Abstract

Title of Thesis: **EEG data compression**

Yun-chu Wu, Master of Science in Biomedical Engineering, 1991.

Thesis directed by: Dr. Stanley Reisman

Electrical Engineering Department, NJIT, Newark, NJ

Dr. Elizabeth Pinkhasov

Neuroscience Department

UMDNJ-New Jersey Medical School, Newark, NJ.

Dr.David Kristol

Biomedical Engineering Department, NJIT, Newark, NJ

This paper presents two different ways to compress EEG data- – direct data compression and a data transformation technique. The Adaptive Delta modulation and Huffman coding are used in the former method to predict or interpolate the data. Linear orthognal transformation algorithms are used in the latter method to detect and reduce the redundancies of the data by analyzing the spectral and energy distribution. Each method is implemented by programming the computer. The experimental results of their efficiencies and errors with different requirements and under different situations are compared and discussed. By comparing the EEG data compression degree and normalized square error, the paper shows that the adaptive delta coding followed by Huffman coding is the best way to compress the EEG data.

EEG Data Compression

by
Yun-Chu Wu

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Biomedical Engineering
1991

Approval Sheet

Title of Thesis: EEG Data Compression

Name of Candidate: YunChu Wu

Master of Science in Biomedical Engineering

Thesis and Abstract Approved: ______________________ __________

Dr. Stanley Reisman Date

Electrical Engineering Department, NJIT

______________________ __________

Dr. Elizabeth Pinkhasov Date

Neurosciences Department, UMDNJ

______________________ __________

Dr. David Kristol Date

Biomedical Engineering Department, NJIT

Vita

Name: Yun-Chu Wu

Degree and date to be conferred: MSBME, 1991

Secondary education: The Si-Xi senior high school

Collegiate institutions attended	Dates	Degree	Date of Degree
Shanghai University of Science and Technology	1981-87	BSBME	1987
New Jersey Institute of Technology	1989-91	MSBME	1991

Major: Biomedical Engineering

To dear dad and mom

Acknowledgement

Many people have assisted me during the course of this work. There are too many such people to thank individually. However, I would like to acknowledge my advisor Dr. Stanley Reisman whose inspiration and guidance benefitted me significantly. Without his support this work could not have been finished.

A very special thanks is extended to Dr. Swany Laxminarayan for his encouragement and suggestions. His help in regard of Walsh Transformation System, both the reference materials and software application, and comments on the research work are valuable.

I would like to thank Dr. Elizabeth Pinkhasov for her support in obtaining the EEG data in UMDNJ.

Also, I would like to thank my parents and Mr.Shing Lee for their support and deep loves.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sixty years ago, Han Berger recorded the Electrical potential at the surface of the scalp given from the cortical and subcortical layers of the brain[1]. It was known as the electroencephalogram or more simply, the EEG. The recorded potentials are represented by many fluctuating waves whose frequency may vary from 0.1 to 30Hz. These frequencies are broken down into the following bands or ranges: Delta (below 3.5Hz); Theta (4-7.5Hz); Alpha (8-13Hz); Beta (above 13Hz). The clinical electroencephalographer correlates central nervous system functions as well as dysfunctions and diseases with certain patterns of the EEG on an empirical basis[1]. Thus EEG is of fundamental importance in diagnosing certain brain defects and mental disorders, and in characterizing sleep.

Since a large number of channels (16-24) is required to be recorded simultaneously from all electrodes on the scalp and a long term recording is required to monitor the EEG wave, the efficiency of storing and recording EEG technique is significant[2]. Electronic amplifiers and pen-writers have long been part of the EEG laboratory. In the 1950s, some laboratories began to add tape recording and telemetry systems for longer term recording , and then video recorders have been used to capture both the movements and the EEGs of epileptic patients. More recently, some laboratories and clinics have begun to use the computer to analyze the electroencephalogram and it became more and more important to store the EEG in the computer rather than on paper or video tape[10]. Since the computer has limited storage capacity, the long-term and large numbers of channels of EEG recording in a digital format must be compressed.

The existing methods for data compression can be classified into three categories (1). direct data handling methods; (2). transformation methods; (3). parameter extraction methods[3]. Usually in the medical data compression, the first two methods are used. In ECG data compression, many algorithms based on direct data handling have been proposed in the literature[5],[6],[8],[9],[11],[13]. The amplitude zero time epoch coding (AZTEC) algorithm[5],[6] has great data reduction properties, but it is not visually acceptable to a cardiologist. The AZTEC technique converts raw ECG sample points into plateaus and slopes. The stored values for each plateau are the amplitude value of the line and its length. The production of an AZTEC slope starts when the number of samples needed to form a plateau is less than three. The slope is saved whenever a plateau of three samples or more can be formed. The stored values for the slope are the duration (number of samples of the slope) and the final elevation (amplitude of last sample point). The coordinate reduction time encoding system (CORTES)[7],[8] which is a hybrid of the turning point and AZTEC algorithms, is effective in off- line data reduction, but it can not be applied in real-time ECG data compression. Delta code algorithm[9] is based on the fact that the difference in amplitude between successive samples is typically smaller than the amplitude of the samples themselves. A modified technique called "delta coding with threshold" for compression of three-lead (X, Y, Z) ECG signal is proposed in [13]. Whenever the absolute value of the difference between adjacent pair samples in any of the three ECG lead signals exceeds a preset threshold, data are saved. Otherwise data are considered redundant and, hence, eliminated. The retained data comprises the amplitude difference, between the pair samples at the time slot for each of the three-lead ECG signals, along with the elapsed since the last saved data. The transform compression methods which used in ECG data compression are Karhuner-Loeve transform[16], the fast Walsh transforms[17] and the Fourier Descriptor[3]. All the transform methods are similar. For example, the FFT technique transfer the data from time domain to frequency domain. According the distribution of frequency spectrum of the signal, the data can be compressed by storing a fraction of a Fourier spectrum to reconstruct the signal. The prediction-interpolation method, variable length encoding method and the transformation methods can also be applied in Biomedical Images[12]. As examples, for blood cell analysis and X-ray images, two-dimensional transformations are generally used with procedures, such as thresholding, variable length encoding and prediction- interpolation. In particular, fast Fourier (FFT) and fast Walsh transforms (FWT) can be used; while for nuclear medicine images different transformations (Fourier, Hadamard, Haar) in fast form with thresholding are employed[14],[15].

The main goal of any compression technique is to achieve maximum data volume reduction while preserving the significant signal morphology features upon reconstruction[4]. In this project, we tried several methods to compress the EEG data. The techniques we used are (1) direct data compression; (2) transformation methods. The direct data compression techniques rely on utilizing prediction or interpolating algorithms. A prediction algorithm utilizes a *priori* knowledge of some previous samples, while an interpolation algorithm employs a *priori* knowledge of both previous and future samples. These techniques attempt to reduced redundancy in a data sequence by examining a successive number of neighboring samples. The direct data compression methods we used here include two methods: the adaptive delta coding and the Huffman coding, which will be discussed separately in chapter 3 and chapter 4. In chapter 3 we will apply the adaptive delta modulation to compress the EEG data and in chapter 4 we will apply the Huffman coding (variable length coding) to compress the EEG data. In ADM we store the codeword of difference between adjacent pair samples instead of the data themselves. In Huffman coding we assign every data a different codeword, the length of codeword depends on the relative frequency of the data in the data file. Unlike the direct data compression, most of the transformation compression techniques which have been employed in data compression involve preprocessing the input signal by means of a linear orthogonal transformation and reducing the amount of data needed to adequately represent the original signal.

Transformation compression methods mainly utilize spectral and energy distribution analysis for detecting redundancies. Many discrete orthogonal transforms have been employed in digital signal representation such as Fourier (FT), Walsh (WT), Cosine (CT) etc. In the next chapter we will discuss how to use the Walsh transformation to reduce the data.

The A/D converter used in this project is the Metrabyte DASH-16[18], which is a single board with 16 single ended / 8 differential analog input channels. It is used to perform control and data acquisition for the PC/XT/AT and compatible computers (Fig 1.1). After A/D converting we obtain a packed data file formed by the DASH-16 software. Thus the real data file was obtained by unpacking the file directly from the A/D converter. Sampled data is 12 bits long including one bit that indicates whether the attached packet is positive or negative.

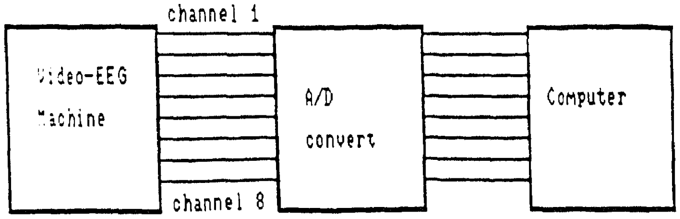


Figure 1.1: System diagram

Implementation of the EEG data compression is in the Unix system. All programs were written in Pascal language. The experimental data were acquired from the video tapes of seizuré patients admitted to the intensive Video-EEG unit of the Neurology department of UMDNJ. The video tapes also were recorded by this Video-EEG machine. This machine can record the real-time EEG data directly from the patient and at the same time we can see the real-time EEG on the screen of the machine, as well as providing at output channel to the A/D converter and computer.

