	14	0.4 %
7	1.4 %	13	0.89%
8	6.7 %	12	4.46%
9	23.9 %	11	20.0 %
10	39.89 %		

SEGSNR dB	ENTROPY BITS	$\bar{L}$ b/sample	EFFICIENCY
12.2	2.32	2.42	0.957


AMPLITUDE
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)


AMPLITUDE
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

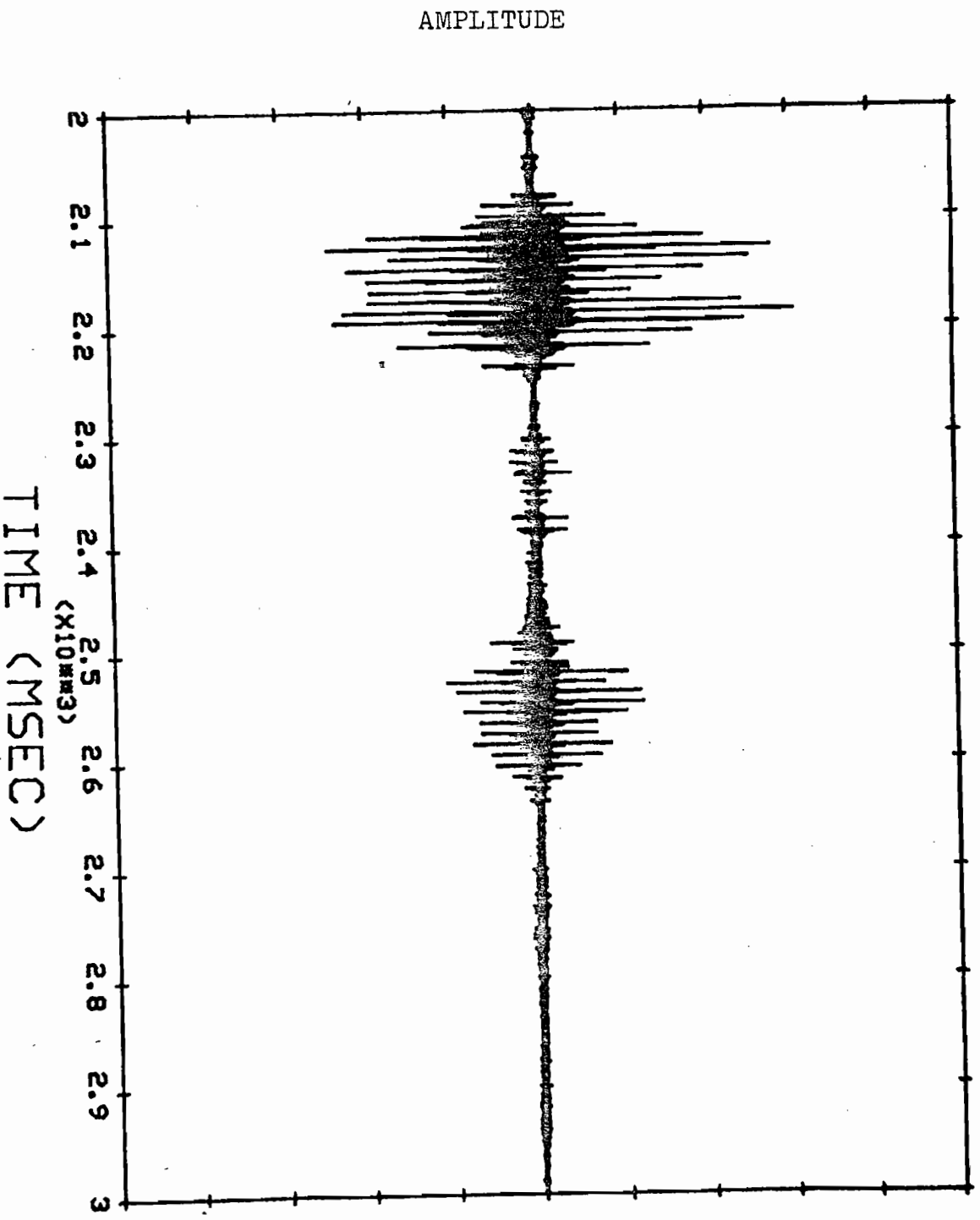
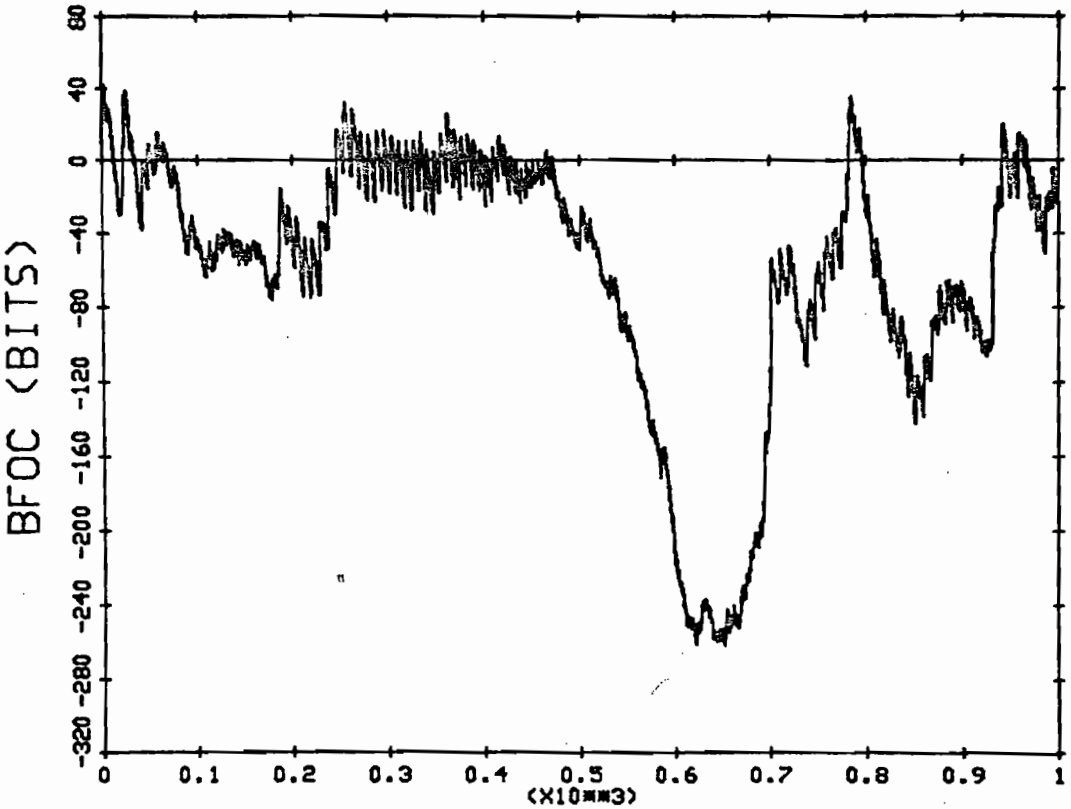
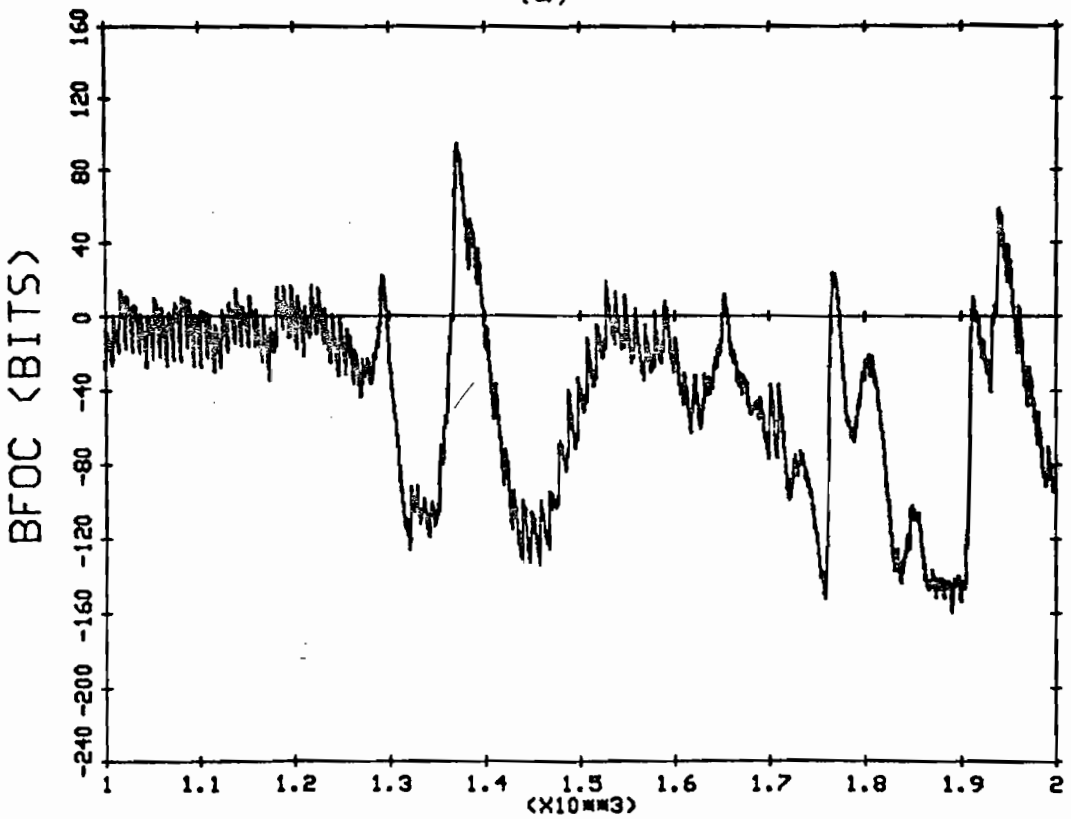


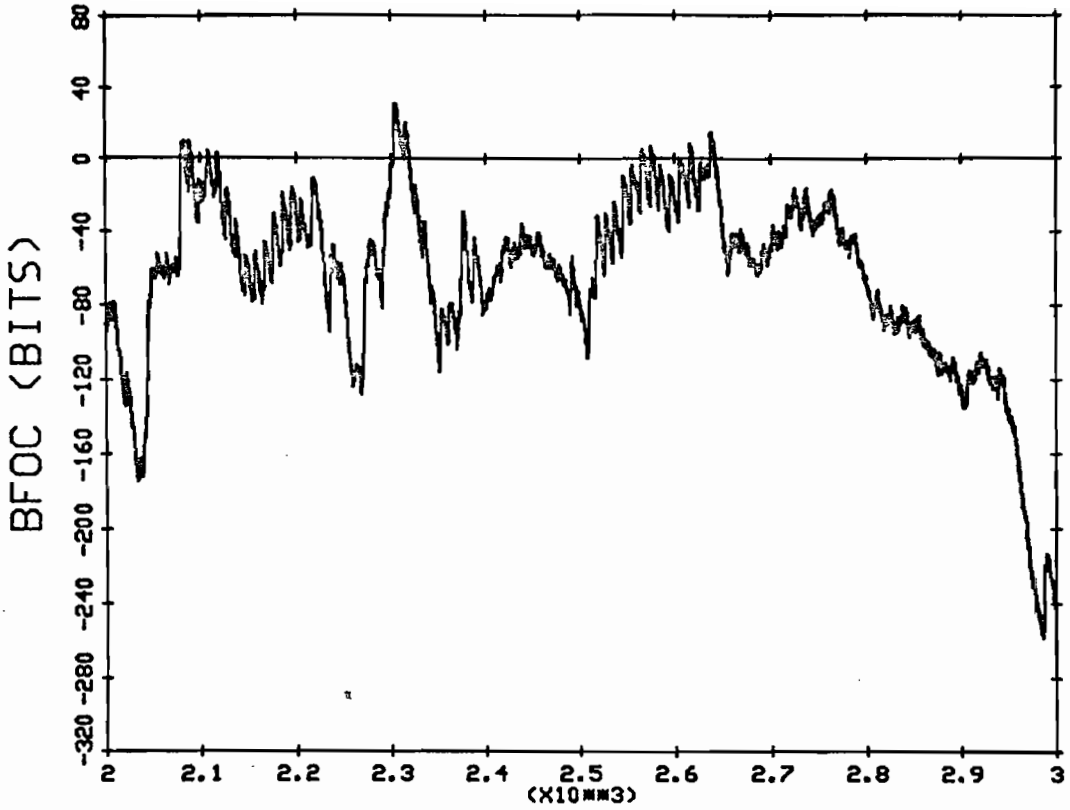
Fig. 4.9 (a), (b) & (c) Decoded Output at Receiver verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size


BFOC (BITS)
80
40
0
-40
-80
-120
-160
-200
-240
-280
-320
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)

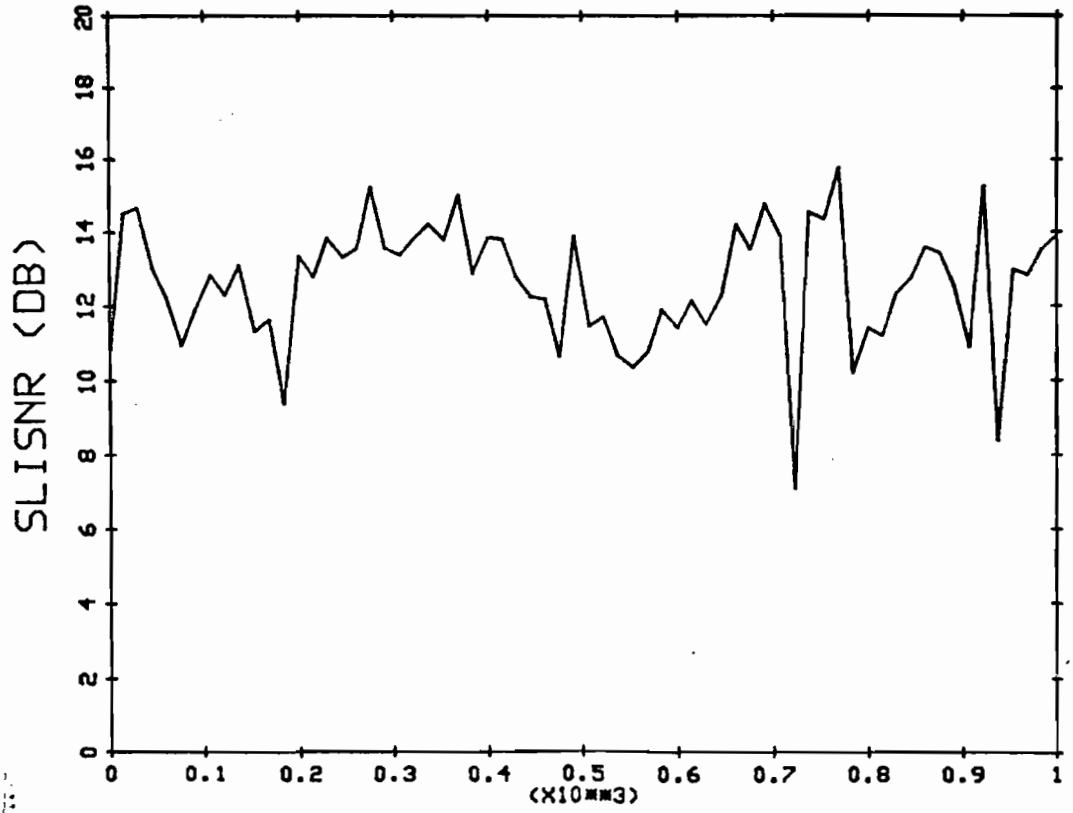

BFOC (BITS)
160
120
80
40
0
-40
-80
-120
-160
-200
-240
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

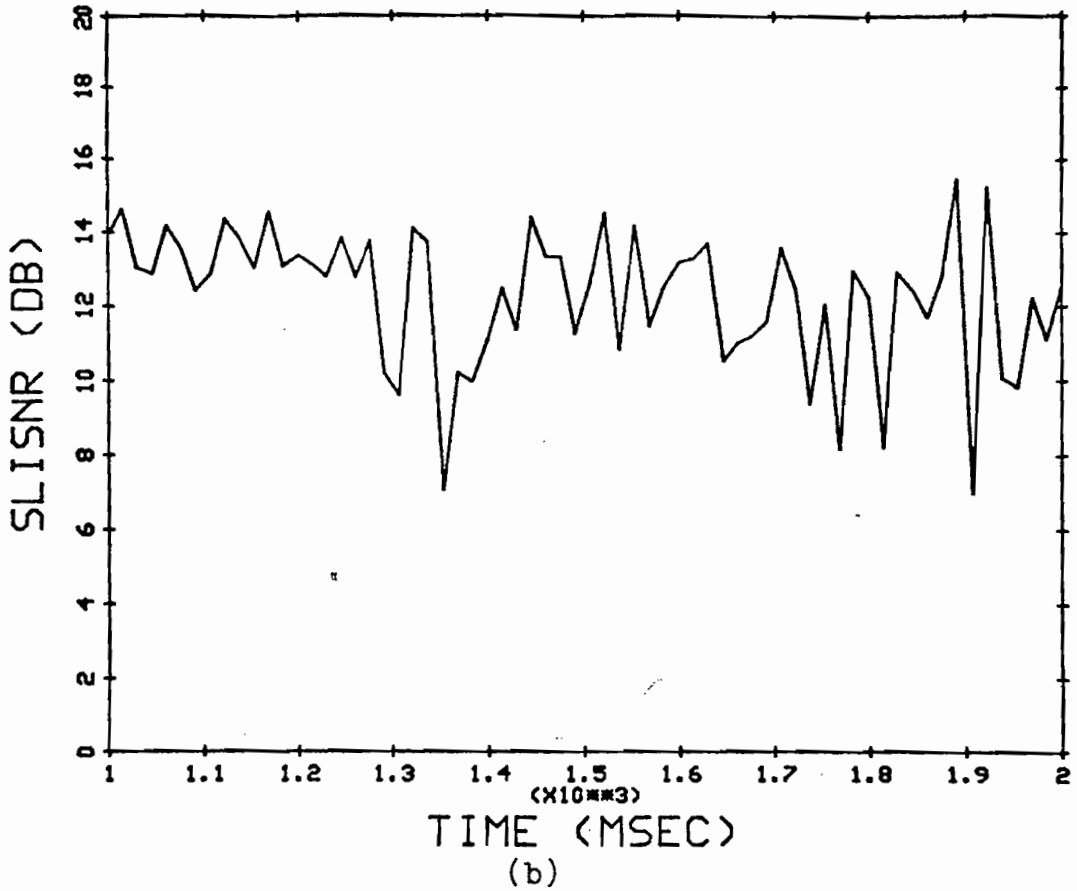


(c)

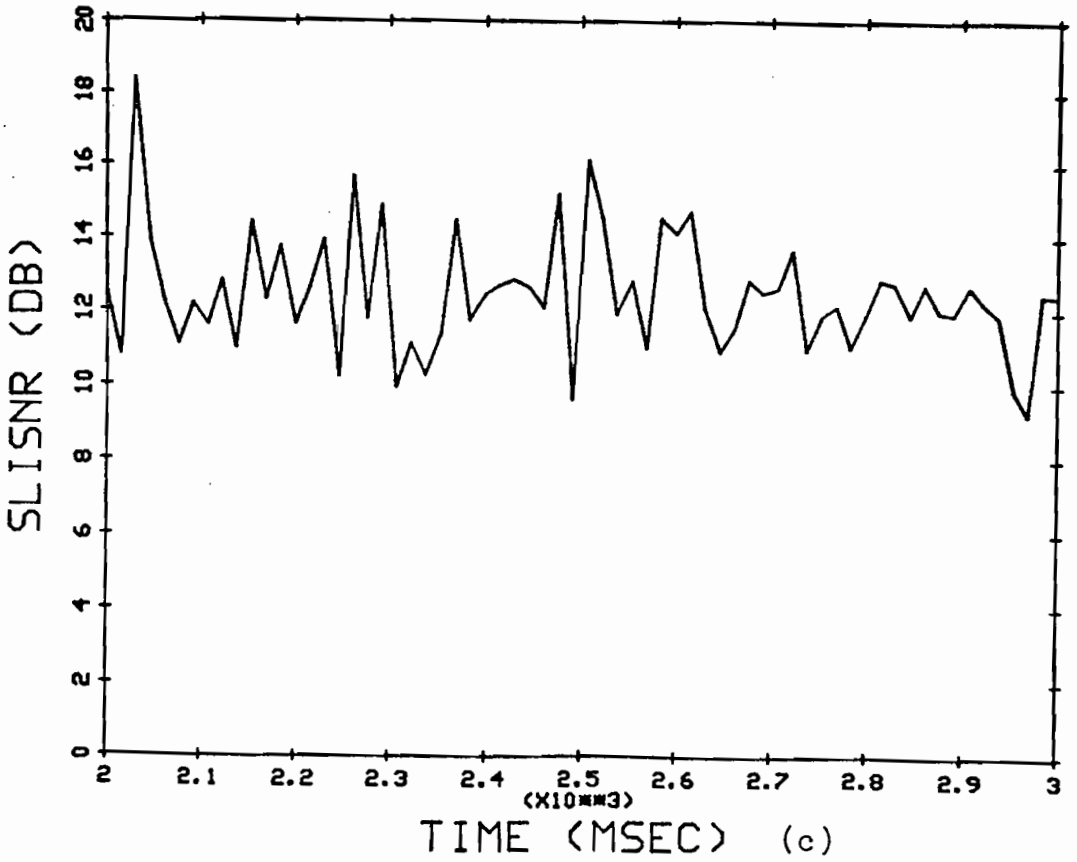
Fig. 4.10 (a),(b)&(c) BFOC verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size



(a)



(b)



(c)

Fig. 4.11 (a),(b)&(c) SLISNR verse Time with Buffer Management Achieved by Adaptive Quantization Step-Size

4.2.3 Block Quantization

In this simulation, we assume that the problem of quantizing and encoding the side information can be separated from the quantization and encoding of the input samples. Therefore, the side information, quantization step-size, is not quantized in this simulation. However, we do indeed taken into account the number of bits B_s required to send this side information for meaningful comparisons with other coders. As the quantization step-size is the only side information we have to send off, therefore B_s depends only on the range of variation of the step-size. Parallel the same reasons in subsection 4.22, we have

$$\Delta_{xmin} = 0.0113 \times 0.4 = 0.00452 \qquad 4.42$$

and

$$\Delta_{xmax} = 2 \times 20 = 40 \qquad 4.43$$

In this simulation, B_s is set equal to 20 bits because

$$2^{20} = 1048576 \qquad 4.44$$

and therefore, the resolution is small enough even for fixed equal length encoding of the step-size. Equation 3.26 thus reduces to

$$B_T \leq B_N \leq (Tr/Sr) \times M - 20 = B_{max} \qquad 4.45$$

where

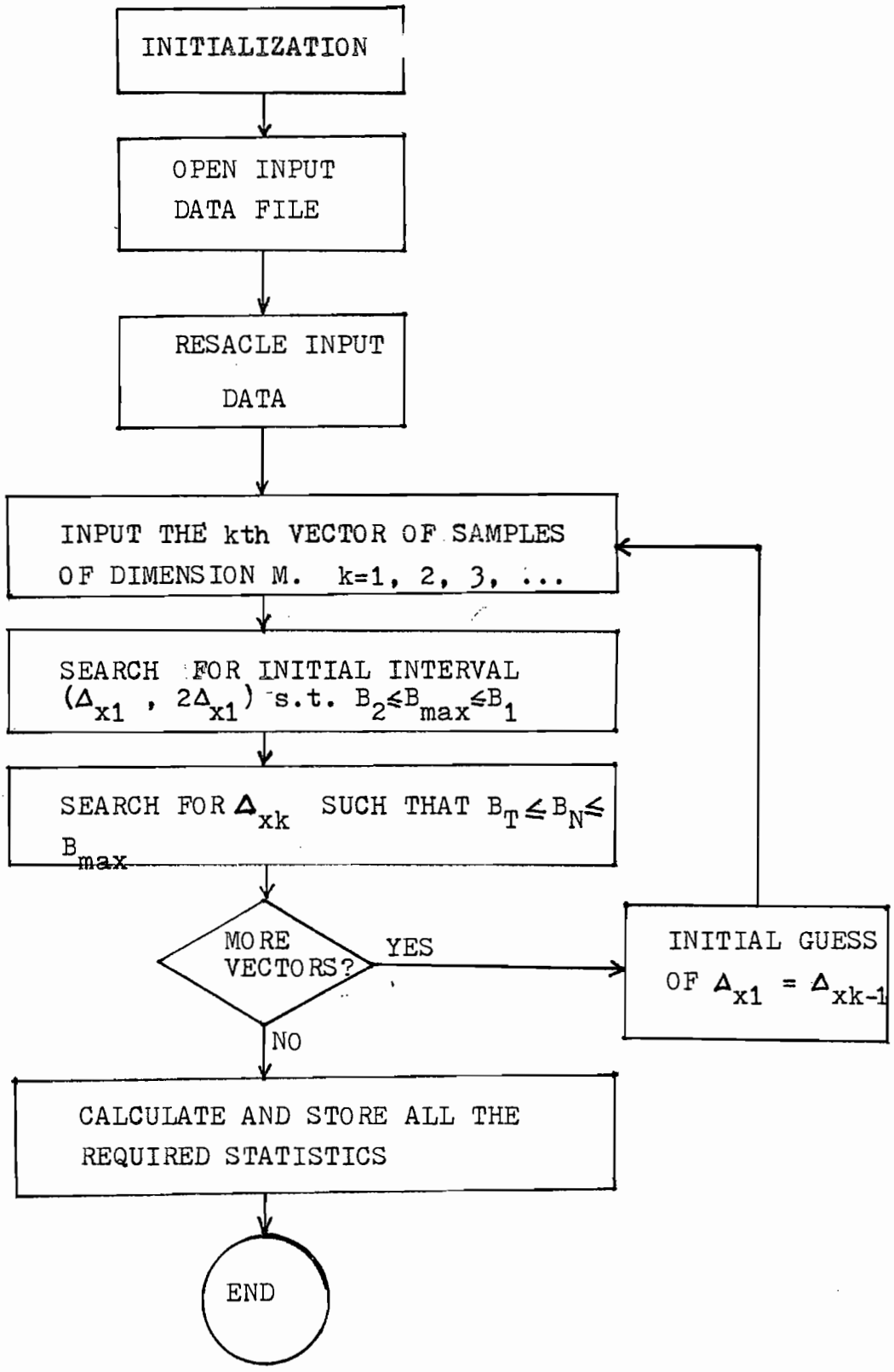
$$\left.\begin{aligned} &B_{max} = \text{buffer size} = BS \\ &B_T = B_{max} - (N-1)/2 \\ &Tr = 16\ KB/S \\ &Sr = 6.5\ \text{samples/s} \\ &N = 19 \\ &M = \text{input vector dimension} \end{aligned}\right\} 4.46$$

The flow-chart for searching the right step-size for the kth vector that results in B_N satisfies equation 4.45 is shown in Fig. 4.12.

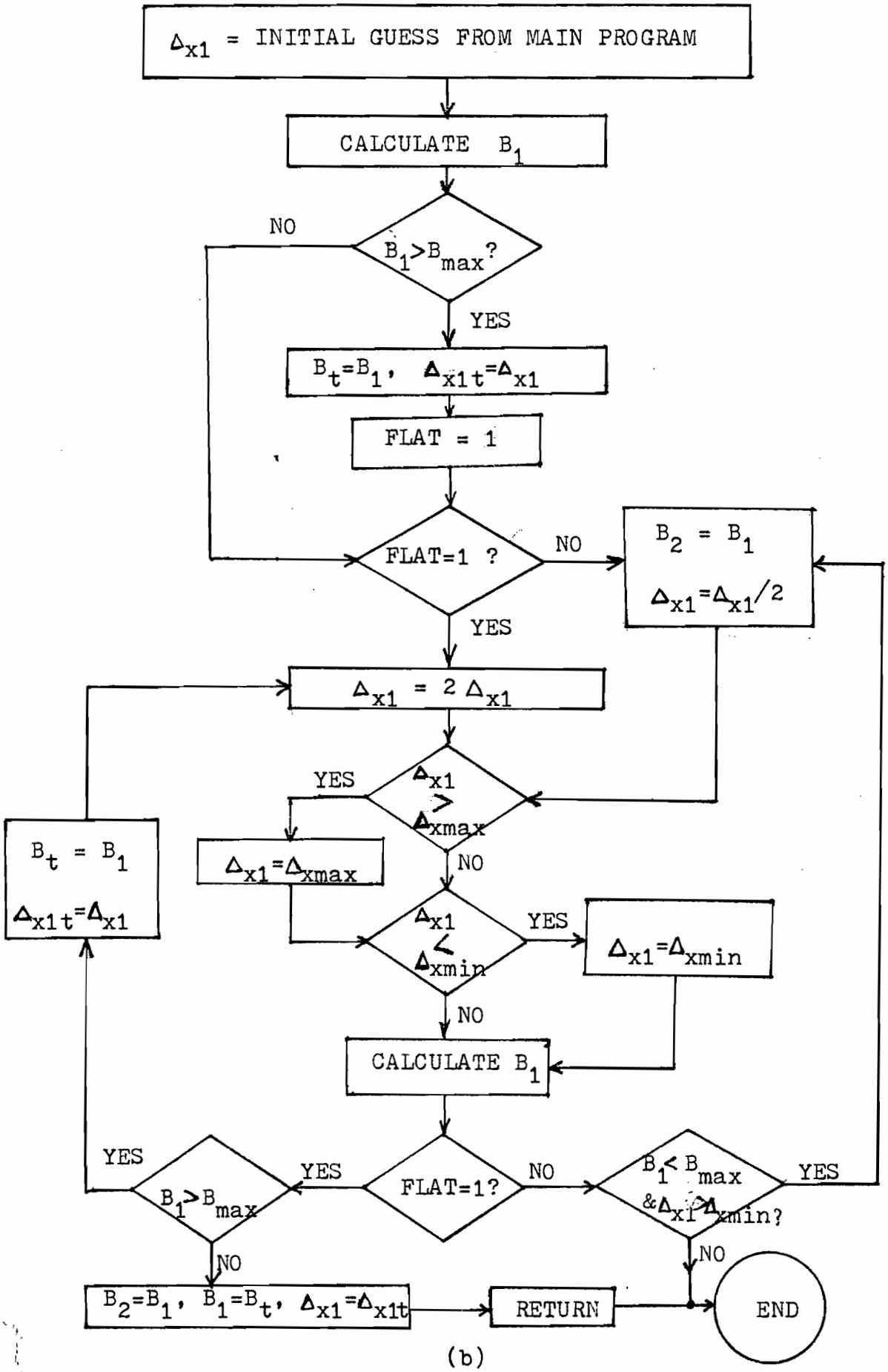
Now, we first proceed to find the optimum vector dimension M. Fig. 4.13 shows the segmental SNR verse vector dimension curve. The maximum is around M=300 samples. With M=300 samples, transmission delay

$$D = 2 \times M/S_r = 2 \times 300/6.5 \text{ msec} = 92.3 \text{ msec} \qquad 4.47$$

This is close to 100 msec, but is still acceptable. The coder performance with M=300 samples is tabulated in table 4.9. The decoded output waveform at the receiver and the sliding SNR are shown in Fig. 4.14 and Fig 4.15 respectively.


INITIALIZATION
OPEN INPUT DATA FILE
RESACLE INPUT DATA
INPUT THE kth VECTOR OF SAMPLES OF DIMENSION M. k=1, 2, 3, ...
SEARCH FOR INITIAL INTERVAL $(\Delta_{x1}, 2\Delta_{x1})$ s.t. $B_2 \leq B_{max} \leq B_1$
SEARCH FOR Δ_{xk} SUCH THAT $B_T \leq B_N \leq B_{max}$
MORE VECTORS?
YES
INITIAL GUESS OF $\Delta_{x1} = \Delta_{xk-1}$
NO
CALCULATE AND STORE ALL THE REQUIRED STATISTICS
END

(a)


Δ_{x1} = INITIAL GUESS FROM MAIN PROGRAM
CALCULATE B_1
NO
$B_1 > B_{max}$?
YES
$B_t = B_1$, $\Delta_{x1t} = \Delta_{x1}$
FLAT = 1
FLAT=1 ?
NO
$B_2 = B_1$
$\Delta_{x1} = \Delta_{x1}/2$
YES
$\Delta_{x1} = 2\,\Delta_{x1}$
YES
$\Delta_{x1} > \Delta_{xmax}$
$B_t = B_1$
$\Delta_{x1t} = \Delta_{x1}$
$\Delta_{x1} = \Delta_{xmax}$
NO
$\Delta_{x1} < \Delta_{xmin}$
YES
$\Delta_{x1} = \Delta_{xmin}$
NO
CALCULATE B_1
YES
$B_1 > B_{max}$
YES
FLAT=1?
NO
$B_1 < B_{max}$ & $\Delta_{x1} > \Delta_{xmin}$?
YES
NO
$B_2 = B_1$, $B_1 = B_t$, $\Delta_{x1} = \Delta_{x1t}$
RETURN
END

(b)

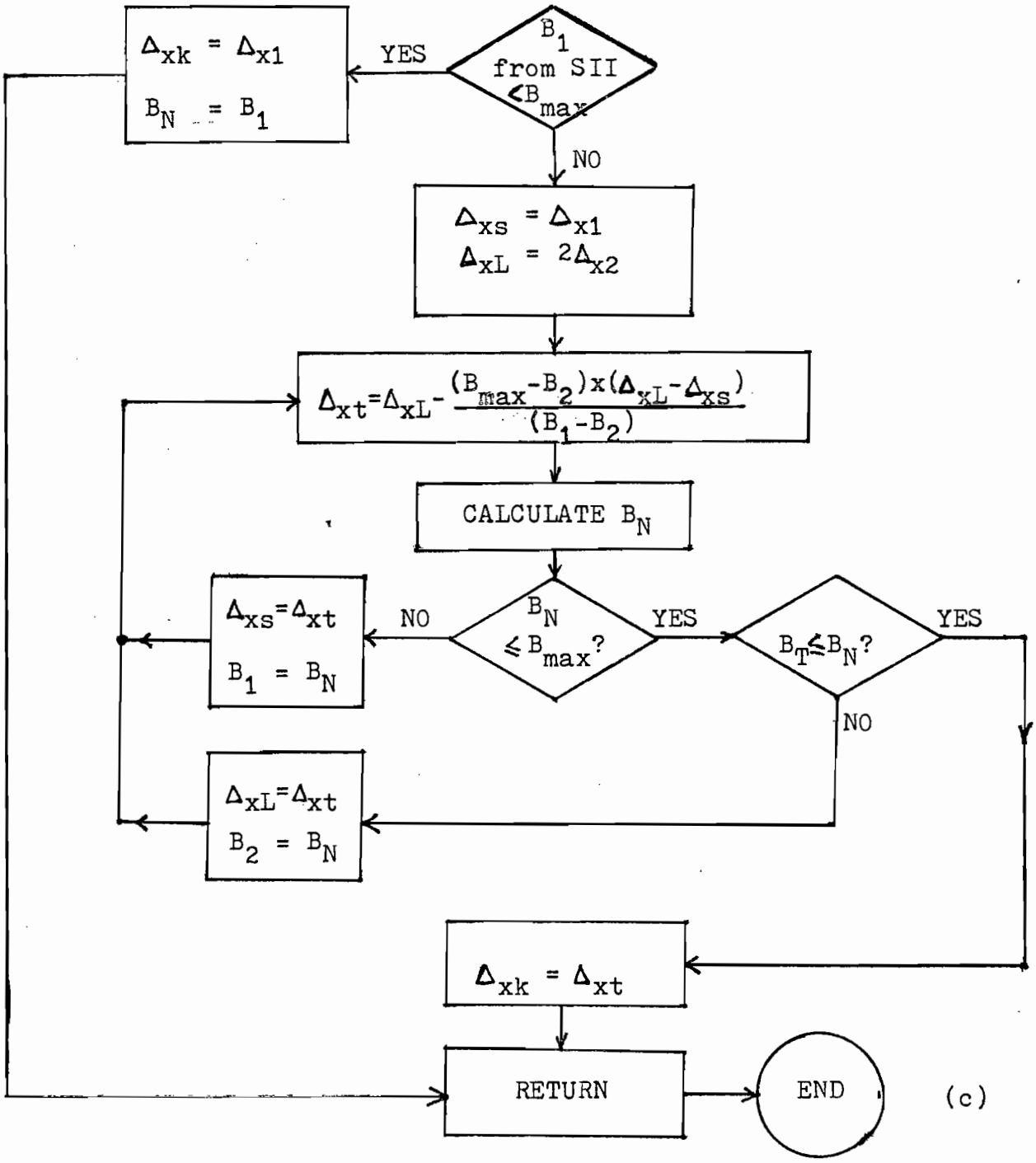


Fig. 4.12 (a) Flow-Chart of Main Program
(b) Flow—Chart of Subroutine SII, Calculate $[\Delta_{x1}, 2\Delta_{x1}]$
(c) Flow-Chart of Subroutine SDELXK, Calculate Δ_{xk}

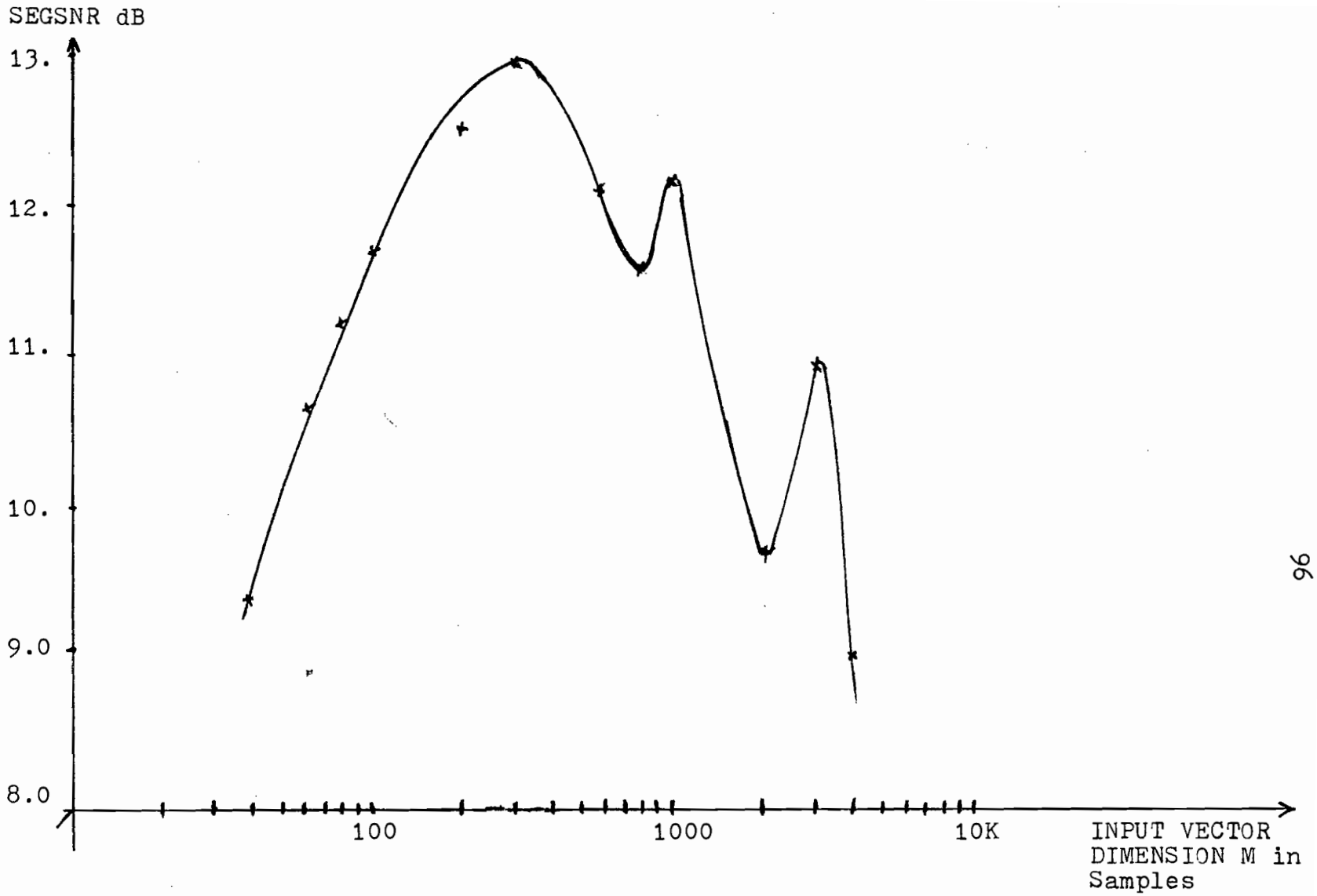


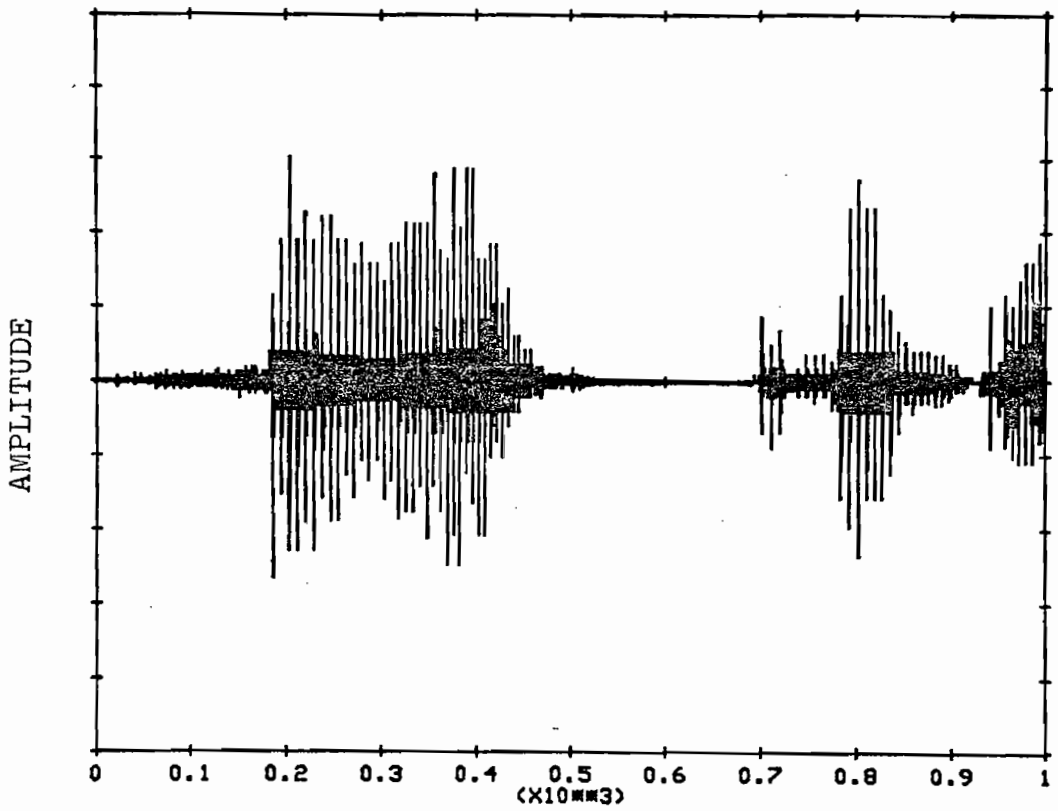
Fig. 4.13 SEGSNR verse VECTOR DIMENSION M