Chapter 2

Walsh transformation

2.1 Walsh transform data-compression algorithm

The orthogonal transforms in data compression enables signal representations which can be used to reduce the amount of redundant information[19],[20]. If a discrete signal consists of N sampled values, then it can be looked upon as being a point in an N-dimensional space. Each sampled value is then a component of the (N*1) data vector X which represents the signal in this space. For a more efficient representation, we obtain an orthogonal transform of X which results in $Y = TX$. Our objective is to select a subset of M components of Y, where M is less than N. The remaining (N-M) components can be set equal to zero and the signal is reconstructed using the M retained components of Y. Thus the amount of data is reduces from N to M.

Let T denote the desired orthogonal transform, whose transpose T' is given by $T' = [\phi_1 \phi_2 ... \phi_N]$ where ϕ_i is an (N*1) vector and the basis vectors ϕ_m are orthonormal.

$$(\phi_i', \phi_j) = \begin{cases} 1 & \text{if,} i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

For each vector X belonging to a given class of data vectors, we obtain $Y = TX$ where $X' = [x_1 x_2 ... x_n]$, and $Y' = [y_1 y_2 ... y_N]$. Thus $TT' = 1$ and $X = T'Y = [\phi_1 \phi_2 ... \phi_N]Y$, which yields $X = y_1\phi_1 + y_2\phi_2 + ... + y_N\phi_N = \sum_{i=1}^{N} y_i\phi_i$. We wish to retain a subset $y_1, y_2, ..., y_n$ of the components of Y and yet estimate X. This can be done by replacing the remaining (N-M) component of Y with zero to obtain

$$\tilde{X}(M) = \sum_{i=1}^{M} y_i\phi_i \tag{2.2}$$

where $\tilde{X}(M)$ denotes the estimate of X.

The Walsh transformation data-compression method employs Walsh functions (Fig-2.2)[17] as the orthogonal basis set. In this method a Walsh-coefficient series is used to represent the sequence obtained by sampling the signal. The Walsh-coefficient series representation of an N-element long sampled signal f(n) can be expressed as

$$f(n) = \sum_{m=0}^{N-1} a_m\phi_m(n) \tag{2.3}$$

where ϕ_m is the m-th discrete Walsh function. The data compression system described above by using the orthogonal representation of the sampled signals is shown in Fig-2.1[17]. To exactly reconstruct f(n), all N a_m coefficients must be computed and transmitted. The data can be compressed by computing and transmitting less than the entire set of N a_m. In the Walsh transform method this technique used to compress the data consists of using only a fraction of a Walsh spectrum to reconstruct the signal. In doing so, the lowest sequence fraction of the spectrum is retained, while the remaining portion of the spectrum is set to zero. Walsh functions are a set of periodic orthogonal functions which are rectangular in nature (Fig-2.2)[17]. The periodicity of the Walsh function is described in terms of square waves of different sequences, and because the

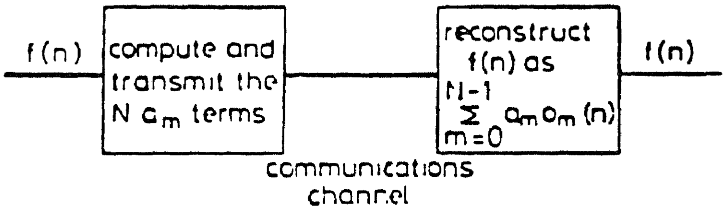

f(n)
compute and transmit the N a_m terms
communications channel
reconstruct f(n) as $\sum_{m=0}^{N-1} a_m \phi_m(n)$
f(n)

Figure 2.1: Walsh transform data-compression algorithm

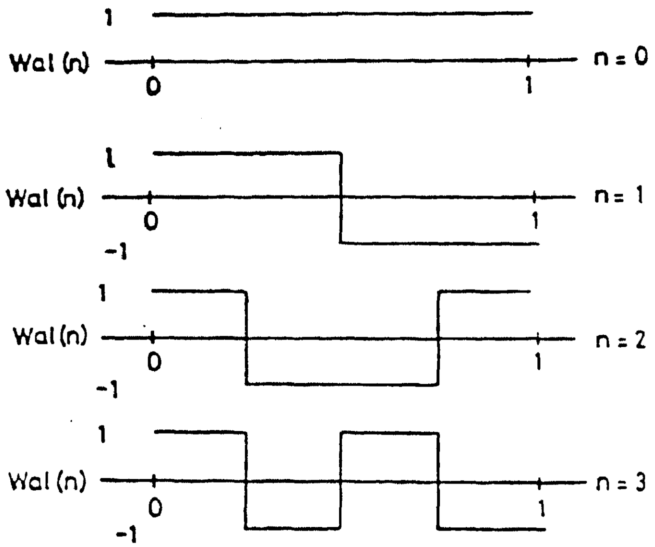


Figure 2.2: Walsh Function

values of the Walsh function are either +1 or -1, it is possible to compute the Walsh transform representation of a sampled signal by adding and subtracting. Such a reduction in computation time would theoretically allows a computer currently programmed to do Fourier analysis of one channel of EEG data to instead do Walsh analysis of several channels in the same time[21]. The Walsh transform F(m) of an N-point discrete signal f(n) can be computed as

$$F(m) = \sum_{n=0}^{N-1} f(n)Wal(m,n) \tag{2.4}$$

2.2 Walsh transformation in EEG data compression

When applying the Walsh transformation to EEG, we first transfer the EEG from time domain to sequency domain. After obtaining the Walsh coefficients of the EEG wave, we set a high sequence fraction of the spectrum to zero. Therefore we can reconstruct the EEG waveform by the compressed spectrum. In this project we have tried three different degrees of compression. The first was to set the highest 1/5 spectrum to zero; the second was to set the highest 3/5 spectrum to zero; the last was to set the highest 17/25 spectrum to zero. Fig 2.3 shows two original EEG waveforms. The plots of the reconstructed EEG waveforms are shown in Fig-2.4, Fig 2.5 and Fig-2.6. Here x axis is the number of sampled data and y axis is the amplitude of EEG wave. These three sets of reconstructed EEG waveforms are obtained from 4/5, 2/5, and 8/25 respectively, of the Walsh spectrum. As we can see from the reconstructed waveforms (Fig-2.4, Fig- 2.5 and Fig-2.6) the more compression that was employed, the more error was obtained. When the data were compressed from 20% to 68% the error increase from 18.4 to 26.5 (we will discuss this error measure further in chapter 6). Also the high frequency signals are cut off. This compression scheme is therefore not suitable for the abnormal EEG waveforms with high frequency spikes.

In order to assess the performance of the compression methods, in addition to visual comparison, the normalized square error is employed. It represents a measure of "goodness" of the reconstructed waveforms. The normalized mean square error between the original and reconstructed EEG waveform is computed as:

$$\frac{\sum_{i=0}^{N} \sqrt{(x_i - \hat{x}_i)^2}}{N} \tag{2.5}$$

where x and $\hat{x}$ are samples of the original and reconstructed data. Thus we can use this error to compare the error between the original and reconstructed waveforms for a specified compression ratio among the different compression methods. From table 6.1 we can see the errors in the Walsh transform data compression method.

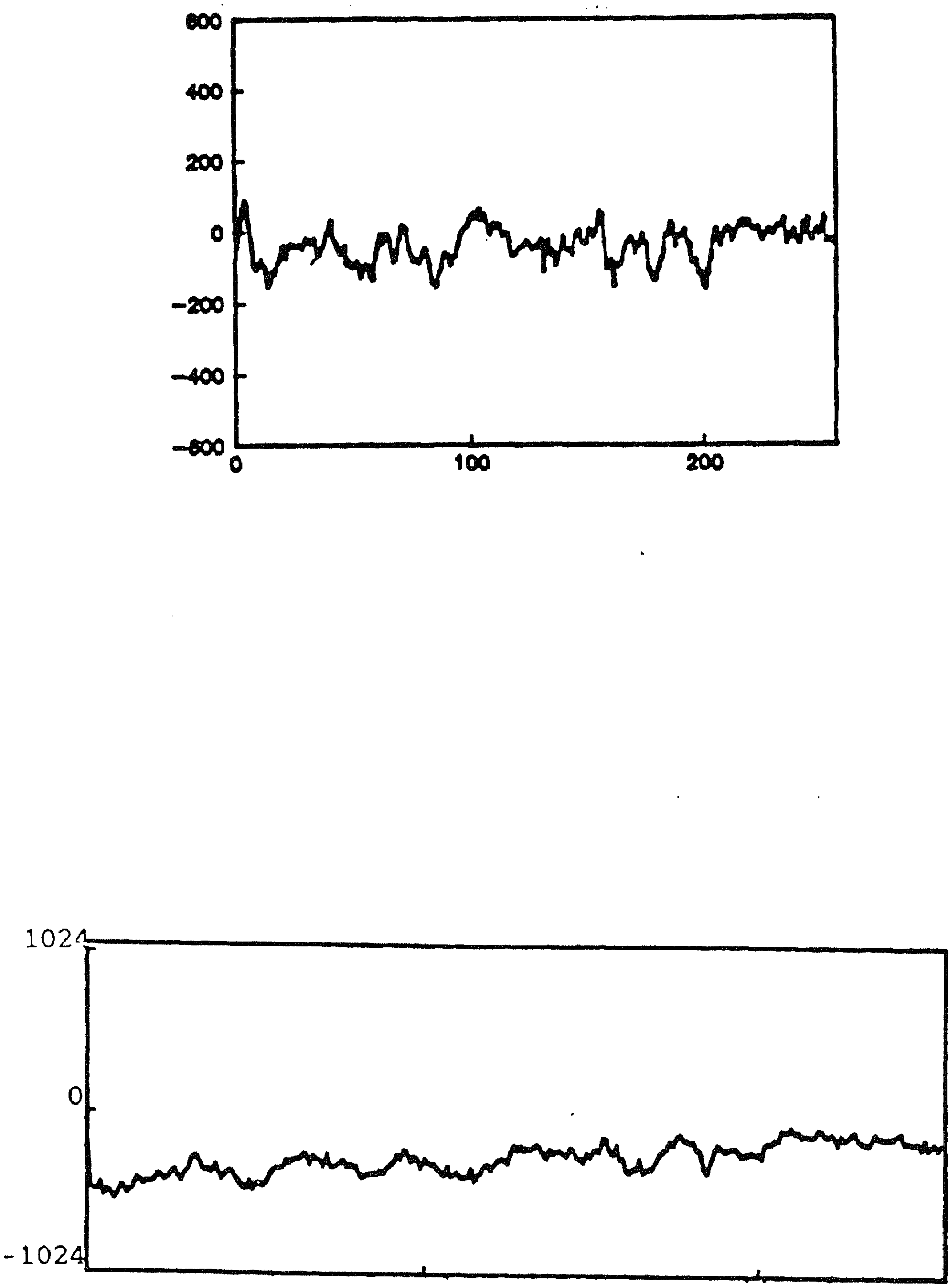


Figure 2.3: Original EEG waveforms

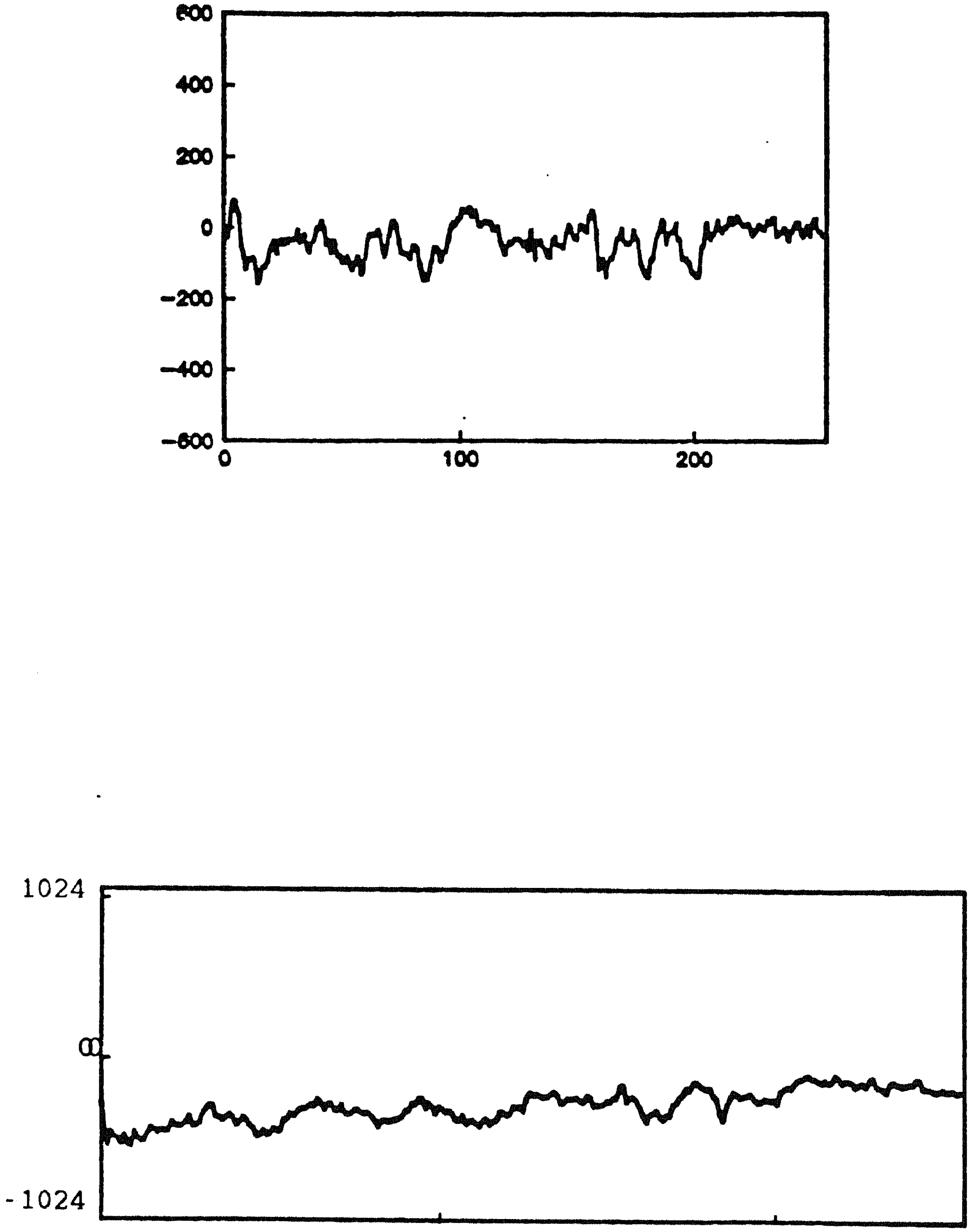
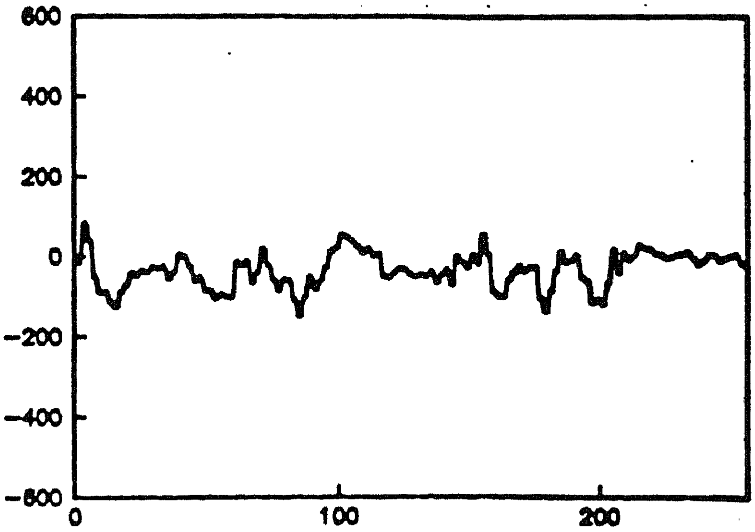