Table 4.9 Block Quantization Coder performance with M=300 samples, N=19, Δ_{xmax}=40, Δ_{xmin}=0.00452, Tr=16 kb/s & Sr=6.5 kSample/s,

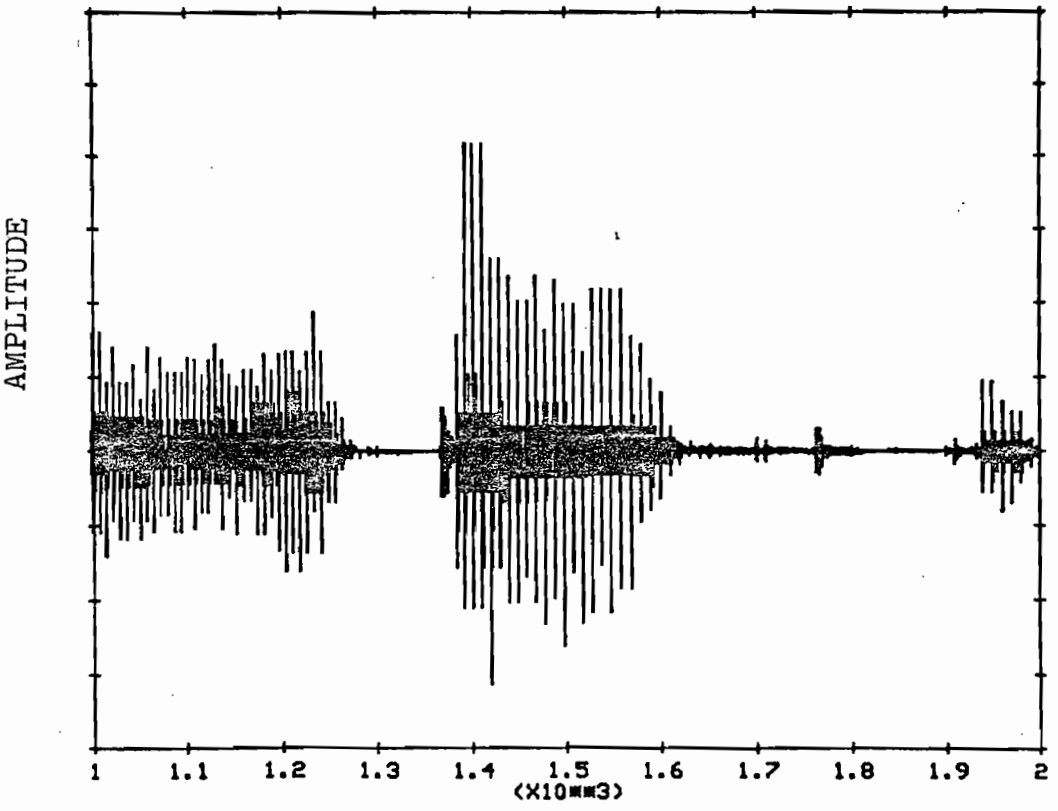
SEGSNR	ENTROPY	$\bar{L}$	EFFICIENCY
dB	Bits	b/sample	
12.9	2.27	2.35	0.966

Amplitude Distribution of the Quantized Outputs.

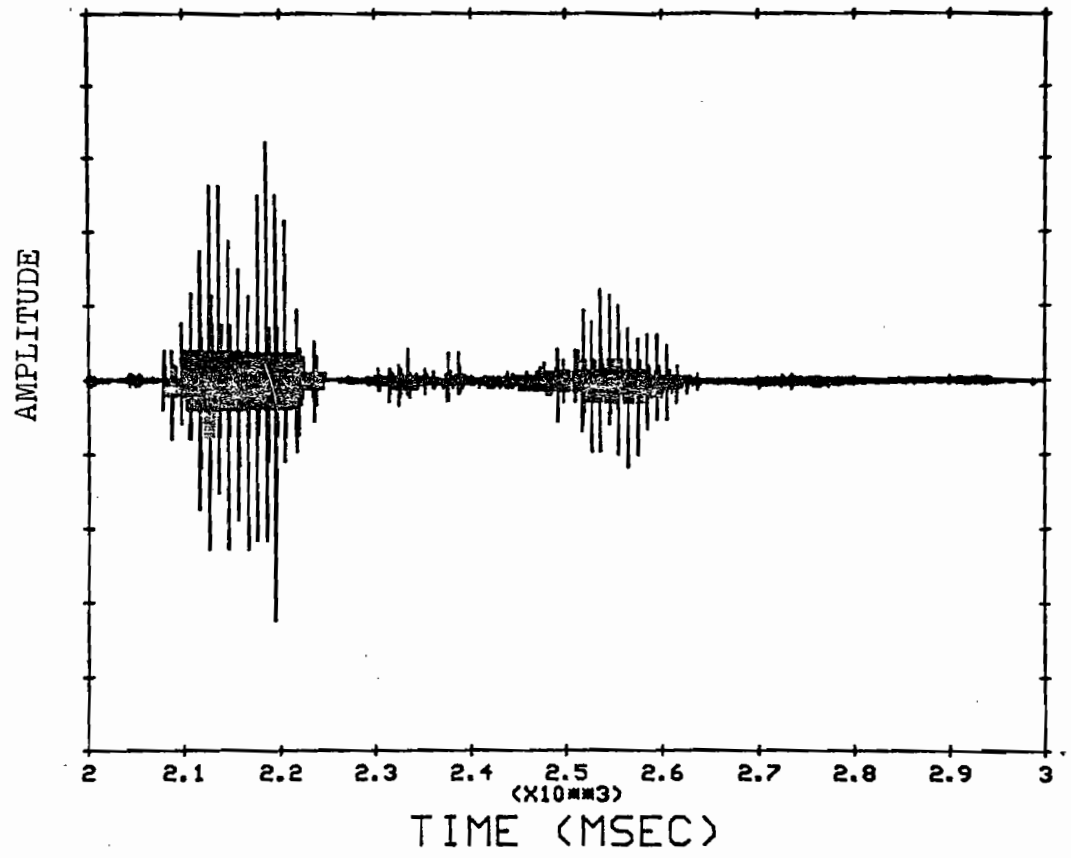
LEVEL NUMBER	DISTRIBUTION	LEVEL NUMBER	DISTRIBUTION
1	0.071%	19	0.13 %
2	0.03 %	18	0.047%
3	0.13 %	17	0.2 %
4	0.24 %	16	0.22 %
5	0.28 %	15	0.3 %
6	0.61 %	14	0.45 %
7	1.5 %	13	1.04 %
8	5.8 %	12	4.2 %
9	22.4 %	11	18.1 %
10	44.1 %		


AMPLITUDE
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(X10**3)
TIME (MSEC)

(a)

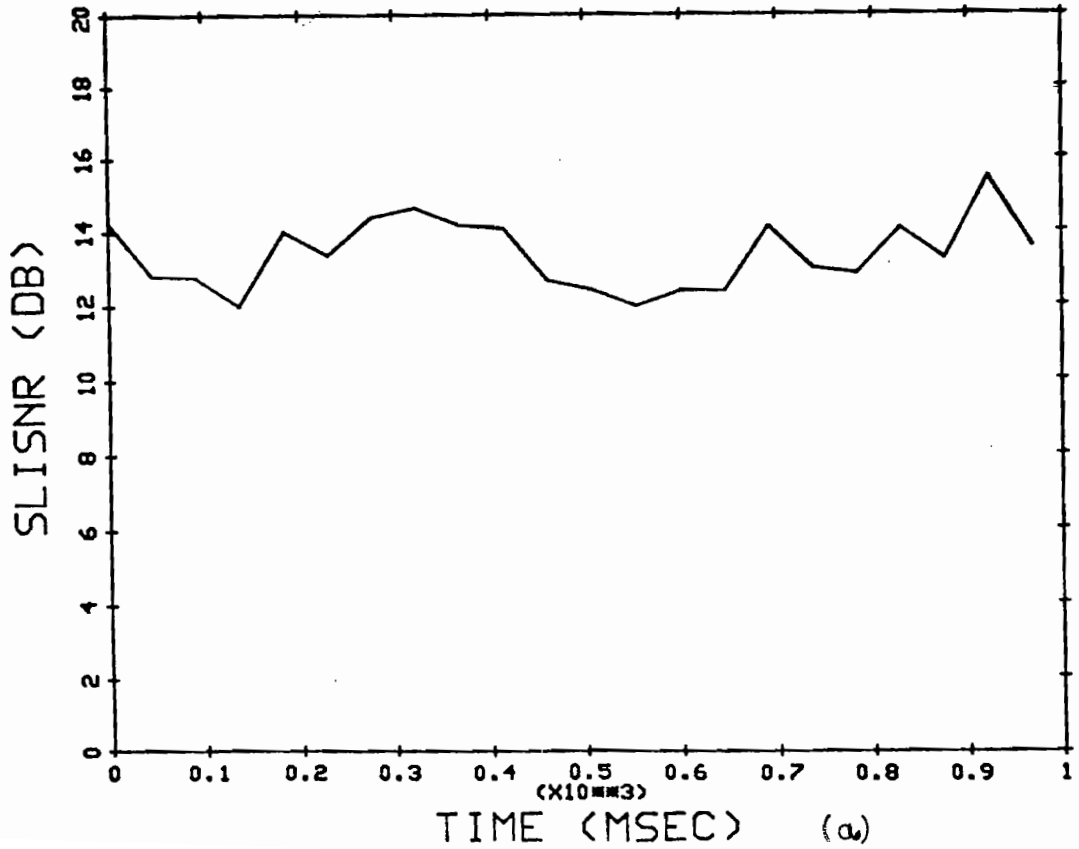

AMPLITUDE
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
(X10**3)
TIME (MSEC)

(b)

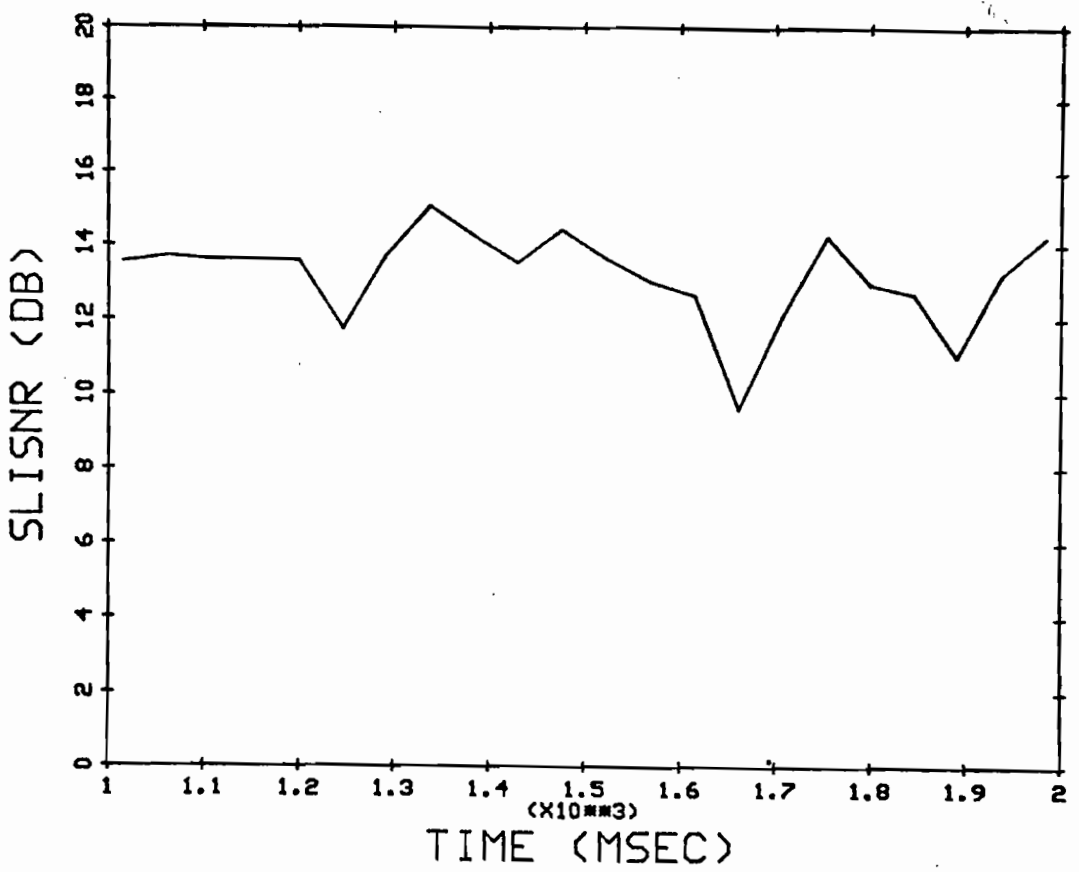


(c)

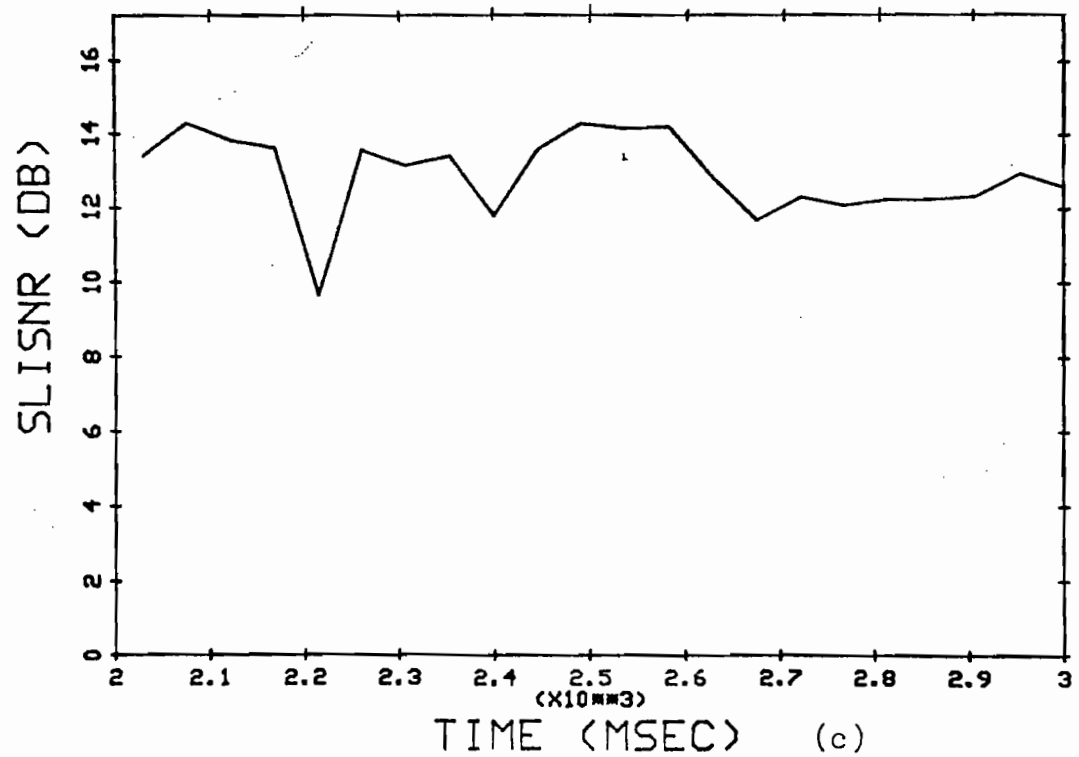
Fig. 4.14 a,b&c Decoded Output at Receiver verse Time with Buffer Management Achieved by Block Quantization



(a)



(b)



(c)

Fig. 4.15 (a),(b) & (c) SLISNR verse Time with Buffer Management Achieved by Using Block Quantization

4.3 Comparisons

In this subsection, we compare the performance of all the proposed coders and the ANQ + FLC coder. For the last coder, we assume the code-word length of each quantization level can be of non-interger. As

$$Tr = 16 \text{ kb/s and} \tag{4.48}$$

$$Sr = 6.5 \text{ kSample/s,} \tag{4.49}$$

therefore

$$N = 2\ Tr/Sr = 2\ 16/6.5 = 5.5 \tag{4.50}$$

Therefore we study the performance of an optimum ANQ + FLC coder with N=5 and 6. For fair comparisons, MT_{min} and MT_{max} for this coder are set equal to 0.0113 and 34.2 respectively same as those in Table 4.2. Other optimum parameters which minimize the quantization noise are tabulated in Table 3.10. The output levels of the quantizer in this table are obtained by using the iteration method proposed by Max (1), and, Lapacian density function with STD equals to 1 is used as the pdf. The multipliers are experimental optimum values with "SRFMT" as input.

Table 4.10 Optimum Parameters for ANQ + FLC Coder

Level Number	N = 5		N = 6	
	Multiplier	Output Level	Multiplier	Output Level
1	1.90425	-2.2031	2.04875	-2.5037
2	1.03325	-0.8206	1.14325	-1.1163
3	0.73775	0	0.84425	-0.2969
4	1.03325	0.8202	0.84425	0.2969
5	1.90425	-2.2031	1.14325	1.1163
6			2.04875	2.5037

The performance of the coders are tabulated in Table 4.11. The data under the column "% of quantizer overload" is obtained by adding the percentage of occurance of the two outer most levels of the quantizer. This table shows that the BLKQNT coder performs best. The BLKQNT coder is insensitive to different types of inputs. This is because the BLKQNT coder can calculate the statistics of the input vector. In our application here, it has an adaptive adaptation rate during the transitions from low energy to high enery and high energy to low energy. Both AECF and AQSS coders perform more or less the same. The AECF coder performs about 0.2-0.3 dB better than the AQSS coder.

With optimum parameters obtained by using "SRFMT" as input, the performance of these two coders degrade for "FA3" and MB3" inputs. This is because the difference in STD between low energy and high energy segements is much larger in "FA3"/"MB3" than in "SRFMT"/"SRFMP". As the decaying rates of these two coders are so small and are not adaptive, therefore, the SNR during the transition from high energy to low energy segments is very low. In additon, the SNR during transition from low energy to high energy is also degraded because the increase in the adaptation rate is limited by either ΔB_1 or $\delta_{\Delta x1}$, the prefixed parameter. On the other hand, BLKQNT does not process these two shortcomings. If the application of the coder is restricted to only one type of inputs, i.e. either difference or direct speech signals, then different optimum parameters can be obtained for different types of inputs. In that case, on the average, BLKQNT coders are only 0.4 dB better than AECF coders and 0.7 dB better than AQSS coders. The ANQ + FLC coder is indeed the worst! In addition to comparatively low segemental SNR, the percentage of quantizer overload is absolutely unacceptable. Table 4.11 shows that, on the average, both AECF and AQSS coders are more than 2dB better than the ANQ + FLC coder. If the application of the coders is restricted to one type of inputs, than the ANQ + FLC coder will give a segmental SNR about 3 dB less than the other two feedback coders.