Figure 2.4: Reconstructed waveforms obtained using the Walsh spectrum CR=0.8



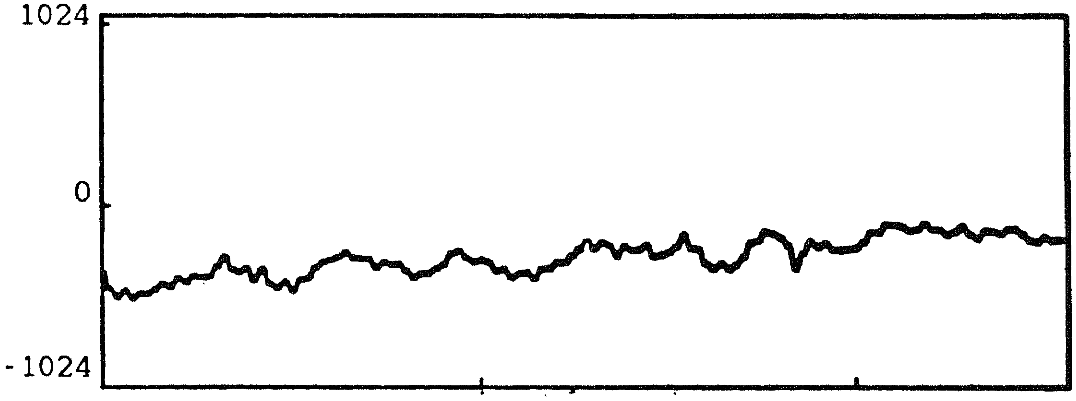


Figure 2.5: Reconstructed waveforms obtained using the Walsh spectrum CR=0.4

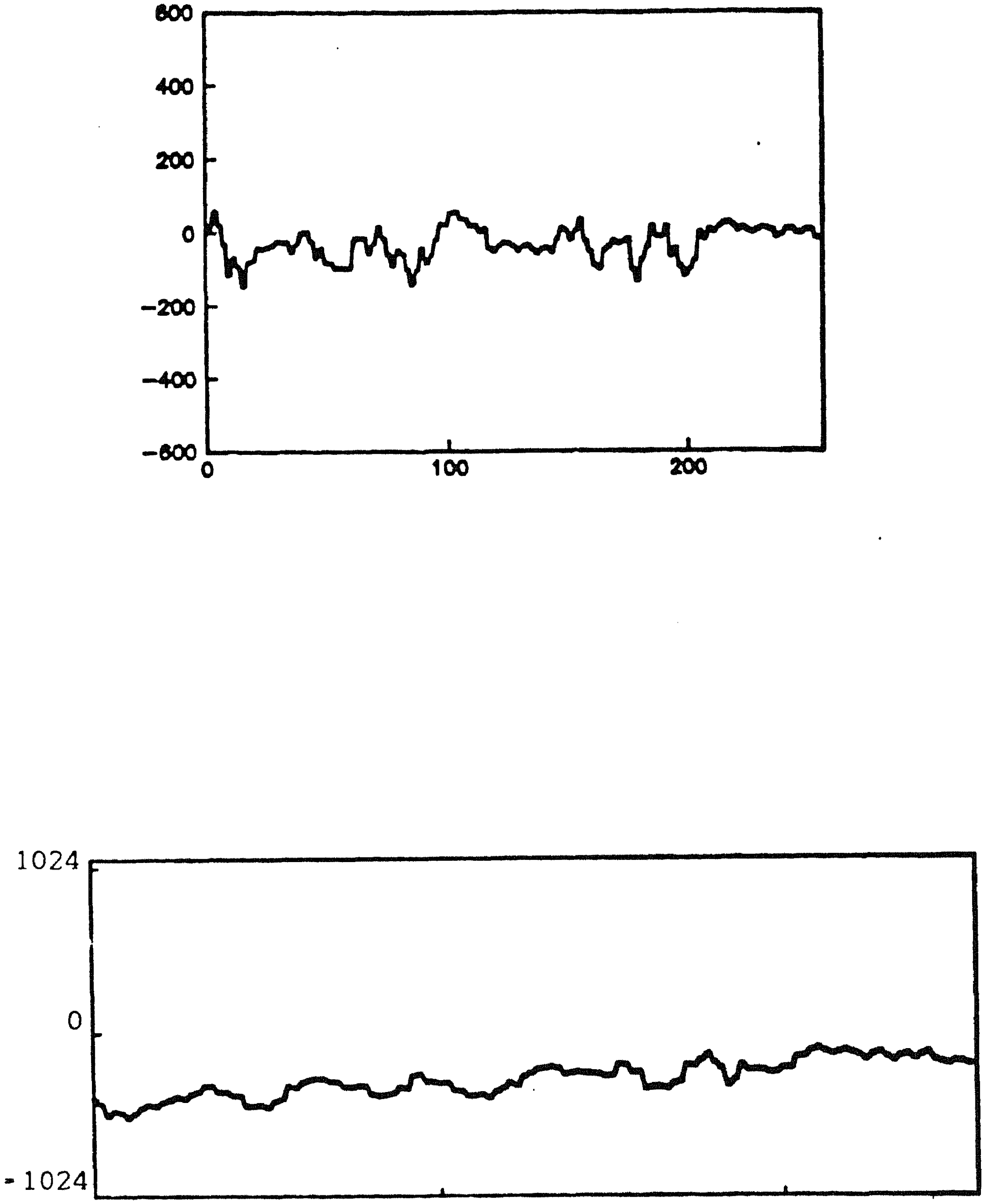


Figure 2.6: Reconstructed waveforms obtained using the Walsh spectrum CR=0.32

Chapter 3

Adaptive Delta coding method

3.1 Delta modulation

3.1.1 linear delta modulation

The schematic diagram of the basic delta modulation system is shown in Fig-3.1[22]. The delta modulator acts as an analog to digital converter. It consists of an analog input signal x(t) and a binary output signal L(t). The relationship between x(t) and L(t) is such that L(t) is a binary representation of x(t), where the rate of occurrence of each binary pulse is directly proportional to the instantaneous slope of x(t). If the slope of the input signal x(t) is positive, the output waveform L(t) will have more positive pulses than negative ones. The situation is reversed, when x(t) has a negative slope (Fig-3.2)[22].

The difference between x(t) and y(t) is the error signal e(t). If e(t) is larger than zero, a positive pulse will be produced at the output of the encoder. When this pulse is integrated, y(t) is increased by a positive step. This increase in y(t) will be subtracted from x(t) and a change in the magnitude of the error signal occurs. If the error has not become negative by the next clock instant, the output of the encoder will be a positive pulse again. As long as e(t) is

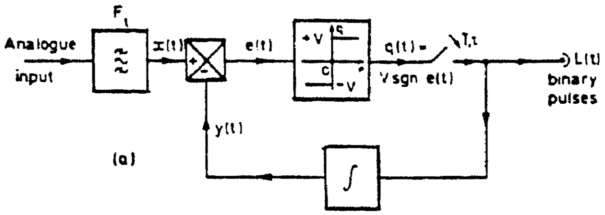


Figure 3.1: Basic delta modulator system

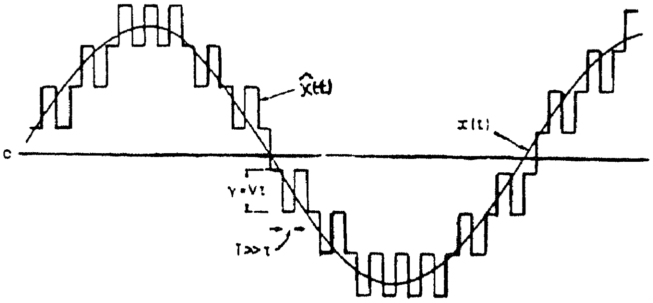


Figure 3.2: Linear D.M. waveform when the encoder is tracking the input signal

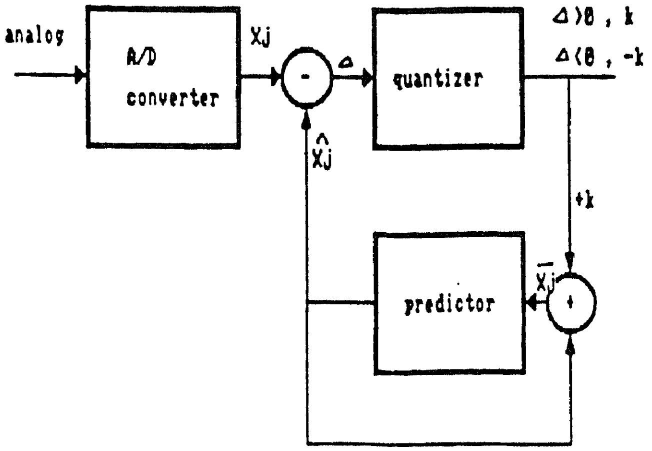


Figure 3.3: The principle of digital delta modulation

larger than zero at successive clock instants, a sequence of positive pulses will be produced. Eventually y(t) will become greater than x(t), at which point e(t) becomes negative and a negative pulse will occur at the output of the encoder.

3.1.2 Digital delta modulation

The principle of digital delta modulation which is shown in Fig-3.3[23] is to use +1 or -1 to represent the difference between each sample value and its predecessor. The predictor is used to predict the current value x_j by using the previous x_{j-1}. After subtraction between the current value x_j and the previous x_{j-1}, a difference δ is obtained. When this difference passes the quantizer the receiver will receive +k or -k depending on whether $\delta > 0$ or $\delta < 0$. This value also feeds back to the predictor to adjust the prediction $\hat{x}_j$ by adding or subtracting k.