Table 4.11 Comparisons of Coder Performance

(with the parameters of each individual coder same as in those in table 3.5, 3.8, 3.9 & 3.10 respectively)

CODER	SENTENCE	SEGSNR	ENTROPY	$\overline{L}$	EFFICIENCY	% of Quantizer
		db	Bits b/sample			Overload
AECF with	SRFMT	12.47	2.34	2.44	0.959	0.32%
(i)	SRFMP	12.22	2.34	2.44	0.961	0.36%
MT_k=eq. 4.7a	FA3	11.45	2.34	2.42	0.967	0.06%
	MB3	11.70	2.35	2.42	0.971	0.11%
(ii)	SRFMT	12.49	2.34	2.44	0.962	0.24%
MT_k=eq. 4.7b	SRFMP	12.30	2.35	2.42	0.967	0.29%
	FA3	11.54	2.34	2.42	0.967	0.05%
	MB3	11.78	2.35	2.43	0.967	0.10%
AQSS	SRFMT	12.2	2.32	2.42	0.957	0.28%
	SRFMP	12.1	2.32	2.42	0.959	0.34%
	FA3	11.34	2.31	2.4	0.963	0.07%
	MB3	11.76	2.34	2.42	0.965	0.13%

Table 4.11 Comparisons of Coder Performance, (Cont'd)

CODER	SENTENCE	SEGSNR	ENTROPY	$\bar{L}$	EFFICIENCY	% of Quantizer
		dB	Bits b/sample			Overload
BLKONT	SRFMT	12.9	2.27	2.35	0.966	0.20%
	SRFMP	12.78	2.27	2.35	0.966	0.28%
	FA3	13.3	2.30	2.36	0.975	0.03%
	MB3	13.1	2.31	2.36	0.979	0.05%
ANQ+FLC	SRFMT	8.1	2.05	2.32	0.883	13.4 %
with	SRFMP	8.4	2.05	2.32	0.883	13.5 %
(i) N=5	FA3	8.9	2.02	2.32	0.870	11.2 %
	MB3	8.7	2.00	2.32	0.861	10.6 %
(ii) N=6	SRFMT	9.3	2.24	2.585	0.867	7.1 %
	SRFMP	9.5	2.24	2.585	0.867	7.2 %
	FA3	10.2	2.21	2.585	0.855	5.0 %
	MB3	9.6	2.22	2.585	0.855	5.1 %

CHAPTER 5

CONCLUSIONS

The results presented in the previous chapter show that of the three buffer management methods, namely (i) adaptive expansion and contraction factor, (ii) adaptive quantization step-size and (iii) block quantization, proposed to solve the buffer overflow problem introduced by using variable-length codes (entropy codes), block quantization scheme gives the largest segmental SNR. With transmission rate equals to 16 kb/s and sampling rate equals to 6.5 kSamples/s, block quantization scheme with block dimension of 300 samples gives a segmental SNR about 13 dB for both difference and direct speech inputs. With optimum parameters obtained by using difference speech signals as input, adaptive expansion and contraction factor scheme gives a segmental SNR of 12.5 dB for difference inputs and 11.7 dB for direct inputs. Adaptive quantization step-size scheme performs more or less the same as adaptive expansion and contraction scheme. It gives a segmental SNR of 12.2 dB for difference inputs and 11.6 dB for direct inputs.

At 16 kb/s, feedback systems using variable-length codes plus buffer management strategies proposed in this study perform about 2 dB better than those using fixed-length encoding . If the application is restricted to either difference input or direct input, then using entropy coding + buffer management will be 3 dB better than using fixed-length encoding.

In addition, coders using entropy encoding + buffer management are subjectively much better than coders using fixed-length encoding because the percentage of quantization overload of entropy coders is more than 50 times less than those not using entropy coding.

After all, a few comments have to be made on the trade-off between the complexity of hand-ware implementation and the performance of the proposed coders. Block quantization coders, obviously, are comparatively more complicate to build. On the other hand, the other two types of feedback coders are fairly easy to implement physically. If the application of the coder is restricted to either difference inputs or direct speech input, then it is better to use the feedback coding scheme. This is because a coder with adaptive expansion and contraction factors performs only 0.4 dB worse than a block quantization coder.

REFERENCES

1. L.R. Rabiner and R.W. Schafer, Digital Processing of Speech Signals, Englewood Cliffs, N.J., Prentice Hall, 1978.

2. Bell Telephone Lab, Transmission Systems for Communications, Winston-Salem, North Carolina, pp. 143, 1971.

3. R.G. Gallager, Information Theory and Reliable Communication, Wiley, New York, 1968.

4. J. Max, "Quantizing for Minimum Distortion", IRE Trans. Inform. Theory, Vol. IT-6, pp. 7-12, March 1960.

5. M.D. Paez and T.H. Glisson, "Minimum Mean-Square-Error Quantization in Speech PCM and DPCM Systems", IEEE Trans. Commun., Vol. COM-20, pp. 225-230, April 1972.

6. A. Gersho, "Principles of Quantization", IEEE Communications Society Magazine, Vol. 15, no. 5, pp. 16-29, September 1977.

7. P. Noll, "A Comparative Study of Various Schemes for Speech Encoding", Bell System Tech. J., Vol. 54, No. 9, pp. 1597-1614, November 1975.

8. R.W. Stroh, "Optimum and Adaptive Differential Pulse Code Modulation", Ph.D. Thesis, Polytechnic Institute of Brooklyn, 1970.

9. P. Noll, "Adaptive Quantizing in Speech Coding Systems", Proc. 1974 Zurich Seminar on Digital Communications, Zurich, March 1974.

10. N.S. Jayant, "Adaptive Quantization with a One Word Memory", Bell System Tech. J., pp. 1119-1144, September 1973.

11. P. Cummiskey, N.S. Jayant, and J.L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech", Bell System Tech. J., Vol. 52, No. 7, pp. 1105-1118, September 1973.

12. D.J. Goodman, "Theory of an Adaptive Quantize", IEEE Trans. Commun., Vol. COM-22, pp. 1037-1045, August 1974.

13. D. Mitra, "Mathematical Analysis of an Adaptive Quantizer", Bell System Tech. J., Vol. 53, No. 5, pp. 867-899, June 1974.

14. T. Berger, Rate Distortion Theory, Englewood Cliffs, NJ : Prentice-Hall, 1971.

15. D.A. Huffman , "A Method for the Construction of Minimum-Redundance Codes", Proc. IRE, Vol. 40, pp. 1098-1101, September 1952.

16. K. Virupaksha, "Entropy-Coded Adaptive Differential Pulse-Code Modulation (DPCM) for Speech", IEEE Trans. in Commun., Vol. COM-22, No. 6, June 1974.

17. S.U.H. Qureshi, and G.D. Forney, "A 9.6/16 kb/s Speech Digitizer", 1975 IEEE Int. Conf. on Commun., Vol. 11, pp. 30-31-30-36. June 16-18, 1975.

18. J. Makhoul, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech", IEEE Trans. on ASSP, Vol. ASSP-27, No. 1, February 1979.

19. D.L. Cohn and J.L. Melsa, "The Residual Encoder - An Improved ADPCM System for Speech Digitization, "IEEE Trans. on Commun., Vol. Com-23, No. 9, September 1975.

20. P.T. Brady, "Effect of Transmission Delay on Conversational Behaviour on Echo-free Telephone Circuits", Bell System Tech. J. Vol. 50, pp. 115-134, January 1971.

21. S.K. Goyal, and J.B. O'Neal, "Entropy Coded Differential Pulse-Coded Modulation Systems for Television", in 1974 Nat. Telecommun. Conf. Rec., pp. 72-76.

22. H. Gish and J.N. Pierce, "Asymptotically Efficient Quantizing", IEEE Trans. Inform. Theory, Vol. IT-14, pp. 676-683, September 1968.

23. P. Mermelstein, "Evaluation of a Segmental SNR Measured as an Indicator of the Quality of ADPCM Coded Speech", presented at Joint Meeting of the Acoustical Society of America and Acoustical Society of Japan, Honolulu, Hawaii, November 1975.

24. P. Castellino, G. Madena, L. Nebbia and C. Scngliala, "Bit Rate Reduction by Automatic Adaptation of Quantizer Step-size in DPCM Systems", in Proc. 1974 Int. Zurich Seminar Digital Communications, pp. B6(1) - B6(6).

25. D.L. Cohn, "The Relationship Between an Adaptive Quantizer and a Variance Esimator", IEEE Trans. Inform. Theory, Vol. IT-21, November 1975.

26. D.A. Wismer, R. Chattergy, Introduction to Nonlinear Optimization, North Holland, Amsterdam, The Netherlands, 1979.

APPENDICES PROGRAM LISTINGS

This appendix lists the FORTRAN programs used in the simulations of the proposed coders.

A. Adaptive Expansion-Contraction Factors Module

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       AECF
C
C PURPOSE:
C
C       An interactive program to simulate the coder which performs
C       (i)   adaptive uniform quantization,
C       (ii)  entropy coding and
C       (iii) buffer management achieved by changing the
C             multilpiers associated with each quantization level
C
C MEDIA:
C
C       ING.AECF]AECF.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       $Run ING.AECF]AECF
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       None
C
C IMPORTANT VARIABLES:
C
C       A               R*4     A constant for the equation to
C                               calculate the multipliers associated
C                               with each quantization level
C       ACWL            R*4     Average code-word length
C       BFOC            R*4     An array of buffer occupancy with each
C                               element equal to the buffer contents just
C                               before the next comming sampling instant
```

```
C       BMAX            R*4     Maximum value of the adaptive buffer
C                               management variable B
C       BMIN            R*4     Minimum value of the adaptive buffer
C                               management variable B
C       BPSA            R*4     Number of bits taken out of the buffer per
C                               sampling period
C       CL              I*4     An array of code-word length
C       DELB1           R*4     Delta B for increasing buffer occupancy
C                               and BFOC greater than 0
C       DELB2           R*4     Delta B for increasing buffer occupancy
C                               and BFOC less than 0
C       DELB3           R*4     Delta B for decreasing buffer occupancy
C                               and BFOC greater than 0
C       DELB4           R*4     Delta B for decreasing buffer occupancy
C                               and BFOC less than 0
C       DELX            R*4     Quantization step-size
C       EFF             R*4     Coding efficiency
C       ETP             R*4     Quantizer output entropy
C       FILE_SPC        A*50    File specification
C       IL              I*4     Lower quantizer level
C       INPUTT          A*2     Input type, either direct(D) or difference
C                               (DF) input
C       IU              I*4     Upper quantizer level
C       M               R*4     An array of multipliers associated with
C                               each quantization level
C       MTMAX           R*4     Maximum output of the multiplier logic unit
C       MTMIN           R*4     Minimum output of the multiplier logic unit
C       MT0             R*4     Initial output of the multiplier logic unit
C       NBFR            I*4     Negative buffer range with "0" reference
C                               point set at any position within the buffer
C       NBLK            I*4     Total number of blocks of input data with
C                               recordsize of each block equal to NSAMPR
C       NLEV            I*4     Number of quantization levels
C       NS              I*4     Total number of input samples
C       NSAMPR          I*4     Number of storage units/samples per record
C                               (machine dependent)
C       NSEG            I*4     Number of segments or blocks
C       NSL             I*4     Total number of samples lost due to buffer
C                               overflow
C       PBFR            I*4     Positive buffer range with "0" reference
C                               point set at any position within the buffer
C       PROB            R*4     An array of probability of occurence of
C                               each quantization level
C       QSLTOC          R*4     An array of the number of occurence of
C                               each quantization level
C       RUNIT           I*4     Read unit
C       SEGL            I*4     Number of samples per segment or block
C       SEGSNR          R*4     Segmental SNR
C       SLISNR          R*4     Sliding SNR
C       SR              R*4     Sampling rate in kSample/s
C       TR              R*4     Channel transmission rate in kb/s
C       WUNIT           I*4     Write unit
C       X               R*4     An array of input samples
C       XRANGE          R*4     The quantization range of the quantizer
C                               without clipping
C       YQ              R*4     An array of quantizer output levels
C       YQR             R*4     An array of decoded quantizer outputs at
C                               receiver
C       VMAX            R*4     Maximum allowable input amplitude
C
```

```
C COMMON BLOCKS:
C
C       See [NG.AECF]FBCODERCB.FOR
C
C ROUTINES REQUIRED:
C
C       AUQ_AECF_EC       In [NG.AECF] subdirectory
C       BFMAN             In [NG.UTLPGM] subdirectory
C       CMPQNT            <---------- " " ---------->
C       ENTROPY           <---------- " " ---------->
C       QUNAT             <---------- " " ---------->
C       SLI_SNR           <---------- " " ---------->
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Medium and speed depends on the length of the speech
C       utterance under processed
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Nov. , 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------


        PROGRAM AECF

C---
C   Include feedback coder common block
C---

        INCLUDE '[NG.COMMONB]FBCODERCB.FOR'

C---
C   Specify the type of IX, IVQR, RUNIT, WUNIT, INPUTT & FILE_SPC
```

```
C---
      INTEGER*2
     1          IX,
     1          IYQR,
     1          RUNIT,
     1          WUNIT
      CHARACTER
     1          INPUTT*2,
     1          FILE_SPC*50
C---
C   Specify the dimension of IX, IYQR, QSLTOC, X & YQR
C---
      DIMENSION
     1          IX(NSAMPR),
     1          IYQR(NSAMPR),
     1          QSLTOC(MAXLEV),
     1          X(MAXBLK*NSAMPR),
     1          YQR(MAXBLK*NSAMPR)
C---
C   Initialize A, CL, DELB1, DELB2, DELB3, DELB4, DELX, MTMAX,
C   MTMIN, MT0, NBFR, NLEV, PBFR, RUNIT, VMAX & WUNIT
C---
      DATA
     1       A/.982/,
     1       CL
     1         /
     1          18,
     1          16,
     1          14,
     1          12,
     1          10,
     1          8,
     1          6,
     1          4,
     1          2,
     1          1,
     1          3,
     1          5,
     1          7,
     1          9,
     1          11,
     1          13,
     1          15,
     1          17,
     1          18,
     1          13*0
     1         /,
     1       DELB1/.0002/,
     1       DELB2/.0001/,
     1       DELB3/-.0001/,
     1       DELB4/-.0001/,
     1       DELX/1.1706/,
     1       MTMAX/34.17/,
     1       MTMIN/.011300301/,
     1       MT0/.011300301/,
```

```
     1       NBFR/-401/,
     1       NLEV/19/,
     1       PBFR/401/,
     1       RUNIT/5/,
     1       VMAX/20./,
     1       WUNIT/6/

C---
C   Set up the equivalence between (BMAX,ZMAX), (BMIN,ZMIN),
C   (DELB1,DEL21), (DELB2,DEL22), (DELB3,DEL23) & (DELB4,DEL24)
C---

      EQUIVALENCE
     1            (BMAX,ZMAX),
     1            (BMIN,ZMIN),
     1            (DELB1,DEL21),
     1            (DELB2,DEL22),
     1            (DELB3,DEL23),
     1            (DELB4,DEL24),
     1            (PROB,QSLTOC)

C---
C  Read in channel transmission rate
C---

      ERROR=1
1     WRITE(WUNIT,1000)
      READ(RUNIT,*,ERR=10000) TR

C---
C   Read in sampling rate
C---

      ERROR=17
2     WRITE(WUNIT,1100)
      READ(RUNIT,*,ERR=10000) SR


C---
C   Read in input file specification
C---

      ERROR=18
3     WRITE(WUNIT,1200)
      READ(RUNIT,1300,ERR=10000) FILE_SPC

C---
C   Read in input type
C---
      ERROR=19
4     WRITE(WUNIT,1400)
      READ(RUNIT,1300,ERR=10000) INPUTT

C---
C   Open input file
C---

      ERROR=2
      IDUN=20
      OPEN(UNIT=IDUN,
```

```
     1     NAME=FILE_SPC,
     1     TYPE='OLD',
     1     ACCESS='DIRECT',
     1     ASSOCIATEVARIABLE=IDINDEX,
     1     ERR=10000)

C---
C   Read in input data and calculate the total number of input
C   blocks
C   * Note the file header is not read
C---

      ERROR=3
      NBLK=0
      DO 100 J=2,MAXBLK
      READ(IDUN'J,END=110,ERR=110) (IX(I),I=1,NSAMPR)
      NBLK=NBLK+1
      DO 100 I=1,NSAMPR
      X((J-2)*NSAMPR+I)=FLOAT(IX(I))
100   CONTINUE
      ERROR=6
      GO TO 10000
110   IF(NBLK.LT.1) GO TO 10000
      ERROR=5
      CLOSE(UNIT=IDUN,
     1     DISP='SAVE',
     1     ERR=10000)

C---
C   Calculate the total number of input samples
C---

      NS=NBLK*NSAMPR

C---
C   Calculate the total number of bits taken out of the buffer
C   per sampling period
C---

      BPSA=TR/SR

C---
C  Rescale the input samples
C---
      IF(INPUTT.EQ.'D') THEN
      SCALE=3./2816.4
      ELSE
      SCALE=1./315.54
      END IF
      DO 120 I=1,NS
      X(I)=X(I)*SCALE
120   CONTINUE

C---
C  Calculate NSEG
C---

      NSEG=(NS/SEGL)-2

C---
```

```
C   Calculate IL & IU
C---

        IL=(NLEV+1)/2
        IU=(NLEV/2)+1

C---
C   Calculate the fixed quantizer output level with quantization
C   step_size equal to DELX
C---

        XRANGE=DELX*FLOAT(NLEV)/2.0
        CALL CMPQNT(NLEV,XRANGE,Y0)

C---
C   Simulate the input datas
C---

        CALL AUQ_AECF_EC(NS,X,YQR)

C---
C   Read in output file name from user
C---

        ERROR=20
5       WRITE(WUNIT,1500)
        READ(RUNIT,1300,ERR=10000) FILE_SPC

C---
C   Store decoded output at receiver in file
C---

        ERROR=7
        IOPUN=90
        IOPINDEX=1
        OPEN(UNIT=IOPUN,
        1    NAME=FILE_SPC,
        1    TYPE='NEW',
        1    ACCESS='DIRECT',
        1    ASSOCIATEVARIABLE=IOPINDEX,
        1    RECORDSIZE=NSAMPR,
        1    ERR=10000)
        ERROR=4
        DO 140 J=2,NBLK+1
        DO 130 I=1,NSAMPR
        IYQR(I)=IFIX(YQR((J-2)*NSAMPR+I)/SCALE)
130     CONTINUE
        WRITE(IOPUN'J,ERR=10000) (IYQR(I),I=1,NSAMPR)
140     CONTINUE
        ERROR=5
        CLOSE(UNIT=IOPUN,
        1     DISP='SAVE',
        1     ERR=10000)
C---
C   Calculate the sliding SNR and segmental SNR
C---

        CALL SLI_SNR(X,YQR,NSEG,SEGL,NLEV,NS,SLISNR,SEGSNR)

C---
```

```
C   Store the sliding SNR into file
C---

        ERROR=8
        ISNRUN=91
        ISNRINDEX=1
        OPEN(UNIT=ISNRUN,
        1     NAME='SLISNR.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=ISNRINDEX,
        1     RECORDSIZE=1,
        1     ERR=10000)
        ERROR=9
        DO 150 I=1,NSEG
        WRITE(ISNRUN'ISNRINDEX,ERR=10000) SLISNR(I)
150     CONTINUE
        ERROR=10
        CLOSE(UNIT=ISNRUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   Store buffer occupancy in file
C---

        ERROR=11
        IBOUN=92
        IBOINDEX=1
        OPEN(UNIT=IBOUN,
        1     NAME='BFOC.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=IBOINDEX,
        1     RECORDSIZE=1,
        1     ERR=10000)
        ERROR=12
        DO 160 I=1,NS
        WRITE(IBOUN'IBOINDEX,ERR=10000) BFOC(I)
160     CONTINUE
        ERROR=13
        CLOSE(UNIT=IBOUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   change the number of occurence of each quantization level to
C   probability of occurence
C---

        DO 170 I=1,NLEV
        PROB(I)=@SLTOC(I)/NS
170     CONTINUE

C---
C   Calculate the quantizer output entropy
C---

        CALL ENTROPY(NLEV,PROB,ETP)
```

```
C---
C   Calculate the average code_word length
C---

        ACWL=0.
        DO 180 I=1,NLEV
        ACWL=ACWL+PROB(I)*CL(I)
180     CONTINUE

C---
C   Calculate the coding efficiency
C---

        EFF=ETP/ACWL

C---
C   Store the number of samples lost due to buffer overflow,
C   entropy, average code-word length, coding eddiciency, segmental
C   SNR and the probability of occurence of each quantizer level
C   in file
C--

        ERROR=14
        ISTTUN=93
        ISTTINDEX=1
        OPEN(UNIT=ISTTUN,
        1     NAME='STTS.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=ISTTINDEX,
        1     FORM='FORMATTED',
        1     RECORDSIZE=512,
        1     ERR=10000)
        ERROR=15
        WRITE(ISTTUN'ISTTINDEX,1600) NSL,
        1                            ETP,
        1                            ACWL,
        1                            EFF,
        1                            SEGSNR,
        1                            (PROB(I),I=1,NLEV)
        ERROR=16
        CLOSE(UNIT=ISTTUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   Program stop
C---

        STOP

C---
C   Prompt the error message
C---

10000   IF(ERROR.GT.16.OR.ERROR.EQ.1) THEN
        NERR=1
        ELSE
        NERR=ERROR
        END IF
```

```
        WRITE(WUNIT,*) '%ERROR >>> '//ERROR_MESSAGE(NERR)

C---
C   Prompt the enter message again
C---

        IF(ERROR.EQ.1) GO TO 1
        IF(ERROR.EQ.2) GO TO 3
        IF(ERROR.EQ.17) GO TO 2
        IF(ERROR.EQ.18) GO TO 3
        IF(ERROR.EQ.19) GO TO 4
        IF(ERROR.EQ.20) GO TO 5
        STOP

C---
C   Formats
C---

1000    FORMAT('$ ENTER CHANNEL TRANSMISSION RATE  (R*4) >>> ')
1100    FORMAT('$ ENTER SAMPLING RATE  (R*4) >>> ')
1200    FORMAT('$ ENTER INPUT FILE SPECIFICATION ( < A50 ) >>> ')
1300    FORMAT(A)
1400    FORMAT('$ ENTER INPUT TYPE (Type D or DF) >>> ')
1500    FORMAT('$ ENTER OUTPUT FILE SPECIFICATION ( < A50 ) >>> ')
1600    FORMAT(I7,63F15.7)
        END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       AUQ_AECF_EC
C
C PURPOSE:
C
C       To simulate the coder with:-
C          (i)    a uniform quantizer
C          (ii)   adaptive expansion and contraction factors
C                 associated with each quantization level for the
C                 tracking of the the input power level and for
C                 the elimination of the buffer overflow problem
C          (iii) an entropy encoder
C
C MEDIA:
C
C       [NG.AECF]AUQAECFEC.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       TO:-    NS      I*4     Total number of input samples
C               X       R*4     An array of input speech samples
C
C       From:-  YQR     R*4     An array of decoded quantizer outputs
C                               at the receiver
C
C IMPORTANT VARIABLES:
C
C       A       R*4     A constant for the equation to calculate the
C                       adaptive multipliers associated with each
C                       quantization level
C       B       R*4     The adaptive buffer management variable
C       BFOC    R*4     An array of buffer occupancy just before the
C                       next comming sampling instant
C       BFOCL   R*4     Buffer occupancy just before the present
C                       sampling instant
C       BMAX    R*4     Maximum value of the adaptive buffer management
C                       variable B
C       BMIN    R*4     Minimum value of the adatpive buffer management
C                       variable B
```

```
C       BPSA    R*4     Number of bits taken out of the buffer per
C                       sampling period
C       CL      I*4     An array of code-word length
C       DELB1   R*4     Delta B for increasing buffer occupancy and
C                       BFOC greater than 0
C       DELB2   R*4     Delta B for increasing buffer occupancy and
C                       BFOC less than 0
C       DELB3   R*4     Delta B for decreasing buffer occupancy and
C                       BFOC greater than 0
C       DELB4   R*4     Delta B for decreasing buffer occupancy and
C                       BFOC less than 0
C       IL      I*4     Lower quantizer level
C       IU      I*4     Upper quantizer level
C       K       I*4     The kth decoded sample at the receiver
C       L       I*4     The quantization level number that the present
C                       sample falls
C       M       R*4     An array of multipliers associated with each
C                       quantization level
C       MT      R*4     The output of the multiplier logic unit
C       MTMAX   R*4     Maximum output of the multiplier logic unit
C       MTMIN   R*4     Minimum output of the multiplier logic unit
C       MT0     R*4     The initial output of the multiplier logic unit
C       NBFR    I*4     Negative buffer range with "0" reference point
C                       set at any position within the buffer
C       NLEV    I*4     Total number of quantization levels
C       NS      I*4     Total number of input speech samples
C       PBFR    I*4     Positive buffer range with "0" reference point
C                       set at any position within the buffer
C       QINPUT  R*4     Quantizer input
C       QSLTOC  R*4     An array of the number of occurance of each
C                       quantization level
C       QFFT    I*4     Buffer overflow flat
C       UFFT    I*4     Buffer underflow flat
C       VMAX    R*4     Maximum allowable input amplitude
C       YQ      R*4     An array of quantizer output levels
C       YQR     R*4     An array of decoded quantizer output at receiver
C
C COMMON BLOCKS:
C
C       See [NG.COMMONB]FBCODERCB.FOR
C
C ROUTINES REQUIRED:
C
C       BFMAN           In [NG.UTLPGM] subdirectory
C       QUANT           <---------- " " ---------->
C
C SIDE EFFECTS:
C
C       (i)   return the array of number of occurance of each quantizer
C             level to calling program through common block PRMBLK
C       (ii)  return the array of buffer occupancy BFOC to calling
C             program through common block BFBLK
C       (iii) return the number of samples lost due to buffer overflow
C             to calling program through common block PRMBLK
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
```

```
C
C       Small and speed depends on the length of input speech
C       utterance(s)
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Nov. ,1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       Not yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------

        SUBROUTINE AUQ_AECF_EC(NS,X,YQR)

C---
C   Include the coder commer block
C---

        INCLUDE '[NG.COMMONB]FBCODERCB.FOR'

C---
C   Specify the dimension of QSLTQC, X and YQR
C---

        DIMENSION
        1           QSLTQC(MAXLEV),
        1           X(NS),
        1           YQR(NS)

C---
C   Specify the type of QFFT and UFFT
C---

        INTEGER
        1       QFFT,
        1       UFFT
```

```
C---
C   Set up the equivalencce between (B,Z), (BAMX,ZMAX),
C   (BMIN,ZMIN), (DELB1,DELZ1), (DELB2,DELZ2), (DELB3,DELZ3),
C   (DELB4,DELZ4) & (QSLTOC,PROB)
C---

      EQUIVALENCE
     1            (B,Z),
     1            (BMAX,ZMAX),
     1            (BMIN,ZMIN),
     1            (DELB1,DELZ1),
     1            (DELB2,DELZ2),
     1            (DELB3,DELZ3),
     1            (DELB4,DELZ4),
     1            (QSLTOC,PROB)

C---
C   Initialize NSL and MT
C---

      NSL=0
      MT=MT0

C---
C   Calculate BMIN and BMAX
C---

      BMIN=(1-A)/4
      BMAX=1/(A**2)-A

C---
C   set initial value of B equal to BMIN
C---

      B=BMIN

C---
C   Calculate the initial values for the multipliers associated
C   with each quantization level
C---

      DO 100 I=1,IL
      M(IL-I+1)=(A+B*(I-1)**2)
      M(IU+I-1)=M(IL-I+1)
100   CONTINUE

C---
C   Initialize K and BFOCL
C---

      K=1
      BFOCL=0

C---
C   Start simulate the input
C---

      DO 400 I=1,NS
```

```
C---
C   Determine the input to the quantizer
C---

            IF(ABS(X(I)).LE.VMAX) THEN
            QINPUT=X(I)/MT
            ELSE
            SIGNX=1.
            IF(X(I).LT.0) SIGNX=-1.
            QNTPUT=VMAX*SIGNX/MT
            END IF

C---
C   Get the quantization level number for the input
C---

            CALL QUANT(QINPUT,L,YQ,NLEV)

C---
C   Increase the number of occurance of the corresponding
C   quantization level by 1
C---

            QSLTOC(L)=QSLTOC(L)+1

C---
C   Update the adaptive buffer management variable B and the buffer
C   occupancy
C---

            CALL BFMAN(BFOCL,BPSA,CL(L),DELB1,DELB2,DELB3,DELB4,
        1               PBFR,NBFR,BMAX,BMIN,B,BFOC(I),OFFT,UFFT)

C---
C   Check for buffer overflow
C---

            IF(OFFT.EQ.1) THEN

C---
C   In case of buffer overflow, then
C       (i)   increase the number of samples lost due to buffer
C             overflow by 1 and
C       (ii)  there is no decoded output at receiver
C---

            NSL=NSL+1
            ELSE

C---
C   In case buffer is not currently overflown, then
C       (i)    determine decoded quantizer output at receiver and
C       (ii)   increase the number of samples received by 1
C---

            YQR(K)=YQ(L)*MT
            K=K+1
            END IF

C---
```

```
C   Update the multipliers associated with each quantization level
C---

            M(L)=(A+B*(L-IL)**2)

C---
C   Update the output of the multiplier logic unit
C---

            MT=MT*M(L)
            IF(MT.LE.MTMIN) MT=MTMIN
            IF(MT.GE.MTMAX) MT=MTMAX

C---
C   Set BFOCL equal to BFOC(I)
C---

            BFOCL=BFOC(I)
400      CONTINUE

C---
C   Return to main program
C---

         RETURN
         END
```

B. Adaptive Quantization Step-Size Module

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       AQSS
C
C PURPOSE:
C
C       An interactive program to simulate the coder which performs
C       (i)   adaptive uniform quantization,
C       (ii)  entropy encoding and
C       (iii) buffer management achieved by changing the quantization
C             step-size.
C
C MEDIA:
C
C       [NG.AQSS]AQSS.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       $RUN [NG.AQSS]AQSS.FOR
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       None
C
C IMPORTANT VARIABLES:
C
C       ALPHA           R*4     A constant for the equation to calculate
C                               the estimated standard deviation (STD)
C                               of the input
C       ACWL            R*4     Average code-word length
C       BFOC            R*4     An array of buffer occupancy with each
C                               element equal to the buffer contents
C                               just before the next comming sampling
C                               instant
C       DELXMAX         R*4     Maximum value of the adaptive buffer
C                               management variable DELX
C       DELXMIN         R*4     Minimum value of the adaptive buffer
C                               management variable DELX
C       BPSA            R*4     Number of bits taken out of the buffer per
C                               sampling interval
C       CL              I*4     An array of code-word length
```

```
C       DELX            R*4     Quantization step-size
C       DELDELX1        R*4     Delta DELX for increasing buffer occupancy
C                               and BFOC greater than 0
C       DELDELX2        R*4     Delta DELX for increasing buffer occupancy
C                               and BFOC less than 0
C       DELDELX3        R*4     Delta DELX for decreasing buffer occupancy
C                               and BFOC greater than 0
C       DELDELX4        R*4     Delta DELX for decreasing buffer occupancy
C                               and BFOC less than 0
C       EFF             R*4     Coding efficiency
C       ETP             R*4     Quantizer output entropy
C       ESTD0           R*4     Initial output of the STD estimator
C       FILE_SPC        A*50    File specification
C       IL              I*4     Lower quantizer level
C       INPUTT          A*2     Input type, either direct(D) or difference
C                               (DF) input
C       IU              I*4     Upper quantizer level
C       NBFR            I*4     Negative buffer range with "0" reference
C                               point set at any position within the
C                               buffer
C       NBLK            I*4     Total number of blocks of input data with
C                               recordsize of each block equal to NSAMPR
C       NLEV            I*4     Number of quantization levels
C       NS              I*4     Total number of input samples
C       NSAMPR          I*4     Number of storage units/samples per record
C                               (machine dependent)
C       NSEG            I*4     Number of segments or blocks
C       NSL             I*4     Total number of samples lost due to buffer
C                               overflow
C       PBFR            I*4     Positive buffer range with "0" reference
C                               point set at any position within the
C                               buffer
C       PROB            R*4     An array of probability of occurence of
C                               each quantization levels
C       QSLTOC          R*4     An array of the number of occurence of
C                               each quantization level
C       RUNIT           I*4     Read unit
C       SEGL            I*4     Number of samples per segment or block
C       SEGSNR          R*4     Segmental SNR
C       SLISNR          R*4     Sliding SNR
C       SR              R*4     Sampling rate in kSample/s
C       STDMAX          R*4     Maximum output of the STD estimator
C       STDMIN          R*4     Minimum output of the STD estimator
C       TR              R*4     Channel transmission rate in kb/s
C       WUNIT           I*4     Write unit
C       X               R*4     An array of input samples
C
C
C COMMON BLOCKS:
C
C       See [NG.COMMONB]FBCODERC8.FOR
C
C ROUTINES REQUIRED:
C
C       AUG_ARSS_EC     In [NG.ARSS] subdirectory
C       BFMAN           In [NG.UTLPGM] subdirectory
C       CMPQNT          <---------- " " ---------->
C       ENTROPY         <---------- " " ---------->
C       QUANT           <---------- " " ---------->
C       SLI_SNR         <---------- " " ---------->
```

```
C
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Medium and speed depends on the length of the speech
C       utterance under processed
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Nov. , 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------
```

```
        PROGRAM AQSS

C---
C   Include feedback coder common block
C---

        INCLUDE 'DNG.COMMONBJFBCODERCB.FOR'

C---
C   Set up the equivalence between (ALPHA,A), (DELDELX1,DELZ1),
C   (DELDELX2,DELZ2), (DELDELX3,DELZ3), (DELDELX4,DELZ4),
C   (DELXMAX,ZMAX), (DELXMIN,ZMIN), (ESTD0,MT0), (QSLTOC,PROB),
C   (STDMAX,MTMAX) & (STDMIN,MTMIN)
C---

        EQUIVALENCE
        1           (ALPHA,A),
        1           (DELDELX1,DELZ1),
        1           (DELDELX2,DELZ2),
        1           (DELDELX3,DELZ3),
        1           (DELDELX4,DELZ4),
```

```
     1             (DELXMAX,ZMAX),
     1             (DELXMIN,ZMIN),
     1             (ESTD0,MT0),
     1             (QSLTOC,PROB),
     1             (STDMAX,MTMAX),
     1             (STDMIN,MTMIN)

C---
C   Specify the type of IX, IYQR, RUNIT, WUNIT, FILE_SPC &
C   INPUTT
C---

      INTEGER*2
     1         IX,
     1         IYQR,
     1         RUNIT,
     1         WUNIT
      CHARACTER
     1         FILE_SPC*50,
     1         INPUTT

C---
C   Specify the dimension of IX, IYQR, QSLTOC, X & YQR
C---

      DIMENSION
     1          IX(NSAMPR),
     1          IYQR(NSAMPR),
     1          QSLTOC(MAXLEV),
     1          X(NSAMPR*MAXBLK),
     1          YQR(NSAMPR*MAXBLK)

C---
C   Initialize ALPHA, ACWL, CL, DELDELX1, DELDELX2, DELDELX3,
C   DELDELX4, DELX, DELXMAX, DELXMIN, EFF, ESTD0, NBFR, NLEV,
C   PBFR, RUNIT, SLISNR, STDMAX, STDMIN, VMAX, WUNIT & YQ
C---

      DATA
     1    ALPHA/.96/,
     1    ACWL/0./,
     1    CL
     1      /
     1        18,
     1        16,
     1        14,
     1        12,
     1        10,
     1        8,
     1        6,
     1        4,
     1        2,
     1        1,
     1        3,
     1        5,
     1        7,
     1        9,
     1        11,
     1        13,
     1        15,
```

```
     1        17,
     1        18,
     1        13*0
     1       /
     1     DELDELX1/.02/,
     1     DELDELX2/.01/,
     1     DELDELX3/-.005/,
     1     DELDELX4/-.0066666/,
     1     DELX/0.4/,
     1     DELXMAX/2./,
     1     DELXMIN/.4/,
     1     EFF/0./,
     1     ESTD0/.011300301/,
     1     NBFR/-351/,
     1     NLEV/19/,
     1     PBFR/351/,
     1     RUNIT/5/,
     1     SLISNR/MAXNSEG*0./,
     1     STDMAX/20.1/,
     1     STDMIN/.011300301/,
     1     VMAX/20./,
     1     WUNIT/6/,
     1     Y0/MAXLEV*0./

C---
C  Read in channel transmission rate
C---

      ERROR=1
1     WRITE(WUNIT,1000)
      READ(RUNIT,*,ERR=10000) TR

C---
C   Read in sampling rate
C---

      ERROR=17
2     WRITE(WUNIT,1100)
      READ(RUNIT,*,ERR=10000) SR


C---
C   Read in file specification
C---

      ERROR=18
3     WRITE(WUNIT,1200)
      READ(RUNIT,1300,ERR=10000) FILE_SPC

C---
C   Read in input type
C---

      ERROR=19
4     WRITE(WUNIT,1400)
      READ(RUNIT,1300,ERR=10000) INPUTT

C---
C   Open input file
C---
```

```
      ERROR=2
      IDUN=20
      OPEN(UNIT=IDUN,
     1     NAME=FILE_SPC,
     1     TYPE='OLD',
     1     ACCESS='DIRECT',
     1     ASSOCIATEVARIABLE=IDINDEX,
     1     ERR=10000)

C---
C   Read in input data and calculate the total number of input
C   blocks
C   * Note the file header is not read
C---

      ERROR=3
      NBLK=0
      DO 100 J=2,MAXBLK
      READ(IDUN'J,END=110,ERR=110) (IX(I),I=1,NSAMPR)
      NBLK=NBLK+1
      DO 100 I=1,NSAMPR
      X((J-2)*NSAMPR+I)=FLOAT(IX(I))
100   CONTINUE
      ERROR=6
      GO TO 10000
110   IF(NBLK.LT.1) GO TO 10000
      ERROR=5
      CLOSE(UNIT=IDUN,
     1     DISP='SAVE',
     1     ERR=10000)

C---
C   Calculate the total number of input samples
C---

      NS=NBLK*NSAMPR

C---
C   Calculate the total number of bits taken out of the buffer
C   per sampling period
C---

      BPSA=TR/SR

C---
C  Rescale the input samples
C---

      IF(INPUTT.EQ.'D') THEN
      SCALE=3./2815.4
      ELSE
      SCALE=1./715.54
      END IF
      DO 120 I=1,NS
      X(I)=X(I)*SCALE
120   CONTINUE

C---
C  Calculate NSEG
```

```
C---

        NSEG=(NS/SEGL)-2

C---
C   Calculate IL & IU
C---

        IL=(NLEV+1)/2
        IU=(NLEV/2)+1

C---
C   Simulate the input datas
C---

        CALL AUQ_ACSS_EC(NS,X,YQR)

C---
C   Read in output file name from user
C---

        ERROR=20
5       WRITE(WUNIT,1500)
        READ(RUNIT,1300,ERR=10000) FILE_SPC

C---
C   Store decoded output at receiver in file
C---

        ERROR=7
        IOPUN=90
        IOPINDEX=1
        OPEN(UNIT=IOPUN,
        1    NAME=FILE_SPC,
        1    TYPE='NEW',
        1    ACCESS='DIRECT',
        1    ASSOCIATEVARIABLE=IOPINDEX,
        1    RECORDSIZE=NSAMPR,
        1    ERR=10000)
        ERROR=4
        DO 140 J=2,NBLK+1
        DO 130 I=1,NSAMPR
        IYQR(I)=IFIX(YQR((J-2)*NSAMPR+I)/SCALE)
130     CONTINUE
        WRITE(IOPUN'J,ERR=10000) (IYQR(I),I=1,NSAMPR)
140     CONTINUE
        ERROR=5
        CLOSE(UNIT=IOPUN,
        1     DISP='SAVE',
        1     ERR=10000)

C---
C   Calculate the sliding SNR and segmental SNR
C---

        CALL SLI_SNR(X,YQR,NSEG,SEGL,NLEV,NS,SLISNR,SEGSNR)

C---
C   Store the sliding SNR into file
C---
```

```
        ERROR=8
        ISNRUN=91
        ISNRINDEX=1
        OPEN(UNIT=ISNRUN,
        1     NAME='SLISNR.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=ISNRINDEX,
        1     RECORDSIZE=1,ERR=10000)
        ERROR=9
        DO 150 I=1,NSEG
        WRITE(ISNRUN'ISNRINDEX,ERR=10000) SLISNR(I)
150     CONTINUE
        ERROR=10
        CLOSE(UNIT=ISNRUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   Store buffer occupancy in file
C---

        ERROR=11
        IBOUN=92
        IBOINDEX=1
        OPEN(UNIT=IBOUN,
        1     NAME='BFOC.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=IBOINDEX,
        1     RECORDSIZE=1,
        1     ERR=10000)
        ERROR=12
        DO 160 I=1,NS
        WRITE(IBOUN'IBOINDEX,ERR=10000) BFOC(I)
160     CONTINUE
        ERROR=13
        CLOSE(UNIT=IBOUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   change the number of occurence of each quantization level to
C   probability of occurence
C---

        DO 170 I=1,NLEV
        PROB(I)=QSLTOC(I)/NS
170     CONTINUE

C---
C   Calculate the quantizer output entropy
C---

        CALL ENTROPY(NLEV,PROB,ETP)

C---
C   Calculate the average code_word length
C---
```

```
        ACWL=0.
        DO 180 I=1,NLEV
        ACWL=ACWL+PROB(I)*CL(I)
180     CONTINUE

C---
C   Calculate the coding efficiency
C---

        EFF=ETP/ACWL

C---
C   Store the total number of samples lost due to transmission
C   buffer overflow, entropy, average code-word length, coding
C   efficiency, segmental SNR and the probability of occurence
C   of each quantizer level in file
C--

        ERROR=14
        ISTTUN=93
        ISTTINDEX=1
        OPEN(UNIT=ISTTUN,
        1     NAME='STTS.DAT',
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=ISTTINDEX,
        1     FORM='FORMATTED',
        1     RECORDSIZE=512,
        1     ERR=10000)
        ERROR=15
        WRITE(ISTTUN'ISTTINDEX,1600) NSL,
        1                            ETP,
        1                            ACWL,
        1                            EFF,
        1                            SEGSNR,
        1                            (PROB(I),I=1,NLEV)
        ERROR=16
        CLOSE(UNIT=ISTTUN,
        1      DISP='SAVE',
        1      ERR=10000)

C---
C   Program stop
C---

        STOP

C---
C   Prompt the error message
C---

10000   IF(ERROR.GT.16.OR.ERROR.EQ.1) THEN
        NERR=1
        ELSE
        NERR=ERROR
        END IF
        WRITE(WUNIT,*) '%ERROR >>> '//ERROR_MESSAGE(NERR)

C---
```

```
C   Prompt the enter message again
C---

        IF(ERROR.EQ.1) GO TO 1
        IF(ERROR.EQ.2) GO TO 3
        IF(ERROR.EQ.17) GO TO 2
        IF(ERROR.EQ.18) GO TO 3
        IF(ERROR.EQ.19) GO TO 4
        IF(ERROR.EQ.20) GO TO 5

C---
C   Program stop
C---

        STOP

C---
C    Formats
C---

1000    FORMAT('$ ENTER CHANNEL TRANSMISSION RATE  (R*4) >>> ')
1100    FORMAT('$ ENTER SAMPLING RATE  (R*4) >>> ')
1200    FORMAT('$ ENTER INPUT FILE SPECIFICATION ( < A50 ) >>> ')
1300    FORMAT(A)
1400    FORMAT('$ ENTER INPUT TYPE (Type D or DF) >>> ')
1500    FORMAT('$ ENTER OUTPUT FILE SPECIFICATION ( < A50 ) >>> ')
1600    FORMAT(I7,6(F15.7))

        END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       AUQ_AQSS_EC
C
C PURPOSE:
C
C       To simulate the coder with:-
C          (i)   a standard deviation estimator
C          (ii)  a uniform quantizer with quantization step-size
C                updated every sampling period for the elimination
C                of the buffer overflow problem
C          (iii) an entropy encoder
C
C MEDIA:
C
C       [NG.AECF]AUQAQSSEC.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       TO:-    NS      I*4     Total number of input samples
C               X       R*4     An array of input speech samples
C
C       From:-  YQR     R*4     An array of decoded quantizer outputs
C                               at the receiver
C
C IMPORTANT VARIABLES:
C
C       ALPHA           R*4     A constant for the equation to calculate
C                               the adaptive multipliers associated with
C                               each quantization level
C       BFOC            R*4     An array of buffer occupancy just before
C                               the next comming sampling instant
C       BFOCL           R*4     Buffer occupancy just before the present
C                               sampling instant
C       BPSR            R*4     Number of bits taken out of the buffer
C                               per sampling period
C       CL              I*4     An array of code-word length
C       DELDELX1        R*4     Delta DELX for increasing buffer occupancy
C                               and BFOC greater than 0
```

```
C       DELDELX2        R*4     Delta DELX for increasing buffer occupancy
C                               and BFOC less than 0
C       DELDELX3        R*4     Delta DELX for decreasing buffer occupancy
C                               and BFOC greater than 0
C       DELDELX4        R*4     Delta DELX for decreasing buffer occupancy
C                               and BFOC less than 0
C       DELX            R*4     The quantization step-size and adaptive
C                               buffer  management variable
C       DELXMAX         R*4     Maximum value of the adaptive buffer
C                               management variable DELX
C       DELXMIN         R*4     Minimum value of the adatpive bufter
C                               management variable DELX
C       ESTD            R*4     Output of STD estimator
C       ESTD0           R*4     Initial output of STD estimator
C       IL              I*4     Lower quantizer level
C       IU              I*4     Upper quantizer level
C       K               I*4     The kth decoded sample at the receiver
C       L               I*4     The quantization level number that the
C                               present sample falls
C                               set at any position within the buffer
C       NBFR            I*4     Negative buffer range with "0" reference
C                               point
C       NLEV            I*4     Total number of quantization levels
C       NS              I*4     Total number of input speech samples
C       PBFR            I*4     Positive buffer range with "0" reference
C                               point set at any position within the
C                               buffer
C       QINPUT          R*4     Quantizer input
C       QSLTOC          R*4     An array of the number of occurance of
C                               each quantization level
C       OFFT            I*4     Buffer overflow flat
C       STDMAX          R*4     Maximum output of the multiplier logic
C                               unit
C       STDMIN          R*4     Minimum output of the multiplier logic
C                               unit
C       UFFT            I*4     Buffer underflow flat
C       VMAX            R*4     Maximum allowable input amplitude
C       XRANGE          R*4     The quantizer quantization range
C       YQ              R*4     An array of quantizer output levels
C       YQR             R*4     An array of decoded quantizer output at
C                               receiver .
C
C COMMON BLOCKS:
C
C       See [NG.COMMONB]FBCODERCB.FOR
C
C ROUTINES REQUIRED:
C
C       BFMAN           In [NG.UTLPGM1 subdirectory
C       CMPQNT          <---------- " " ---------->
C       QUANT           <---------- " " ---------->
C
C SIDE EFFECTS:
C
C       (i)   return the array of number of occurance of each quantizer
C             level to calling program through common block PRMBLK
C       (ii)  return the array of buffer occupancy BFOC to calling
C             program through common block BFBLK
C       (iii) return the number of samples lost due to buffer overflow
C             to calling program through common block PRMBLK
```

```
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Small and speed depends on the length of input speech utterance(s)
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Nov.  ,1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       Not yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------


      SUBROUTINE AUQ_AQSS_EC(NS,X,YQR)

C---
C   Include feedback coder common block
C---

      INCLUDE '[NG.COMMONB]FBCODERCB.FOR'

C---
C   Set up the equivalence between (ALPHA,A), (DELDELX1,DELZ1),
C   (DELDELX2,DELZ2), (DELDELX3,DELZ3), (DELDELX4,DELZ4), (DELXMAX,ZMAX),
C   (DELXMIN,ZMIN), (ESTD0,MT0), (QSLTOC,PROB), (STDMAX,MTMAX) &
C   (STDMIN,MTMIN)
C---

      EQUIVALENCE
     1          (ALPHA,A),
     1          (DELDELX1,DELZ1),
     1          (DELDELX2,DELZ2),
     1          (DELDELX3,DELZ3),
     1          (DELDELX4,DELZ4),
     1          (DELXMAX,ZMAX),
     1          (DELXMIN,ZMIN),
     1          (ESTD0,MT0),
     1          (QSLTOC,PROB),
     1          (STDMAX,MTMAX),
     1          (STDMIN,MTMIN)
```

```
C---
C   Specific the type of OFFT & UFFT
C---

      INTEGER
     1        OFFT,
     1        UFFT

C---
C   Specify the dimension of QSLTOC, X & YQR
C---

      DIMENSION
     1          QSLTOC(MAXLEV),
     1          X(NS),
     1          YQR(NS)

C---
C    Initialize ACWL, BFOCL, K, NSL, QSLTOC, ESTD, BFOC & YQ
C---

      DATA
     1    ACWL/0./,
     1    BFOCL/0./,
     1    K/0/,
     1    NSL/0/,
     1    QSLTOC/MAXLEV*0./

      ESTD=ESTD0

      DO 100 I=1,NS
      BFOC(I)=0.
100   CONTINUE

      XRANGE=DELX*FLOAT(NLEV)/2.0
      CALL CMPQNT(NLEV,XRANGE,YQ)

C---
C   Start simulate the input
C---

      DO 120 I=1,NS

C---
C   Determine the input to the quantizer
C---

      IF(ABS(X(I)).GE.VMAX) THEN
      IF(X(I).GE.0.) THEN
      CX=VMAX
      ELSE
      CX=-VMAX
      END IF
      ELSE
      CX=X(I)
      END IF
      QINPUT=CX/ESTD

C---
C   Get the quantization level number for the input
```

```
C---

        CALL QUANT(QINPUT,L,YQ,NLEV)

C---
C   Increase the number of occurance of the corresponding quantization
C   level by 1
C---


        QSLTOC(L)=QSLTOC(L)+1

C---
C   Update the adaptive buffer managememt variable DELX and the buffer
C   occupancy
C---

        CALL BFMAN(BFOCL,BFSA,CL(L),DELDELX1,DELDELX2,DELDELX3,DELDELX4,
        1           PBFR,NBFR,DELXMAX,DELXMIN,DELX,BFOC(I),OFFT,UFFT)

C---
C   Check for buffer overflow
C---

        IF(OFFT.EQ.1) THEN

C---
C   In case of buffer overflow, then
C       (i)   increase the number of samples lost due to buffer overflow
C             by 1 and
C       (ii)  there is no decoded output at receiver
C---

        NSL=NSL+1
        ELSE

C---
C   In case buffer is not currently overflow, then
C       (i)    determine decoded quantizer output at receiver and
C       (ii)   increase the number of samples received by 1
C---
        K=K+1
        YQR(K)=YQ(L)*ESTD
        END IF

C---
C   Update the estimated STD
C---

        BETA=SQRT((1-ALPHA)*(YQ(L)**2)+ALPHA)
        ESTD=ESTD*BETA
        IF(ESTD.LE.STDMIN) ESTD=STDMIN
        IF(ESTD.GT.STDMAX) ESTD=STDMAX

C---
C   Update the quatizer output levels
C---

        XRANGE=DELX*FLOAT(NLEV)/2.0
        CALL CMPQNT(NLEV,XRANGE,YQ)
```

```
C---
C   Set BFOCL equal to BFOC(I)
C---

        BFOCL=BFOC(I)
120     CONTINUE

C---
C   Return to calling program
C---

        RETURN
        END
```