3.2 Adaptive delta modulation

In using delta modulation if the step size is not large enough to track the slope of wave, it will cause slope overload (Fig 3.4)[23]. Slope overload can be minimized by increasing the step size, but as the step size is increased the quantization noise will increase. Thus it is desirable to make the system adaptive. Adaptive delta coding modulates the waveform by applying varied step size quantization in delta modulation[23]. In ADM, the DM encoder feeds back a range of step sizes depending on the polarity of the difference between present and the previous values. If we suppose the minimum step size is r, then the step height will change consecutively from r to rk_2, rk_3, ... rk_n, (where $k_2 < k_3 < ... < k_n$) depending on the slope of the waveform. The step size incrementally decreases when the error changes sign. In Fig 3.5, we can see while the overload slope occurs the larger step size is used and while the slope decreases the step size decreases as well. Thus as we see the ADM attempts to track the slope by increasing or decreasing the step size and avoid the slope overload in DM.

3.3 ADM in EEG data compression

The EEG waveform is very complex. It consists of two kinds of patterns[1]. The simple patterns of waves can be classified as delta ($frequency < 4Hz$), theta($4Hz < f < 8Hz$), alpha ($8 < f < 13$) and beta ($f > 13$). The complex patterns can be placed into two classes. The first consists of patterns of fluctuations which may be considered as combinations of two or more simple patterns. The second consists of specific wave shapes. These specific waveforms

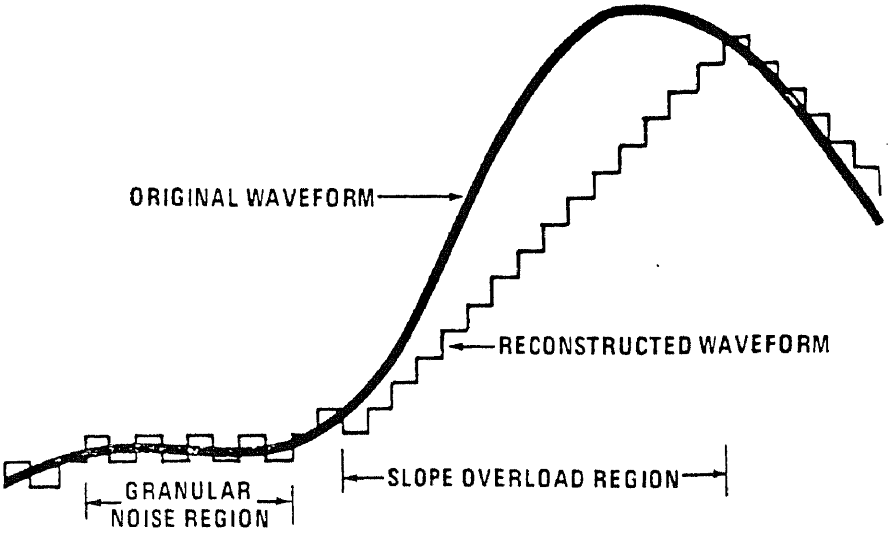


Figure 3.4: Overload in delta modulation

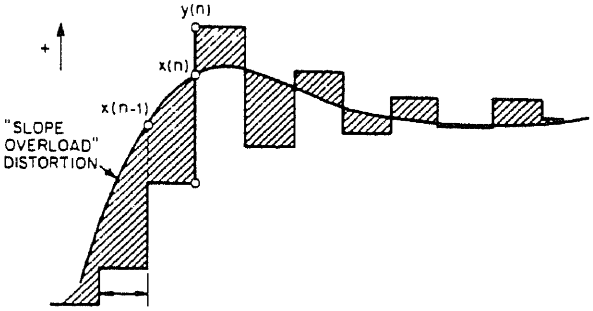


Figure 3.5: Avoid the slope overload in ADM

are fluctuations of amplitude which have a characteristic shape. Among the specific waveforms there are the K-complex (the maximum voltage is very variable but usually about 200μv), lambda waves (the amplitude is usually less than 50μv), spike (a wave transient clearly distinguished from background activity with pointed peak at conventional paper speeds, its amplitude is very variable), spike and wave rhythm (the amplitude may attain 1000 μv), etc (Fig 3.6). Thus, the differences among sampled data are very variable, and can not be replaced by a fixed value. The most effective way is to use the adaptive delta modulation to code the difference between two adjacent data values. Although the range of sampled data is from -2048 to 2048, the difference between the two adjacent samples varies from -120 to 120 in all files examined during this project. We applied the ADM to EEG data compression in two methods. First, we used 5 bits (one bit indicates positive sign or negative sign), and second, we used 4 bits (also one bit indicates positive sign or negative sign). Tables 3.1 and 3.2 show these two ways. In table 3.1, the minimum step size δ equals 8, and k_2, k_3, ..., k_{15} are 2,3, ..., 15, respectively. If the difference is between $k_i\delta + \frac{\delta}{2}$ and $k_{i+1}\delta$ then the code is k_{i+1}. If the difference is between $k_i\delta$ and $k_i\delta + \frac{\delta}{2}$ then the code is k_i. Table 3.2 is the same as table 3.1 but δ equals 16 and k_2, k_3, ..., k_7 are 2,3, ..., 7, respectively.

Fig-3.7, Fig-3.9 and Fig-3.13 are three original EEG waveforms. Fig-3.8, Fig-3.11 and Fig-3.14 are the reconstructed waveforms from the code files which store the coded difference of adjacent data by using the algorithm in table 3.1. Similarly, Fig-3.9, Fig-3.12 and Fig-3.15 are the reconstructed waveforms from the code files which store the coded difference between adjacent data based on table 3.2. The data is compressed from 12bits to 5 bits (table 3.1) and 4 bits (table 3.2), respectively.

As we can see from the reconstruction waveforms the errors are very small whether we use $\delta = 16$ or $\delta = 8$, the method of chapter 2 we again used to calculate the error and the value of 2.547 and 6.019 were obtained, respectively. We will discuss these errors further in chapter 6. Fig 3.7 is a normal EEG waveform while Fig 3.10 has some spikes. From the reconstructed waveforms of Fig 3.5 and Fig 3.8 shown in Fig 3.11 and Fig 3.12 we can see that the high frequencies are not cut off, because the reconstructed spikes in Fig 3.11 and Fig 3.12 are very visible. Also the slow rhythm in Fig 3.13 is not affected in the reconstructed waveforms (Fig 3.14 and Fig 3.15). Obviously the other components in the reconstructed waveforms such as alpha, beta, etc. will not be affected because their frequencies are lower than those of the spikes. If the higher frequency waveforms can be reconstructed well, then the lower frequency waveforms can be reconstructed well.

Table 3.1: Adapt Delta code of EEG data file $\delta = 8$

Difference	Code
$0 \rightarrow \pm 3$	0
$\pm 4 \rightarrow \pm 11$	± 1
$\pm 12 \rightarrow \pm 19$	± 2
$\pm 20 \rightarrow \pm 27$	± 3
$\pm 28 \rightarrow \pm 35$	± 4
$\pm 36 \rightarrow \pm 43$	± 5
$\pm 44 \rightarrow \pm 51$	± 6
$\pm 52 \rightarrow \pm 59$	± 7
$\pm 60 \rightarrow \pm 67$	± 8
$\pm 68 \rightarrow \pm 75$	± 9
$\pm 76 \rightarrow \pm 83$	± 10
$\pm 84 \rightarrow \pm 91$	± 11
$\pm 92 \rightarrow \pm 99$	± 12
$\pm 100 \rightarrow \pm 107$	± 13
$\pm 107 \rightarrow \pm 115$	± 14
$\pm 116 \rightarrow$	± 15

Table 3.2: Adapt Delta code of EEG data file $\delta = 16$

Difference	code
$0 \rightarrow \pm 7$	0
$\pm 8 \rightarrow \pm 23$	± 1
$\pm 24 \rightarrow \pm 39$	± 2
$\pm 40 \rightarrow \pm 55$	± 3
$\pm 56 \rightarrow \pm 71$	± 4
$\pm 72 \rightarrow \pm 87$	± 5
$\pm 88 \rightarrow \pm 103$	± 6
$\pm 104 \rightarrow \pm 120$	± 7

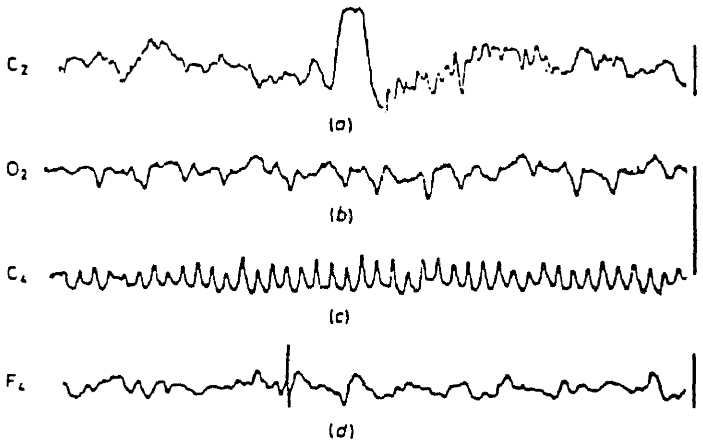


Figure 3.6: Examples of specific waveforms. (a) K-complex. (b) Lambda wave. (c) Mu rhythm (d) Spike