C. Block Quantization Module

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C
C MODULE NAME:
C
C       BLKQNT
C
C PURPOSE:
C
C       An interactive program simulate the coder which uses
C       block quantization to solve the buffer overflow problem
C       introduced by using variable length codes in digital
C       processing of speech signals.
C
C MEDIA:
C
C       [NG.BLKQNT]BLKQNT.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       $ Run [NG.BLKQNT]BLKQNT
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       None
C
C IMPORTANT VARIABLES:
C
C       See [NG.COMMONB]FFCODERCB.FOR
C       and
C       ACWL    R*4     Average code-word lenght
C       BMAX    I*4     Buffer Size
C       BS      I*4     Total number of bits required to code the side
C                       information
C       BT      I*4     BMAX-(NLEV-1)/2 bits
C       DELXK   R*4     The right quantization step-size delta Xk for
C                       the kth input vector
C       DELX1   R*4     Quantization step-size delta X1
C       DELX2   R*4     Quantization step-size delta X2
C                       (equals to 2*DELX1)
C       ETP     R*4     Entropy
C       FILE_-
C       SPC     A*60    File specification
C       GDELX1  R*4     Guess of delta X1
```

```
C       IBLK    I*4     Total number of input vectors
C       IBLKL   I*4     Vector dimension
C       IEND    I*4     End of the kth input vector
C       INPUTT  A*2     Input type; either direct (D) or difference
C                       (DE) input
C       ISLTOC  R*4     An array of number of occurance of each
C                       quantization level
C       IST     I*4     Beginning of the kth input vector
C       NBLK    I*4     Total number of blocks of input data with
C                       recordsize of each block equal to NSAMPR
C       RUNIT   I*4     Read unit
C       SR      R*4     Sampling rate in kSample/s
C       SEGSNR  R*4     Segmental SNR
C       SLISNR  R*4     Slide SNR
C       TR      R*4     Channel transmission rate in kb/s
C       WUNIT   I*4     Write unit
C
C COMMON BLOCKS:
C
C       [NG.COMMONB]FFCODERCB.FOR
C
C ROUTINES REQUIRED:
C
C       BQNTDP          In [NG.BLKQNT] subdirectory
C       CMPQNT          In [NG.UTLPGM] subdirectory
C       ENTROPY         <---------- " " ---------->
C       QUANT           <---------- " " ---------->
C       SDELXK          In [NG.BLKQNT] subdirectory
C       SII             <---------- "  " --------->
C       SLISNR          <---------- "  " --------->
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Medium and speed depends on length of input speech data
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 19, 1988.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None  yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------
```

```
      PROGRAM BLKQNT

C---
C   Include block quantization common block
C---

      INCLUDE '[NG.COMMONB]FFCODERCB.FOR'

C---
C   Set up the equivalence between (PROB,ISLTOC)
C---

      EQUIVALENCE
     1            (PROB,ISLTOC)

C---
C   Specify the type of BMAX, BS, BT, OPUN, OPINDEX, RUNIT,
C   WUNIT, IX, ISLTOC, FILE_SPC & INPUTT
C---

      INTEGER
     1        BMAX,
     1        BS,
     1        BT,
     1        OPUN,
     1        OPINDEX,
     1        RUNIT,
     1        WUNIT

      INTEGER*2 IX

      REAL
     1    ISLTOC

      CHARACTER
     1          FILE_SPC*60,
     1          INPUTT*2

C---
C   Specify the dimension of ISLTOC, IX & IYQR
C---

      DIMENSION
     1          ISLTOC(MAXLEV),
     1          IX(NSAMPR),
     1          IYQR(NSAMPR)

C---
C   Initialize  the variables
C---

      DATA
     1    BS/20/,
```

```
     1    CL
     1      /
     1        18,
     1        16,
     1        14,
     1        12,
     1        10,
     1        8,
     1        6,
     1        4,
     1        2,
     1        1,
     1        3,
     1        5,
     1        7,
     1        9,
     1        11,
     1        13,
     1        15,
     1        17,
     1        18,
     1        13*0
     1      /,
     1    DELXMIN/.00425/,
     1    DELXMAX/40./,
     1    ISLTOC/MAXLEV*0/,
     1    NLEV/19/,
     1    PROB/MAXLEV*0./,
     1    RUNIT/5/,
     1    SLISNR/MAXBLK*0./,
     1    VMAX/20./,
     1    WUNIT/6/

C---
C   Read in channel transmission rate
C---

        ERROR=1
1       WRITE(WUNIT,1000)
        READ(RUNIT,*,ERR=10000) TR

C---
C   Read in sampling rate
C---

        ERROR=17
2       WRITE(WUNIT,1100)
        READ(RUNIT,*,ERR=10000) SR

C---
C   Read in vector dimension
C---

        ERROR=18
3       WRITE(WUNIT,1200)
        READ(RUNIT,*,ERR=10000) IBLKL

C---
C   Read in input data file name
C---
```

```
        ERROR=19
4       WRITE(WUNIT,1300)
        READ(RUNIT,1400,ERR=10000) FILE_SPC

C---
C   Read in input type
C---

        ERROR=20
5       WRITE(WUNIT,1500)
        READ(RUNIT,1400,ERR=10000) INPUTT

C---
C   Open input file
C---

        ERROR=2
        IDUN=20
        IDINDEX=1
        OPEN(UNIT=IDUN,
        1    NAME=FILE_SPC,
        1    TYPE='OLD',
        1    ACCESS='DIRECT',
        1    ASSOCIATEVARIABLE=IDINDEX,
        1    ERR=10000)

C---
C   Read in input data and calculate the total number of input
C   blocks
C   *Note the file header, the first record, is not read
C---

        ERROR=3
        NBLK=0
        DO 100 J=2,MAXBLK
        READ(IDUN'J,END=110,ERR=110) (IX(I),I=1,NSAMPR)
        NBLK=NBLK+1
        DO 100 I=1,NSAMPR
        X((J-2)*NSAMPR+I)=FLOAT(IX(I))
100     CONTINUE
        ERROR=6
        GO TO 10000
110     IF(NBLK.LT.1) GO TO 10000
        ERROR=5
        CLOSE(UNIT=IDUN,
        1     DISP='SAVE',
        1     ERR=10000)

C---
C   Calculate the total number of input samples
C---

        NS=NBLK*NSAMPR

C---
C   Calculate the total number of input vectors of samples
C---

        IBLK=(NS/IBLKL)
```

```
C---
C   Calculate the total number of samples going to be
C   simulated
C---

        NNS=IBLK*IBLKL

C---
C   Rescale the input
C---

        IF(INPUTT.EQ.'D') THEN
        SCALE=3./2816.4
        ELSE
        SCALE=1./315.54
        END IF
        DO 120 I=1,NS
        X(I)=X(I)*SCALE
120     CONTINUE

C---
C   Calculate GDELX1, BMAX AND BT
C---

        GDELX1=DELXMAX
        BMAX=IFIX((FLOAT(IBLKL)/SR)*TR)-BS
        BT=BMAX-(NLEV/2)

C---
C   Simulate the input speech samples and generate the
C   corresponding decoded output at the receiver
C---

        DO 140 K=1,IBLK

        IST=(K-1)*IBLKL+1                                    ! Specify the
                                                             ! beginning
                                                             ! of the kth
        IEND=IST+IBLKL-1                                     ! input vector
                                                             ! Specify the
                                                             ! end of the
                                                             ! equalkth input
                                                             ! vector
        CALL SII(NLEV,DELXMAX,DELXMIN,GDELX1,BMAX,
        1         IST,IEND,DELX2,DELX1)                      ! Search the
                                                             ! interval
                                                             ! [delX1,
                                                             !  2*delX1]
        CALL SDELXK(DELX2,DELX1,BT,BMAX,IST,IEND,DELXK,YQ)   ! Search delXk
                                                             ! Calculate the
                                                             ! decoded
                                                             ! output at the
                                                             ! receiver
        DO 130 I=IST,IEND
        CALL QUANT(X(I),L,YQ,NLEV)                           ! Calculate the
        YQR(I)=YQ(L)                                         ! decoded output
                                                             ! at the
                                                             ! receiver
        ISLTOC(L)=ISLTOC(L)+1                                ! Update the
```

```
                                                                    ! number of
                                                                    ! occurance of
                                                                    ! each quanti-
                                                                    ! zation level
130     CONTINUE

        GDELX1=DELXK                                                ! Update initial
                                                                    ! guess for delX
140     CONTINUE

C---
C   Calculate the sliding SNR
C---

        CALL SLI_SNR(X,YQR,IBLK,IBLKL,NLEV,NNS,SLISNR,SEGSNR)

C---
C   Change the number of occurance of each quantization level to
C   probability of occurance
C---

        DO 150  I=1,NLEV
        PROB(I)=ISLTOC(I)/NNS
150     CONTINUE

C---
C   Calculate the entropy
C---

        CALL ENTROPY(NLEV,PROB,ETP)

C---
C   Calculate the average code-word length
C---
        ACWL=0
        DO 160 I=1,NLEV
        ACWL=ACWL+PROB(I)*CL(I)
160     CONTINUE

C---
C   Read in output file name
C---

        ERROR=21
6       WRITE(WUNIT,1600)
        READ(RUNIT,1400,ERR=10000) FILE_SPC

C---
C   Store the decoded ouput at the receiver in file
C---

        ERROR=7
        OPUN=90
        OPINDEX=1
        OPEN(UNIT=OPUN,
        1     NAME=FILE_SPC,
        1     TYPE='NEW',
        1     ACCESS='DIRECT',
        1     ASSOCIATEVARIABLE=OPINDEX,
        1     RECORDSIZE=NSAMPR,
```

```
     1    ERR=10000)
      ERROR=4
      NBLKP=NNS/NSAMPR
      DO 180  J=2,NBLKP+1
      DO 170  I=1,NSAMPR
      IYQR(I)=IFIX(YQR((J-2)*NSAMPR+I)/SCALE)
170   CONTINUE
      WRITE(OPUN'J) (IYQR(I),I=1,NSAMPR)
180   CONTINUE
      ERROR=5
      CLOSE(UNIT=OPUN,
      1     DISP='SAVE',
      1     ERR=10000)

C---
C   Store sliding SNR in file
C---

      ERROR=8
      ISNRUN=91
      ISNRINDEX=1
      OPEN(UNIT=ISNRUN,
      1    NAME='SLISNR.DAT',
      1    TYPE='NEW',
      1    ACCESS='DIRECT',
      1    ASSOCIATEVARIABLE=ISNRINDEX,
      1    RECORDSIZE=1,
      1    ERR=10000)
      ERROR=9
      DO 190 I=1,IBLK
      WRITE(ISNRUN'ISNRINDEX) SLISNR(I)
190   CONTINUE
      ERROR=10
      CLOSE(UNIT=ISNRUN,
      1     DISP='SAVE',
      1     ERR=10000)

C---
C   Store entropy, average code-word length, segmental SNR and the
C   probability of occurance of each quantizer level in file
C---

      ERROR=14
      ISTTUN=92
      ISTTINDEX=1
      OPEN(UNIT=ISTTUN,
      1    NAME='STTS.DAT',
      1    TYPE='NEW',
      1    ACCESS='DIRECT',
      1    ASSOCIATEVARIABLE=ISTTINDEX,
      1    FORM='FORMATTED',
      1    RECORDSIZE=512,
      1    ERR=10000)
      ERROR=15
      WRITE(ISTTUN'ISTTINDEX,1700)
      1                           ETP,
      1                           ACWL,
      1                           SEGSNR,
      1                           (PROB(I),I=1,NLEV)
      ERROR=16
```

```
      CLOSE(UNIT=ISTTUN,
     1      DISP='SAVE',
     1      ERR=10000)

C---
C   End of program
C---


      STOP

C---
C   Prompt error message
C---

10000 IF(ERROR.GT.16.OR.ERROR.EQ.1) THEN
      NERR=1
      ELSE
      NERR=ERROR
      END IF
      WRITE(WUNIT,*) '%ERROR >>> '//ERROR_MESSAGE(NERR)

C---
C   Prompt the enter message again
C---

      IF(ERROR.EQ.1) GO TO 1
      IF(ERROR.EQ.2) GO TO 4
      IF(ERROR.EQ.17) GO TO 2
      IF(ERROR.EQ.18) GO TO 3
      IF(ERROR.EQ.19) GO TO 4
      IF(ERROR.EQ.20) GO TO 5
      IF(ERROR.EQ.21.OR.7) GO TO 6

C---
C   Program stop
C---

      Stop

C---
C   Formats
C---

1000  FORMAT('$ENTER CHANNEL TRANSMISSION RATE (R*4) >>> ')
1100  FORMAT('$ENTER SAMPLING RATE (R*4) >>> ')
1200  FORMAT('$ENTER VECTOR DIMENSION (I*4) >>> ')
1300  FORMAT('$ENTER INPUT FILE NAME (A60) >>> ')
1400  FORMAT(A)
1500  FORMAT('$ENTER INPUT TYPE (TYPE D OR DF) >>> ')
1600  FORMAT('$ENTER OUTPUT FILE NAME (A*60) >>> ')
1700  FORMAT(63F15.7)
      END
```

```
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       BQNTDP
C
C PURPOSE:
C
C       To calculate the total number of bits  required to code
C       a vector of speech samples for a given quantization
C       step-size
C
C MEDIA:
C
C       [NG.BLKQNT]BQNTDP.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply parameters stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    IEND    I*4     End of the kth input vector
C               IST     I*4     Beginning of the kth input vector
C               YQ      R*4     An array of output levels of the
C                               quantizer
C
C       From:-  BN      R*4     Total number of bits required to code the
C                               kth input vector for a given
C                               quantization step-size
C
C IMPORTANT VARIABLES:
C
C       CL      I*4     An array of code-word length
C       ISLTOC  I*4     An array of the number of occurence of
C                       each quantization level
C       QINPUT  R*4     Input to the quantizer
C       VMAX    R*4     Maximun allowable input amplitude
C
C COMMON BLOCKS:
C
C       See [NG.COMMONBJFFCODERCB.FOR
C
C ROUTINES REQUIRED:
C
C       QUANT           In [NG.UTLPGM] subdirectory
```

```
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Small and speed depends on vector dimension
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       OCT. 19, 1980.
C
C
C MAINTAINED BY:
C
C       NG. S.C.
C
C UPDATES:
C
C       None yet
C
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------


      SUBROUTINE BQNTDP(IST,IEND,YQ,BN)

C---
C  Include block quantization common block
C---
      INCLUDE 'DNG.COMMONBJFFCODERCB.FOR'

C---
C   Specify the type of BN
C---
      INTEGER BN

C---
C   Specify the dimension of ISLTOC
C---
      DIMENSION ISLTOC(MAXLEV)

C---
C   Initialize varisbles
C---
```

```
        BN=0
        DO 100 I=1,NLEV
        ISLTOC(I)=0
100     CONTINUE

C---
C   Get the number of occurance of each quantization slot
C---

        DO 200 I=IST,IEND

        IF(ABS(X(I)).GT.VMAX) THEN                  !Limit the input amplit-
        IF(X(I).GE.0.) QINPUT=VMAX                  !ude towithin the range
        IF(X(I).LT.0.) QINPUT=-VMAX                 ![-VMAX, VMAX]
        ELSE
        QINPUT=X(I)
        END IF
        CALL QUANT(QINPUT,L,VQ,NLEV)                !Get the quantization level
                                                    !number the input sample
                                                    !falls
        ISLTOC(L)=ISLTOC(L)+1                       !Update the number of
                                                    !occurance of this
                                                    !quantization level
200     CONTINUE

C---
C   Calculate BN
C---
        DO 300 I=1,NLEV
        BN=BN+ISLTOC(I)*CL(I)
300     CONTINUE

C---
C   Return to main program
C---

        RETURN
        END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       SDELXK
C
C PURPOSE:
C
C       To search the right quantization step-size DELXK such that
C       the total number of bits  BN required to code the input
C       vector of speech samples is less than BMAX but greater BT
C
C MEDIA:
C
C       ING. BLKQNTJSDELXK
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    BMAX    I*4     Buffer size
C               BT      I*4     BMAX-(NLEV-1)/2 bits
C               DELX1   R*4     Quantization step-size delta X1
C               DELX2   R*4     Quantization step-size delta X2
C                               (equal to 2*DELX1)
C               IEND    I*4     End of the kth input vector
C               IST     I*4     Beginning of the kth input vector
C
C       From:-  DELXK   R*4     Quantization step-size delta Xk for the
C                               kth input vector with corresponding BN,
C                               the total number of bits required to
C                               code the whole vector, less than BMAX
C                               but greater BT
C               Y0      R*4     An array of quantizer output levels
C
C IMPORTANT VARIABLES:
C
C       BN      I*4     The total number of bits required to code the
C                       whole input vector with quantization step-size
C                       equal to DELXK
C       B1      I*4     The total number of bits required to code the
C                       whole input vector with quantization step-size
C                       equal to DELX1
```

```
C        B2       I*4      The total number of bits required to code the
C                          whole input vector with quantization step-size
C                          equal to DELX2=2*DELX1
C        MAXNIT   I*4      Maximun number of iteration
C
C COMMON BLOCKS:
C
C        [NG.COMMONB]FFCODERCB.FOR and
C        BITBLK
C
C ROUTINES REQUIRED:
C
C        BQNTDP           In [NG.BLKQNT] subdirectory
C        CMPQNT           In [NG.UTLPGM] subdirectory
C
C SIDE EFFECTS:
C
C        None
C
C SPECIAL TECHNIQUES:
C
C        None
C
C PROGRAM SIZE AND SPEED:
C
C        Small and speed depends on the vector dimension IBLKL
C
C AUTHOR:
C
C        NG S.C.
C
C DATE CREATED:
C
C        Nov. 3, 1980.
C
C MAINTAINED BY:
C
C        NG S.C.
C
C UPDATES:
C
C        None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------


        SUBROUTINE SDELXK(DELX2,DELX1,BT,BMAX,IST,IEND,DELXK,YQ)
C---
C   Include feedback coder common block
C---
        INCLUDE '[NG.COMMONB]FFCODERCB.FOR'
C---
```

```
C   Specify the type of BMAX, BN, BT, B1 and B2
C---

        INTEGER
        1       BMAX,
        1       BN,
        1       BT,
        1       B1,
        1       B2

C---
C   Create common block BITBLK
C---

        COMMON /BITBLK/B1,B2

C---
C   Initialization
C---

        I=0
        BN=0

C---
C   Calculate the range that BN has to fall
C---

        ITH=BMAX-BT
C---
C   Calculate the number of quantization level for positive input
C   amplitude
C--
        Z=FLOAT(NLEV)/2.0

C---
C   Check if B1 is less than BMAX
C---

        IF(B1.LE.BMAX) THEN
C---
C   If B1 less than BMAX then set the quantization step-size for the
C   kth input vector equal to DELX1 and return to calling program
C---

        DELXK=DELX1
        GO TO 300
        END IF
C---
C   Else search for the right quantization step_size
C---

100     DELXT=DELX2-((BMAX-B2)*(DELX2-DELX1)/(B1-B2))     ! Calculate
                                                          ! DELXT

C---
C   If the number of iterations greater than the maximum number of
C   allowable iteration, set DELXT equal to the 2*DELX1, i.e. the
C   upper limit of the quantization step-size
C---
```

```
        IF(I.GT.MAXNIT) DELXT=DELX2

C---
C   Calculate the quantizer output levels for DELXT
C---

        XRANGE=DELXT*Z
        CALL CMPQNT(NLEV,XRANGE,YQ)

C---
C   Return to calling program if the number of iterations exit the
C   maximum number of allowable iterations
C---

        IF(I.GT.MAXNIT) GO TO 200

C---
C   Else calculate the total number of bits required to code this
C   vector
C---

        CALL BQNTDP(IST,IEND,YQ,BN)

C---
C   Increase number of iterations by 1
C---

        I=I+1
C---
C   Go to 200 if BT < BN < BMX
C---

        ITEST=BMAX-BN
        IF(ITEST.GE.0) THEN
        IF(ITEST.LT.ITH) GO TO 200

C---
C   Set DELX2 equal to DELXT and B2 equal to BN if BN < BT
C   and go to 100, i.e. keep on searching for DELXK
C---

        DELX2=DELXT
        B2=BN
        ELSE

C---
C   Set DELX1 equal to DELXT and B1 equal to BN if BMAX < BN
C   and go to 100, i.e. keep on searching for DELXK
C---

        DELX1=DELXT
        B1=BN
        END IF
        GO TO 100

C---
C   Set the quantization step-size equal to DELXT
C---

200     DELXK=DELXT
```

```
C---
C   Return to calling program
C---
300     RETURN
        END
```

```
C-------------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       SII
C
C PURPOSE:
C
C       To search the quantization step-sizes delta X1 and 2 times
C       delta X1 such that B1 and B2, the corresponding total number
C       of bits required to code the vector of speech samples,
C       satisfy the condition B2 < BMAX < B1
C
C MEDIA:
C
C       [NG.BLKONTJSII.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    BMAX    I*4     Buffer size
C               DELXMAX R*4     Maximum quantization step-size
C               DELXMIN R*4     Minimum quantization step-size
C               DELX1   R*4     Quantization step-size delta X1
C               DELX2   R*4     Quantization step-size delta X2
C                               equal to 2*DELX1
C               GDELX1  R*4     Guess of delta X1
C               IEND    I*4     End of the kth. input vector
C               IST     I*4     Beginning of the kth. input vector
C               NLEV    I*4     Total number of quantization levels
C
C       From:-  DELX1   R*4     Quantization step-size delta X1
C               DELX2   R*4     Quantization step-size delta X2
C                               (equals to 2*DELX1)
C
C IMPORTANT VARIABLES:
C
C       B1      I*4     The total number of bits required to code the
C                       vector of speech samples with quantization
C                       step-size equal DELX1
C       B2      I*4     The total number of bits required to code the
C                       vector of speech samples with quantization
C                       step-size equal 2*DELX2=DELX1
```

```
C       YQ      R*4     An array of quantizer output levels
C
C COMMON BLOCKS:
C
C       BITBLK
C
C ROUTINES REQUIRED:
C
C       BQNTDP          In [NG.BLKQNT] subdirectory
C       CMPQNT          In [NG.UTLPGM] subdirectory
C
C SIDE EFFECTS:
C
C       Return B1 and B2 to calling program in common block BITBLK
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       small and speed depends on the input vector dimension
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None Yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------


      SUBROUTINE SII(NLEV,DELXMAX,DELXMIN,GDELX1,BMAX,IST,IEND,
     1               DELX2,DELX1)

C---
C   Specify the dimension of YQ
C---

      DIMENSION YQ(100)

C---
C   Define the common block BITBLK
C---

      COMMON /BITBLK/B1,B2
```

```
C---
C   Specify the type of B1,B2 and B3
C---

        INTEGER
        1       BMAX,
        1       B1,
        1       B2,
        1       B3

C---
C   Calculate the total number of bits B3 required to code the whole
C   vector with quantization step-size equals to GDELX1
C---

        DELXT=GDELX1
        IFLAT=0
        Z=FLOAT(NLEV)/2.0
        PARM=DELXT*Z
        CALL CMPQNT(NLEV,PARM,YQ)
        CALL BQNTDP(IST,IEND,YQ,B3)

C---
C   Set up the flat if B3 is greater than the buffer size
C---

        IF(B3.GT.BMAX) IFLAT=1
        IF(IFLAT.EQ.0) GO TO 10

C---
C   If B3 is greater than the buffer size then
C   (i)   let DELX1=DELXT
C   (ii)  let B1=B3 and
C   (iii) double the quantization step-size DELXT
C---

5       DELX1=DELXT
        B1=B3
        DELXT=DELXT*2
        GO TO 20

C---
C   If B3 is less than the buffer size then
C   (i)   let DELX2=DELXT
C   (ii)  let B2=B3
C   (iii) reduce the quantization step-size DELXT by half
C---

10      DELX2=DELXT
        B2=B3
        DELXT=DELXT/2

C---
C Check if DELXT is within the range [DELXMIN,DELXMAX]
C---

20      IF(DELXT.GT.DELXMAX) DELXT=DELXMAX
        IF(DELXT.LT.DELXMIN) DELXT=DELXMIN
```

```
C---
C   Calculate the total number of bits B3 required to code the whole
C   vector with updated quantization step-size DELXT
C---

        PARM=DELXT*2
        CALL CMPQNT(NLEV,PARM,YQ)
        CALL BQNTDP(IST,IEND,YQ,B3)

C---
C   If the flat is already set and B3 is less than the buffer size,
C   then
C       (i)   let DELX2=DELXK
C       (ii)  Let B2=B3 and
C       (iii) exit
C---

        IF(IFLAT.EQ.1) THEN
        IF(B3.LT.BMAX) THEN
        DELX2=DELXT
        B2=B3
        GO TO 30
        ELSE

C---
C   If flat is set but B3 is still greater than buffer size, then
C   update DELX1 and B1 and keep on searching DELX2 and B2
C---

        GO TO 5
        END IF
        END IF

C---
C   If the flat is not set and
C   (i)   if B3 is less than the buffer size and DELXT is greater
C         than DELXMIN, then update DELX2 and B2 and keep on
C         searching DELX1 and B1
C   (ii)  if conditions in (i) are not satisfied, then update DELX1
C         and B1 and exit
C---

        IF(B3.LT.BMAX.AND.DELXT.GT.DELXMIN) THEN
        GO TO 10
        ELSE
        DELX1=DELXT
        B1=B3
        END IF

C---
C   Return to calling program
C---

30      RETURN
        END
```

D. Common Block Module

```
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       FBCODERCB.FOR
C
C PURPOSE:
C
C       (i)   To provide a common file for a coder with
C               (a)  a fixed uniform quantizer
C               (b)  adaptive expansion and contraction factors
C                    associated with each quantization level and
C               (c)  an entropy encoder  and
C       (ii)  To indicate the limitation of the simulation programs
C
C MEDIA:
C
C       [NG.COMMONB]FBCODERCB.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Include this file in appropiate programs
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       None
C
C IMPORTANT VARIABLES:
C
C       A       R*4     A constant for the equation to calculate the
C                       multipliers associated with each quantization
C                       level
C       ACWL    R*4     Average code-word length
C       BFOC    R*4     An array of buffer occupancy just before the
C                       next comming sampling instant
C       BPSA    R*4     Number of bits taken out of the buffer per
C                       sampling interval
C       CL      I*4     An array of code-word length
C       DELX    R*4     Quantization step-size
C       DELZ1   R*4     Delta Z for increasing buffer occupancy and BFOC
C                       greater than 0
C       DELZ2   R*4     Delta Z for incressing buffer occupancy and BFOC
C                       less than 0
C       DELZ3   R*4     Delta Z for decreasing buffer occupancy and BFOC
```

```
C                                greater than 0
C       DELZ4    R*4             Delta Z for decreasing buffer occupancy and BFOC
C                                less than 0
C       ERROR    I*4             Error number
C       ERROR_-
C       MESSAGE  A*60            An array of error messages
C       IL       I*4             Lower quantizer level
C       IU       I*4             Upper quantizer level
C       M        R*4             An array of multipliers associated with each
C                                quantization level
C       MAXBLK   I*4             Maximum number of blocks of records
C       MAXERR-
C       ROMESS-
C       AGE      I*4             Maximum number of error message
C       MAXNSEG  I*4             Maximum number of segments or vectors
C       MAXLEV   I*4             Maximum un number of quantization levels
C       MT       R*4             Output of the multiplier logic unit
C       MTMAX    R*4             Maximum output of the multiplier logic unit
C       MTMIN    R*4             Minimum output of the multiplier logic unit
C       MT0      R*4             Initial output of the multiplier logic unit
C       NBFR     I*4             Negative buffer range with "0" reference point
C                                set at any position within the buffer
C       NLEV     I*4             Number of quantization levels
C       NSAMPR   I*4             Number of samples per record (machine dependent)
C       NSEG     I*4             Number of segments or blocks
C       NSL      I*4             Total number of samples lost due to buffer
C                                overflow
C       PBFR     I*4             Positive buffer range with "0" reference point
C                                set at any position within the buffer
C       PROB     R*4             An array of probability of occurance of each
C                                quantization levels
C       SEGL     I*4             Number of samples per segment or block
C       SLISNR   R*4             Sliding SNR
C       YQ       R*4             An array of quantizer output levels
C       VMAX     R*4             Maximum allowable input amplitude
C       ZMAX     R*4             Maximum value of the adaptive buffer management
C                                variable Z
C       ZMIN     R*4             Minimum value of the adaptive buffer management
C                                variable Z
C
C COMMON BLOCKS:
C
C       BFBLK                    In this file
C       BFMBLK                   In this file
C       PRMBLK                   In this file
C       QNTBLK                   In this file
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
```

```
C
C       Small and fast
C
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------
C---
C   Define parameters MAXBLK, MAXERRORMESSAGWE, MAXLEV, MAXNSEG
C   & NSAMPR
C----
        PARAMETER
        1           MAXBLK=150,
        1           MAXERRORMESSAGE=16,
        1           MAXLEV=32,
        1           MAXNSEG=384,
        1           NSAMPR=256
C---
C   Specify the type of CL, ERROR, PBUFR and SEGL
C---
        INTEGER
        1       CL,
        1       ERROR,
        1       PBFR,
        1       SEGL
C---
C   Specify the type of M, MT, MTMAX, MTMIN and MT0
C---
```

```
      REAL
     1     M,
     1     MT,
     1     MTMAX,
     1     MTMIN,
     1     MT0

C---
C   Specify the type of ERROR_MESSAGE
C---

      CHARACTER
     1          ERROR_MESSAGE*60

C---
C   Specify the dimension of BFOC, CL, ERROR_MESSAGE, M, PROB, SLISNR
C   and YQ
C---

      DIMENSION
     1          BFOC(MAXBLK*NSAMPR),
     1          CL(MAXLEV),
     1          ERROR_MESSAGE(MAXERRORMESSAGE),
     1          M(MAXLEV),
     1          PROB(MAXLEV),
     1          SLISNR(MAXNSEG),
     1          YQ(MAXLEV)

C---
C   Create common blocks BFBLK, BFMBLK, PRMBLK and QNTBLK
C---

      COMMON
     1        /BFBLK/BFOC, NBFR, PBFR,
     1        /BFMBLK/DELZ1, DELZ2, DELZ3, DELZ4, ZMAX, ZMIN,
     1        /PRMBLK/IL, IU, PROB, VMAX, MTMAX, MTMIN, MT0,
     1                BPSA, ACWL, NSL, NSEG, SEGL,
     1        /QNTBLK/A, CL, DELX, M, NLEV, YQ

C---
C   Initialize ERROR, ERROR_MESSAGE, M, PROB, SEGL, SLISNR  and YQ
C---

      DATA
     1        ERROR/0/,
     1        ERROR_MESSAGE
     1            /
     1             ' ACCESS MODE VIOLATION ',
     1             ' FAIL TO OPEN SPECIFIED FILE ',
     1             ' FAIL TO READ SPECIFIED FILE ',
     1             ' FAIL TO WRITE SPECIFIED FILE ',
     1             ' FAIL TO CLOSE SPECIFIED FILE ',
     1             ' SPECIFIED BLOCK LONGER THAN 150 BLOCKS OF RECORDS',
     1             ' FAIL TO CREATE SPECIFIED FILE ',
     1             ' FAIL TO CREATE SLIDING SNR FILE ',
     1             ' FAIL TO WRITE INTO SLIDING SNR FILE ',
     1             ' FAIL TO CLOSE SLIDING SNR FILE ',
     1             ' FAIL TO CREATE BUFFER OCCUPANCY FILE ',
     1             ' FAIL TO WRITE INTO BUFFER OCCUPANCY FILE ',
     1             ' FAIL TO CLOSE BUFFER OCCUPANCY FILE ',
```