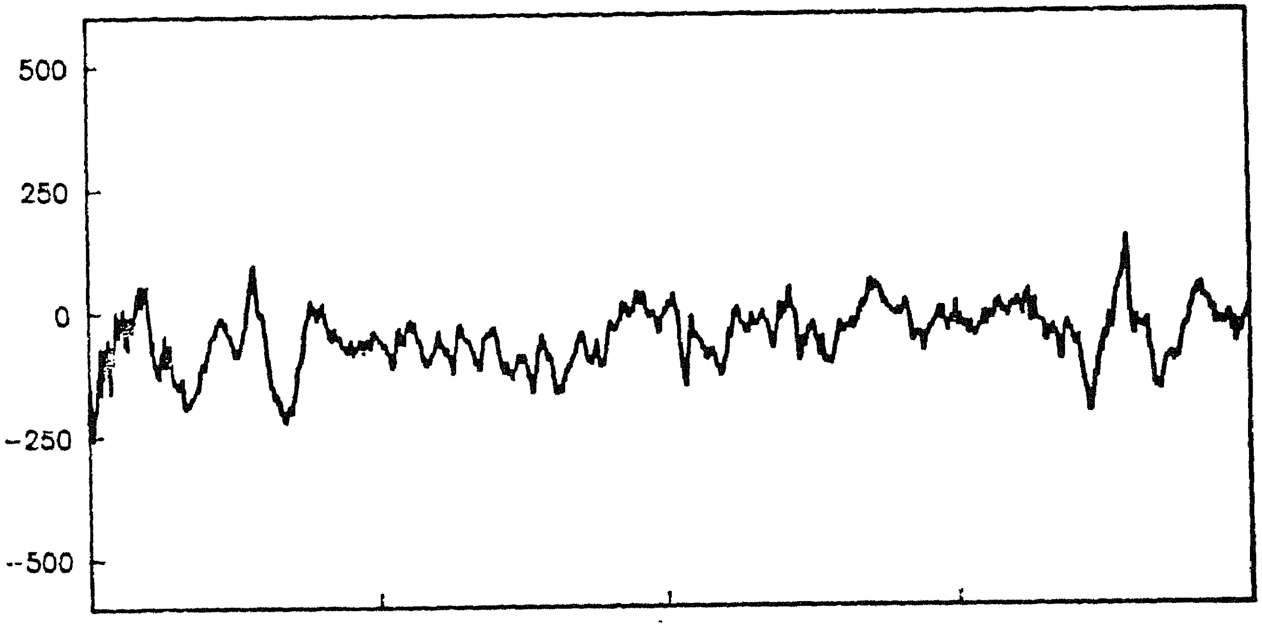


Figure 3.5: Original EEG waveform 1

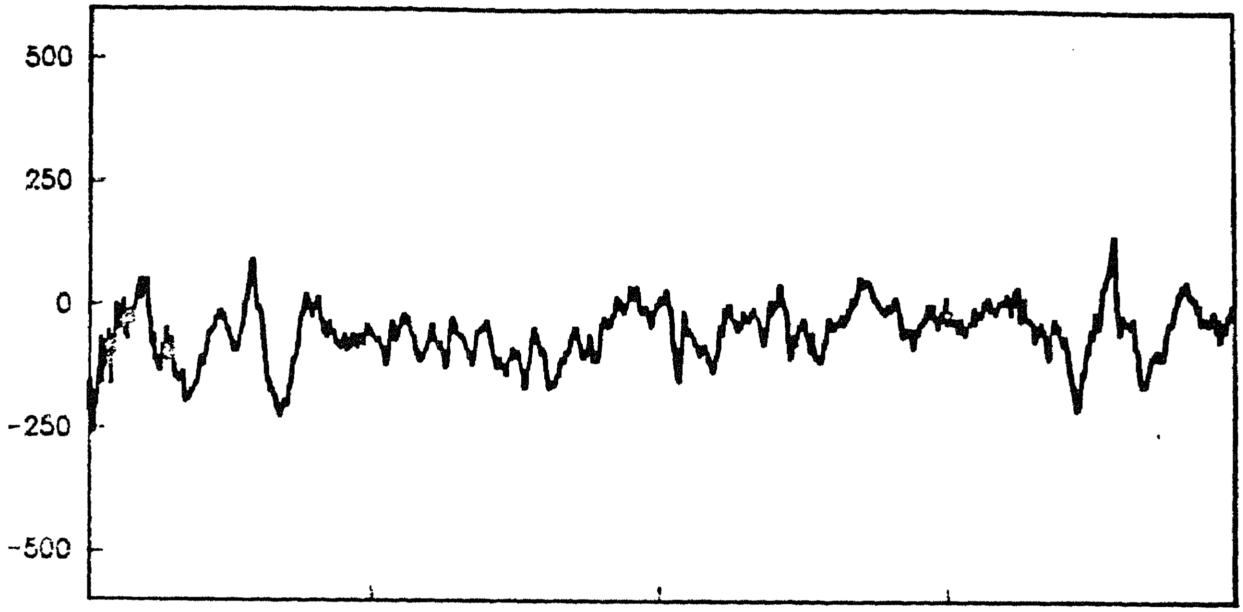


Figure 3.6: Reconstructed waveform 1 by using $\delta = 8$

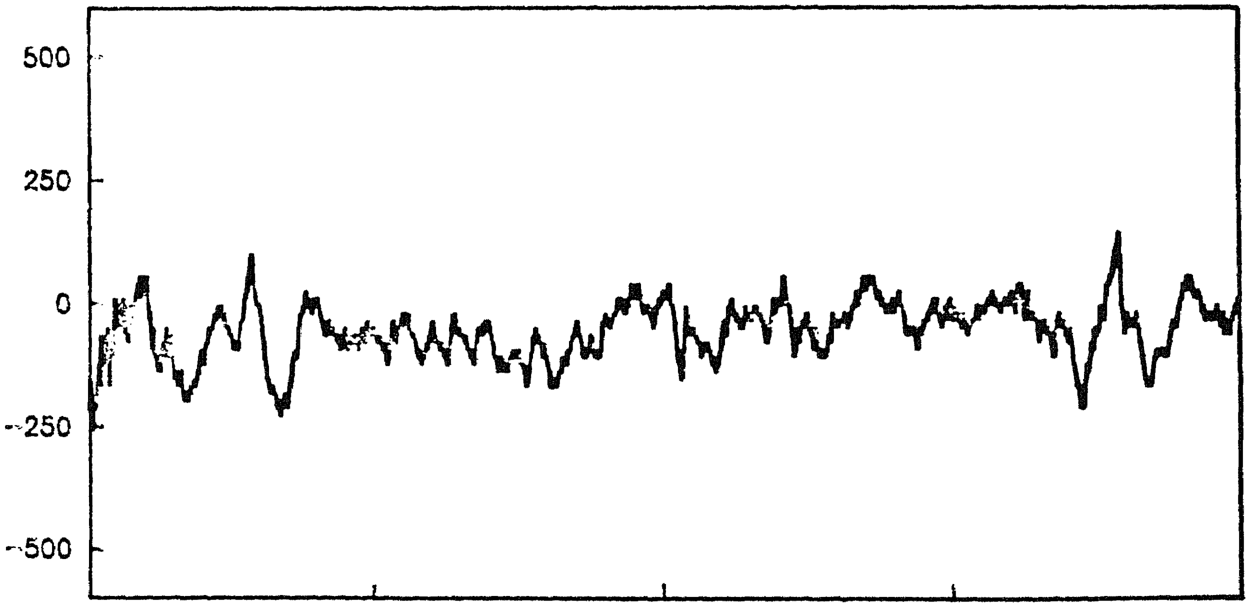


Figure 3.7: Reconstructed waveform 1 by using $\delta = 16$

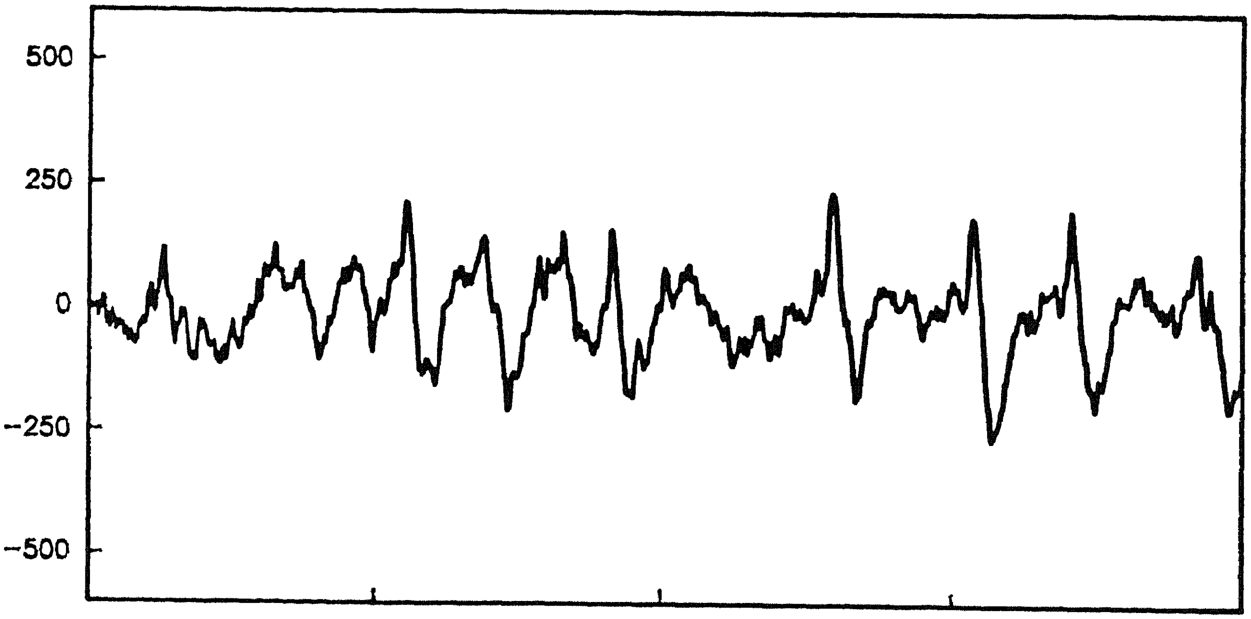


Figure 3.8: Original EEG waveform 2

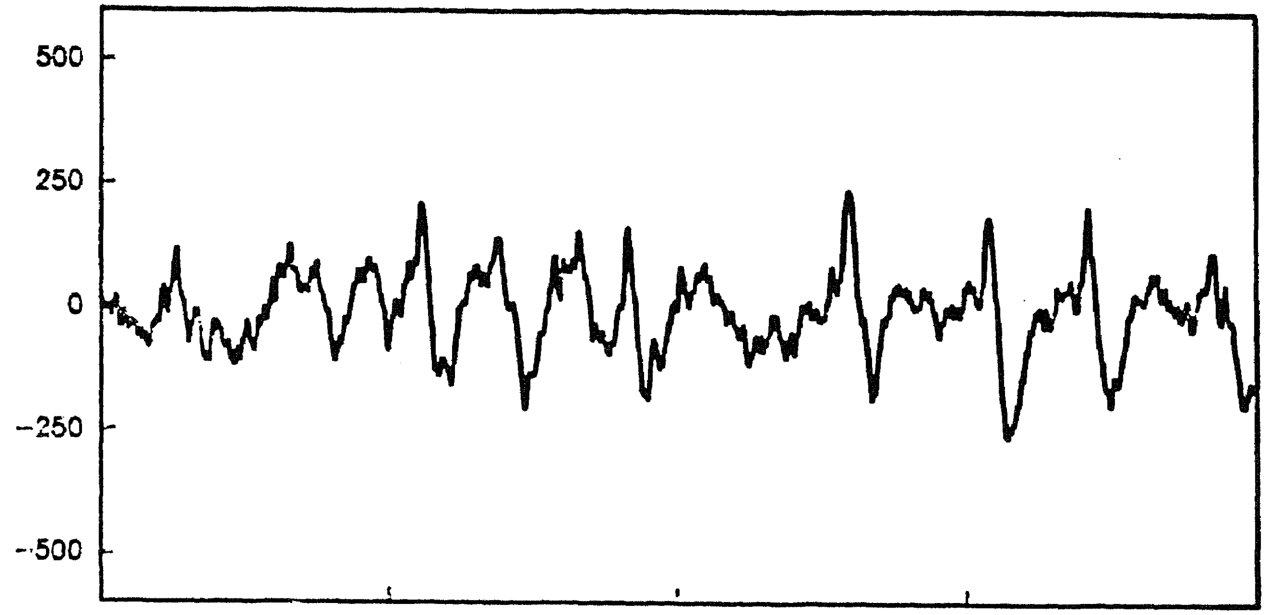


Figure 3.9: Reconstructed waveform 2 by using $\delta = 8$

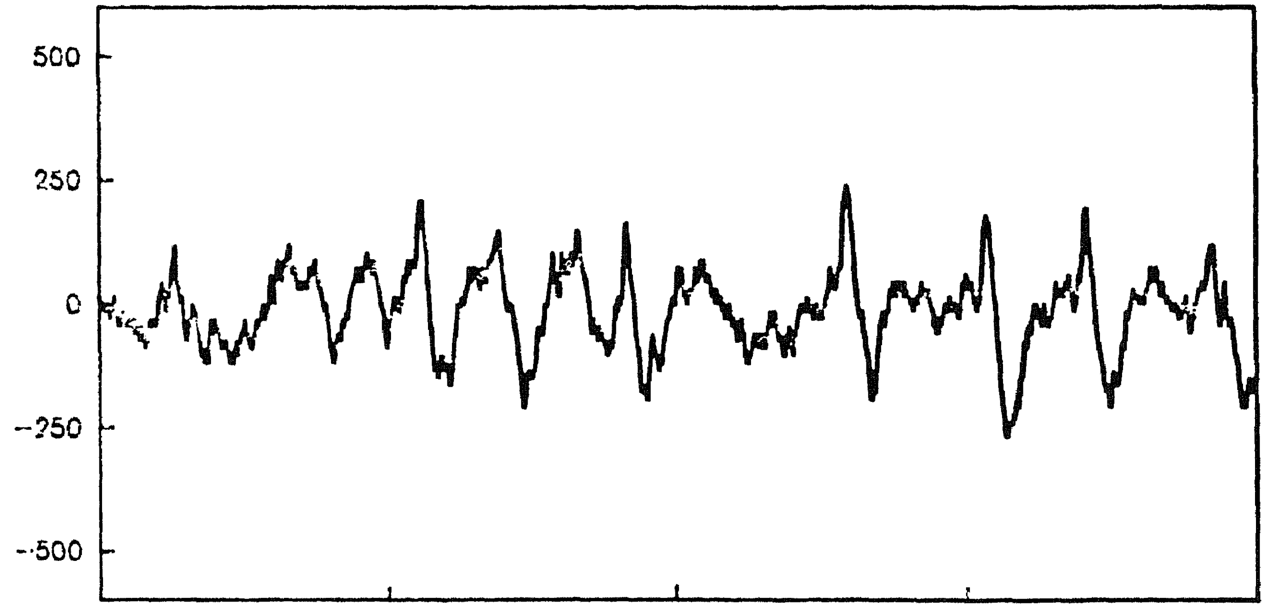


Figure 3.10: Reconstructed waveform 2 by using $\delta = 16$

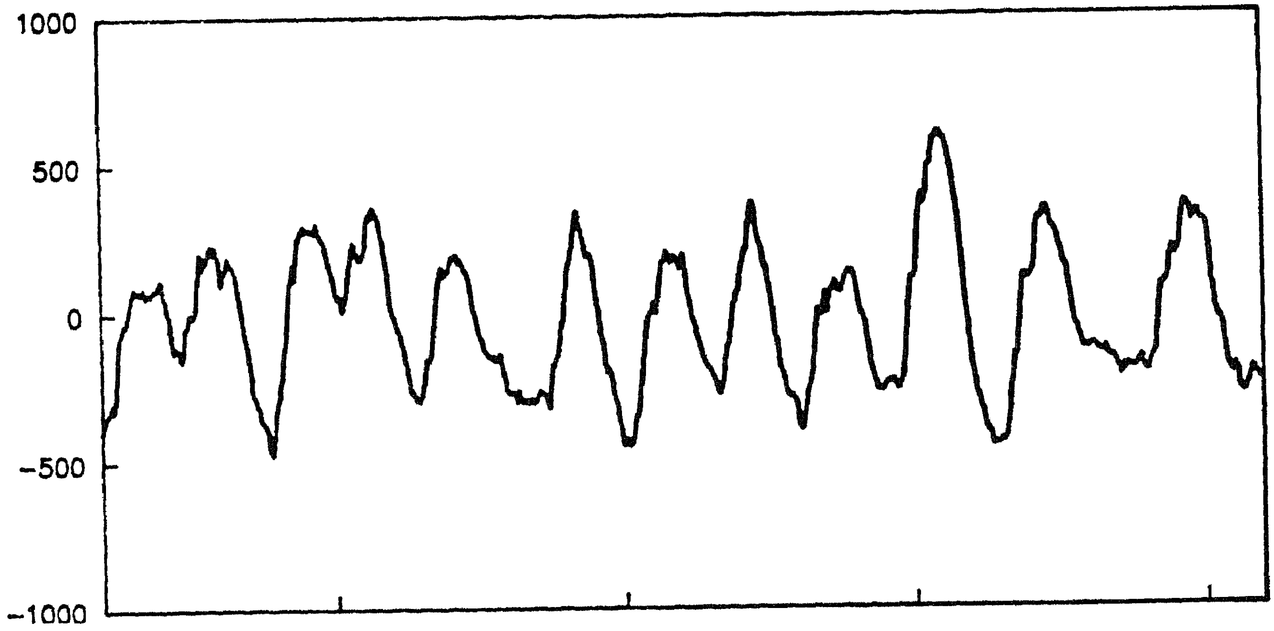