```
1              ' FAIL TO CREATE STATISTICS FILE ',
1              ' FAIL TO WRITE INTO STATISTICS FILE ',
1              ' FAIL TO CLOSE STATISTICS FILE '      .
1              /,
1         M/MAXLEV*0. /,
1         PROB/MAXLEV*0. /,
1         SEGL/100/,
1         SLISNR/MAXNSEG*0. /,
1         YQ/MAXLEV*0. /
```

```
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       FFCODERCB.FOR
C
C PURPOSE:
C
C       To provide a common file for block quantization and to
C       indicate the limitation of the simulation programs
C
C MEDIA:
C
C       [NG.COMMONB]FFCODERCB.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Include this common file in appropiate programs
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       None
C
C IMPORTANT VARIABLES:
C
C       CL       I*4      Code-word length
C       DELXMAX  R*4      Maximum quantization step-size
C       DELXMIN  R*4      Minimum quantization step-size
C       ERROR    I*4      Error number
C       ERROR_-
C       MESSAGE  A*60     An array of error messages
C       MAXBLK   I*4      Maximum number of input blocks
C       MAXERR-
C       ORMESS-
C       AGE      I*4      Maximum number of error messages
C       MAXLEV   I*4      Maximum number of quantization levels
C       MAXNIT   I*4      Maximum number of iteration
C       NLEV     I*4      Number of quantization levels
C       NNS      I*4      Number of input speech samples simulated
C       NS       I*4      Number of input speech samples
C       NSAMPR   I*4      Number of samples per record (machine dependent)
C       PROB     R*4      An array of probability of occurance of each
C                         quantization level
C       SLISNR   R*4      An array of sliding SNR
C       X        R*4      An array of input speech samples
C       Y0       R*4      An array of quantizer output levels
C       YQR      R*4      An array of decoded outputs at receiver
```

```
C       VMAX     R*4      Maximum allowable input amplitude
C
C
C COMMON BLOCKS:
C
C       INPBLK            In this file
C       OUTPBLK           In this file
C       PRMBLK            In this file
C       QNTBLK            In this file
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Small and fast
C
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       Not yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory-------------


C---
C   Define parameters MAXBLK, MAXERRORMESSAGE, MAXLEV, MAXNIT,
C   & NSAMPR
C---

      PARAMETER
     1          MAXBLK=150,
     1          MAXERRORMESSAGE=16,
     1          MAXLEV=32,
```

```
     1          MAXNIT=5,
     1          NSAMPR=256

C---
C   Specify the type of CL, ERROR & ERROR_MESSAGE
C---

      INTEGER
     1        CL,
     1        ERROR

      CHARACTER
     1          ERROR_MESSAGE*60

C---
C   Specify the dimensions of CL, PROB, SLISNR, X, YQR & YQ
C---

      DIMENSION
     1          CL(MAXLEV),
     1          ERROR_MESSAGE(MAXERRORMESSAGE),
     1          PROB(MAXLEV),
     1          SLISNR(MAXBLK),
     1          X(MAXBLK*NSAMPR),
     1          YQR(MAXBLK*NSAMPR),
     1          YQ(MAXLEV)

C---
C   Create common blocks INPBLK, OUTPBLK, PRMBLK and QNTBLK
C---

      COMMON
     1       /INPBLK/X,
     1       /OUTPBLK/YQR,
     1       /PRMBLK/NS, NNS, PROB, VMAX,
     1       /QNTBLK/NLEV, CL, DELXMIN, DELXMAX

C---
C   Initialize ERROR & ERRORMESSAGE
C---

      DATA
     1        ERROR/0/,
     1        ERROR_MESSAGE
     1            /
     1             ' ACCESS MODE VIOLATION ',
     1             ' FAIL TO OPEN SPECIFIED FILE ',
     1             ' FAIL TO READ SPECIFIED FILE ',
     1             ' FAIL TO WRITE SPECIFIED FILE ',
     1             ' FAIL TO CLOSE SPECIFIED FILE ',
     1             ' SPECIFIED BLOCK LONGER THAN 150 BLOCKS OF RECORDS',
     1             ' FAIL TO CREATE SPECIFIED FILE ',
     1             ' FAIL TO CREATE SLIDING SNR FILE ',
     1             ' FAIL TO WRITE INTO SLIDING SNR FILE ',
     1             ' FAIL TO CLOSE SLIDING SNR FILE ',
     1             ' FAIL TO CREATE BUFFER OCCUPANCY FILE ',
     1             ' FAIL TO WRITE INTO BUFFER OCCUPANCY FILE ',
     1             ' FAIL TO CLOSE BUFFER OCCUPANCY FILE ',
     1             ' FAIL TO CREATE STATISTICS FILE ',
     1             ' FAIL TO WRITE INTO STATISTICS FILE ',
```

```
1               ' FAIL TO CLOSE STATISTICS FILE '
1               /
```

E. Utility Programs Module

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       BFMAN
C
C PURPOSES:
C
C       (i)   To update the adaptive buffer management variable Z
C       (ii)  To determine the buffer occupancy just before the
C             next comming sampling instant
C       (iii) To set overflow flat in case of buffer overflow and
C       (iv)  To set underflow flat in case of buffer underflow
C
C MEDIA:
C
C       [NG.UTLPGM]BFMAN.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    BFOCL   R*4     The buffer occupancy just before the
C                               present sampling instant
C               BPSR    R*4     Total number of bits taken out of the
C                               buffer per sampling period
C               CL      I*4     Code-word length of the present sample
C               DELZ1   R*4     Delta Z for increasing buffer occupancy
C                               and BFOC greater than 0
C               DELZ2   R*4     Delta Z for increasing buffer occupancy
C                               and BFOC less than 0
C               DELZ3   R*4     Delta Z for decreasing buffer occupancy
C                               and BFOC greater than 0
C               DELZ4   R*4     Delta Z for decreasing buffer occupancy
C                               and BFOC less than 0
C               NBFR    I*4     Negative buffer range with "0" reference
C                               point set at any position within the
C                               buffer
C               PBFR    I*4     Positive buffer range with "0" reference
C                               point set at any position within the
C                               buffer
C               ZMAX    R*4     Maximun value of the adaptive buffer
```

```
C                                        management variable Z
C                ZMIN      R*4           Minimun value of the adaptive buffer
C                                        management variable Z
C                Z         R*4           The adaptive buffer management variable
C
C       From:-   BFOC      R*4           The buffer occupancy just before the next
C                                        comming sampling instant
C                OFFT      I*4           Buffer overflow flat
C                UFFT      I*4           Buffer underflow flat
C                Z         R*4           The adaptive buffer management variable
C
C IMPORTANT VARIABLES:
C
C       DELZ     R*4       Delta Z
C
C
C COMMON BLOCKS:
C
C       None
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       Nnone
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Small and fast
C
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Nov. ,1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------
```

```
      SUBROUTINE  BFMAN(BFOCL,BPSA,CL,DELZ1,DELZ2,DELZ3,DELZ4,
     1                  PBFR,NBFR,ZMAX,ZMIN,Z,BFOC,OFFT,UFFT)

C---
C   Specify the type of CL, OFFT, PBFR & UFFT
C---

      INTEGER
     1        CL,
     1        OFFT,
     1        PBFR,
     1        UFFT

C---
C   Clear buffer overflow and underflow flat
C---

      OFFT=0
      UFFT=0

C---
C   Check if there is enough room in the buffer to store the
C   code-word of the present sample; i.e. check for buffer
C   overflow
C---

      IF(CL.GT.PBFR-BFOCL) THEN

C---
C   In case of buffer overflow, then
C       (i)   set buffer overflow flat
C       (ii)  determine the buffer occupancy just before the next
C             comming sampling instant with the present sample not
C             being dumpt into the buffer
C       (iii) set the adaptive buffer management variable to its
C             maximun value
C       (iv)  return to calling program
C---

      OFFT=1
      BFOC=BFOCL-BPSA
      Z=ZMAX
      GOTO 100
      END IF

C---
C   Check for buffer underflow
C---

      IF(BPSA.GT.CL+BFOCL-NBFR) THEN
C---
C   In case of buffer underflow, then
```

```
C        (i)    set the buffer underflow flat
C        (ii)   determine the buffer occupancy just before the next
C               comming sampling instant with no bits being taken out of
C               the buffer in this present sampling period
C        (iii)  set the adaptive buffer management variable to its
C               minimun value and
C        (iv)   return to calling program
C---

         UFFT=1
         BFOC=BFOCL+CL
         Z=ZMIN
         GOTO 100
         END IF

C---
C   In case of neither buffer overflow nor underflow, then
C        (i)    determine the buffer occupancy occupancy just before
C               the next comming sampling instant
C        (ii)   determine the change DELZ in the adaptive buffer
C               management variable Z according to the buffer occupancy
C               just before the next comming sampling instant and the
C               tendency of change of the buffer occupancy
C---

         BFOC=BFOCL+FLOAT(CL)-BPSR
         IF(BFOC.GE.0) THEN
         IF(BFOC.GE.BFOCL) THEN
         DELZ=DELZ1
         ELSE
         DELZ=DELZ3
         END IF
         ELSE
         IF(BFOC.GT.BFOCL) THEN
         DELZ=DELZ2
         ELSE
         DELZ=DELZ4
         END IF
         END IF

C---
C   Update the adaptive buffer management variable Z for the next
C   comming sample
C---

         Z=Z+DELZ
         IF(Z.LT.ZMIN) Z=ZMIN
         IF(Z.GT.ZMAX) Z=ZMAX

C---
C   Return to main program
C---

100      RETURN
         END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       CMPQNT
C
C PURPOSE:
C
C       To calculate the output factors of a uniform quantizer
C       for a given quantization range
C
C MEDIA:
C
C       ING. UTLPGM]CMPQNT. FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    NLEV    I*4     Total number of quantization levels
C               XMAX    R*4     Overload factor of the quantizer
C
C       From:-  YQ      R*4     An array of quantizer output factors
C
C IMPORTANT VARIABLES:
C
C       DELX    R*4     quantization step-size
C
C
C COMMON BLOCKS:
C
C       None
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
```

```
C PROGRAM SIZE AND SPEED:
C
C       Small and fast
C
C AUTHOR:
C
C       P. KABAL; modified by NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1990.
C
C MAINTRAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------


        SUBROUTINE CMPQNT(NLEV,XMAX,YQ)

C---
C   Initialize variables
C---
        DIMENSION YQ(NLEV)


C---
C   Calculate the quantization step-size with XMAX as quantization
C   range
C---
        DELX=2.0*XMAX/FLOAT(NLEV)

C---
C   Calculate the quantizer output factors
C---
        XOFFS=XMAX+0.5*DELX
        DO 100 I=1,NLEV
        YQ(I)=FLOAT(I)*DELX-XOFFS
100     CONTINUE

C---
C   Return to main program
C---
        RETURN
        END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       ENTROPY
C
C PURPOSE:
C
C       To calculate the entropy of the quantizer output
C
C MEDIA:
C
C       [NG.UTLPGM]ENTROPY.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    NLEV    I*4     Total number of quantization levels
C               PROB    R*4     An array of probability of occurance
C                               of each quantization level
C
C       From:-  ETP     R*4     Entropy of the quantizer output
C
C IMPORTANT VARIABLES:
C
C       None
C
C COMMON BLOCKS:
C
C       None
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       NOne
C
C SPECIAL TECHNIQUES:
C
C       None
C
```

```
C PROGRAM SIZE AND SPEED:
C
C       Small and fast
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------


        SUBROUTINE ENTROPY(NLEV,PROB,ETP)

C---
C   Specify dimension of PROB
C---

        DIMENSION PROB(NLEV)

C---
C   Calculate the entropy of the quantizer output
C---

        ETP=0.0
        DO 100 I=1,NLEV
        IF(PROB(I).NE.0.0) THEN
        ETP=ETP+(PROB(I)*ALOG10(1/PROB(I)))
        END IF
100     CONTINUE
        ETP=ETP/ALOG10(2.0)

C---
C   Return to calling program
C---

        RETURN
        END
```

```
C-------------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       QUANT
C
C PURPOSE:
C
C       To determine which quantization slot the input sample
C       falls in for a given quantizer
C
C MEDIA:
C
C       [NG.UTLPGM]QUANT.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       TO:-    NLEV    I*4     Total number of quantization levels
C               X       R*4     An input sample
C               YQ      R*4     An array of quantizer output levels
C
C       Form:-  L       I*4     The quantization level number
C
C IMPORTANT VARIABLES:
C
C       None
C
C COMMON BLOCKS:
C
C       None
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
```

```
C
C PROGRAM SIZE AND SPEED:
C
C       Small and fast
C
C AUTHOR:
C
C       P. KABAL; modified by NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------


        SUBROUTINE QUANT(X,L,YQ,NLEV)

C---
C   Specify the dimension of YQ
C---
        DIMENSION YQ(NLEV)

C---
C   Determine the quantization level number L
C---
        IL=1
        IU=NLEV
100     I=(IL+IU)/2
        IF(X.GT.YQ(I)) GO TO 200
        IU=I
        GO TO 300
200     IL=I
300     IF(IU.GT.IL+1) GO TO 100
        IF(X.GE.0.5*(YQ(IL)+YQ(IL+1))) IL=IL+1
        L=IL

C---
C   Return to calling program
C---
        RETURN
        END
```

```
      SUBROUTINE SEARJOC(N,IN,X,XSTEP,FUNCT,CONV,MAXM,FM1,F0)
C                                                      79/11/07 P. KABAL
C  THIS SUBROUTINE MINIMIZES A FUNCTION USING A PATTERN SEARCH
C  TECHNIQUE.
C
C  N       - NUMBER OF VARIABLES, LESS THAN 50
C  X       - VECTOR OF VARIABLES, INITIALLY CONTAINING THE STARTING
C            VALUES.  ON RETURN, X CONTAINS THE OPTIMUM VALUES FOR THE
C            VARIABLES.
C  XSTEP   - VECTOR CONTAINING THE INITIAL STEP SIZES FOR EACH OF THE
C            VARIABLES
C  FUNCT   - NAME OF THE EXTERNAL SUBROUTINE USED TO CALCULATE THE
C            OBJECTIVE FUNCTION.  THE CALLING SEQUENCE FOR FUNCT IS
C            CALL FUNCT(N,IN,X,FM1,F) WHERE F IS THE VALUE OF THE OBJECT-
C            IVE FUNCTION FOR THE VECTOR OF N VARIABLES X.  THE SUBROUTINE
C            NAME MUST APPEAR IN AN EXTERNAL STATEMENT IN THE CALLING
C            ROUTINE.  THE SUBROUTINE FUNCT MUST NOT ALTER THE VALUES
C            IN THE VECTOR X OR THE VALUE OF N.
C  CONV    - FRACTION OF THE STEP SIZE USED AS A CONVERGENCE CRITERION
C  MAXM    - MAXIMUM NUMBER OF PATTERN MOVES PERMITTED
C  F0      - OPTIMUM VALUE FOR THE OBJECTIVE FUNCTION
C
C
      PARAMETER MXVAR=49
      DIMENSION X(N),XSTEP(N),DELX(MXVAR),X0(MXVAR),FM1(IN)
C
C
C  SET UP THE FIRST BASE POINT AND INITIALIZE THE STEP SIZE VECTOR
      DO 80 I=1,N
        X0(I)=X(I)
        DELX(I)=XSTEP(I)
80    CONTINUE
      KK=0
      TEST=ABS(CONV*DELX(1))
C
C  INITIALIZE F0
      CALL FUNCT(N,IN,X,FM1,F0)
C
C  SEARCH LOOP OVER THE VARIABLES IN X
100   NFAIL=0
      DO 250 I=1,N
        XT=X(I)
C
C  INCREMENT X(I) (IN THE DIRECTION OF THE LAST SUCCESSFUL STEP)
        X(I)=XT+DELX(I)
        CALL FUNCT(N,IN,X,FM1,FI)
        IF(FI.LE.F0) GO TO 200
        F0=FI
        GO TO 250
C
C  DECREMENT X(I)
200     X(I)=XT-DELX(I)
        CALL FUNCT(N,IN,X,FM1,FD)
        IF(FD.LE.F0) GO TO 220
        F0=FD
        GO TO 240
C
C  UNSUCCESSFUL IN EITHER DIRECTION
```

```
220         NFAIL=NFAIL+1
            X(I)=XT
            IF(FD.LE.FI) GO TO 250
C
C  REVERSE THE DIRECTION OF THE STEP
240         DELX(I)=-DELX(I)
C
250       CONTINUE
          PRINT *,F0
C
          IF(NFAIL.LT.N) GO TO 320
C
C  ALL STEPS UNSUCCESSFUL, TEST FOR CONVERGENCE
          IF(ABS(DELX(1)).LT.TEST) GO TO 400
C
C  NOT CONVERGED, HALVE THE STEP SIZE
          DO 310 I=1,N
            DELX(I)=0.5*DELX(I)
310       CONTINUE
          GO TO 100
C
320       KK=KK+1
          IF(KK.GE.MAXM) GO TO 400
C
C  MAKE A PATTERN MOVE AND ESTABLISH A NEW BASE POINT
          DO 350 I=1,N
            XT=X(I)
            X(I)=XT+(XT-X0(I))
            X0(I)=XT
350       CONTINUE
C
          CALL FUNCT(N,IN,X,FM1,FI)
          IF(FI.LE.F0) GO TO 360
          F0=FI
          GO TO 100
C
C  PATTERN MOVE UNSUCCESSFUL, CONTINUE SEARCH FROM THE BASE POINT
360       DO 370 I=1,N
            X(I)=X0(I)
370       CONTINUE
          GO TO 100
C
400       RETURN
C
          END
```

```
C----------------McGill-Computer-Vision-and-Graphics-Laboratory----------
C
C
C MODULE NAME:
C
C       SLI_SNR
C
C PURPOSE:
C
C       To calculate the sliding SNR(SLISNR) of the coder
C
C MEDIA:
C
C       ING.UTLPGM]SLISNR.FOR
C
C KEYWORDS:
C
C       VAX-11
C
C USAGE:
C
C       Supply the parameters as stated below
C
C SPECIAL REQUIREMENTS:
C
C       None
C
C LANGUAGE:
C
C       FORTRAN IV PLUS
C
C PARAMETERS:
C
C       To:-    IBLK    I*4     Total number of input vectors
C               IBLKL   I*4     Vector dimension
C               NLEV    I*4     Total number of quantization levels
C               NNS     I*4     Total number of speech samples
C                               simulated
C               X       R*4     An array of input speech samples
C               YQR     R*4     An array of decoded outputs at
C                               receiver
C
C       From:-  SEGSNR  R*4     Segmental SNR
C               SLISNR  R*$     Sliding SNR
C
C IMPORTANT VARIABLES:
C
C       None
C
C COMMON BLOCKS:
C
C       None
C
C ROUTINES REQUIRED:
C
C       None
C
C SIDE EFFECTS:
C
```

```
C       None
C
C SPECIAL TECHNIQUES:
C
C       None
C
C PROGRAM SIZE AND SPEED:
C
C       Small and speed depends on the total number of speech
C       samples simulated
C
C AUTHOR:
C
C       NG S.C.
C
C DATE CREATED:
C
C       Oct. 30, 1980.
C
C MAINTAINED BY:
C
C       NG S.C.
C
C UPDATES:
C
C       None yet
C
C---------------McGill-Computer-Vision-and-Graphics-Laboratory----------------



      SUBROUTINE SLI_SNR(X,YQR,IBLK,IBLKL,NLEV,NNS,SLISNR,SEGSNR)

C---
C   Specify the dimensions of X, YQR & SLISNR
C---

      DIMENSION
     1          X(NNS),
     1          YQR(NNS),
     1          SLISNR(IBLK)

C---
C   Initialize IBOFFSET SEGSNR & SLISNR
C---

      IBOFFSET=0
      SEGSNR=0
      DO 100 I=1,IBLK
      SLISNR(I)=0
100   CONTINUE

C---
C   Calculate the sliding SNR and segmental SNR
C---
```

```
      DO 120 I=1,IBLK
      XX2=0
      YY2=0
      IBOFFSET=(I-1)*IBLKL
      DO 110 J=1,IBLKL
      XX2=XX2+X(J+IBOFFSET)**2
      YY2=YY2+(YQR(J+IBOFFSET)-X(J+IBOFFSET))**2
110   CONTINUE
      SLISNR(I)=10*ALOG10(1+(XX2/YY2))
      SEGSNR=SEGSNR+SLISNR(I)
120   CONTINUE
      SEGSNR=SEGSNR/(IBLK*10.)
      SEGSNR=(10.**SEGSNR)-1
      SEGSNR=10.*ALOG10(SEGSNR)

C---
C   Return to main program
C---

      RETURN
      END
```