Figure 3.11: Original EEG waveform 3

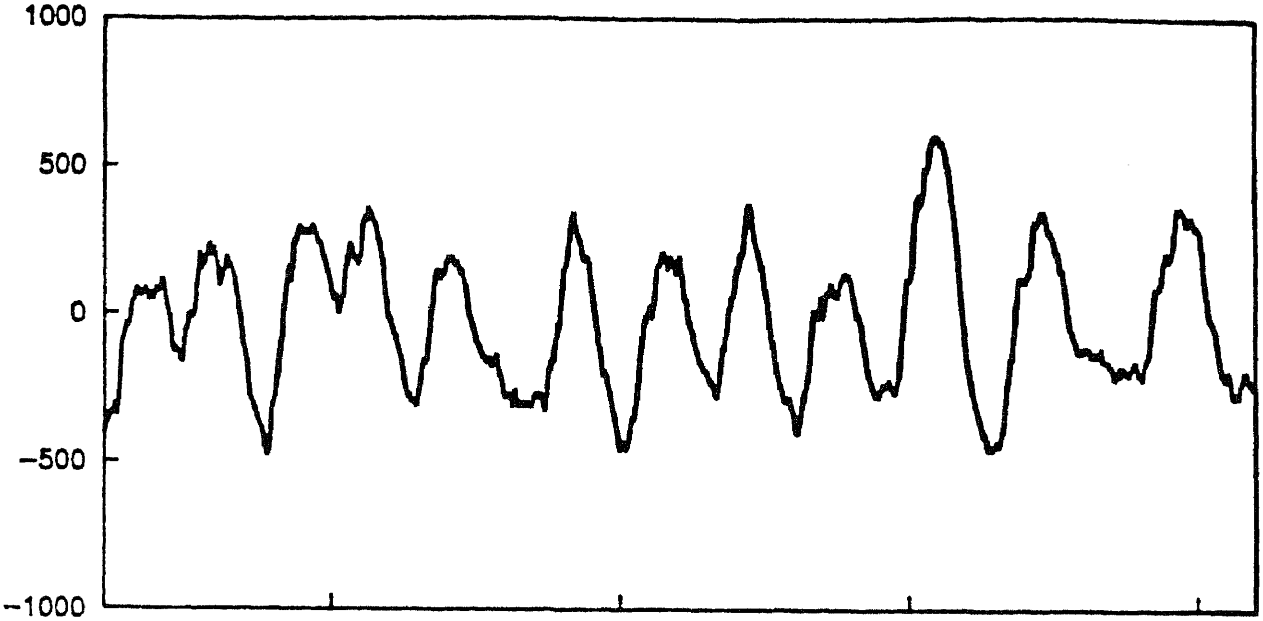


Figure 3.12: Reconstructed waveform 3 by using $\delta = 8$

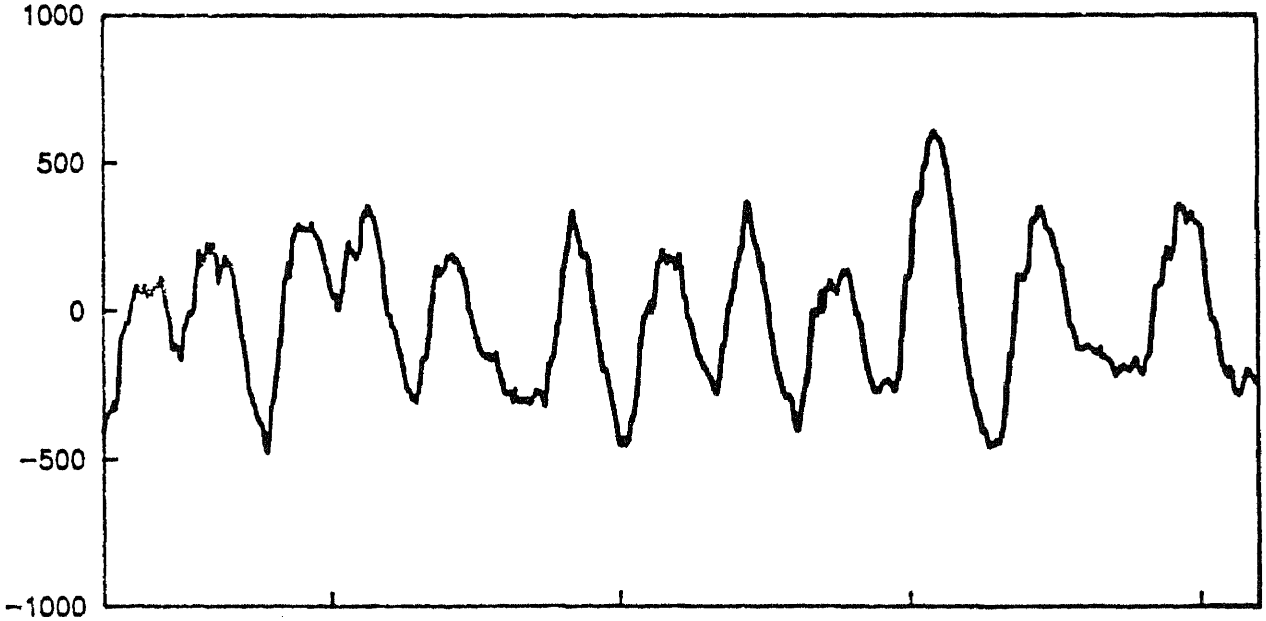


Figure 3.13: Reconstructed waveform 3 by using $\delta = 16$

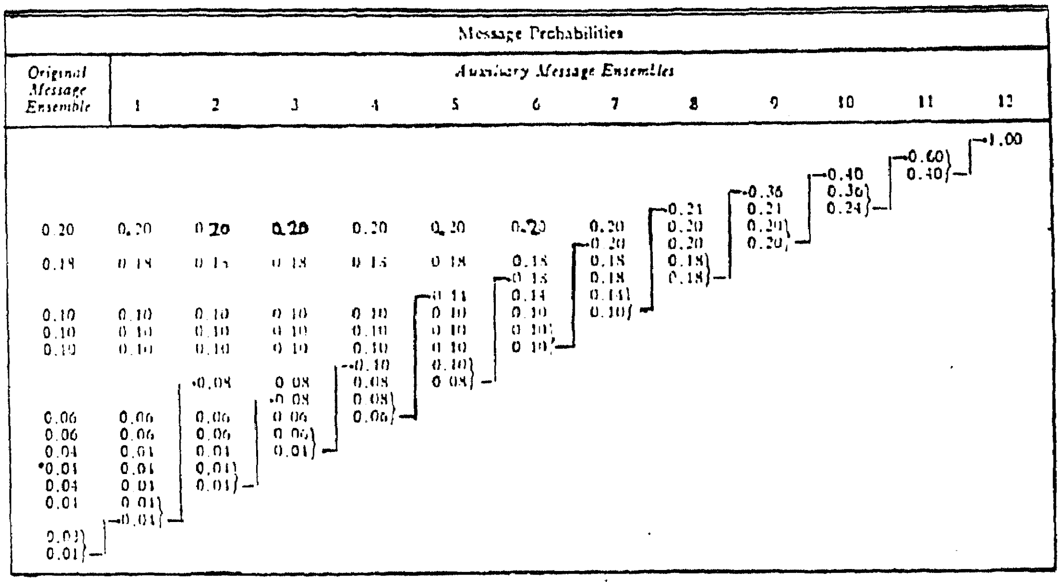
Chapter 4

Huffman coding

4.1 The principle of Huffman coding

Huffman coding is a data-compression technique due to D.A.Huffman which represents the symbols of an alphabet by minimizing the average code length. This algorithm[24] replaces each character by a variable length code according to the relative frequency of that character in the text. This technique is a fixed to variable length coding. The Huffman code is an optimum code since it results in the shortest average code length of all fixed to variable encoding techniques when the frequencies are given. In addition, Huffman codes have a prefix property which means that no short code word appears as the beginning of a longer code word. The Huffman code can be developed through the utilization of a tree structure. Huffman's method makes two passes over the data: one pass is to collect frequency counts of the characters in the text, followed by the construction of a Huffman tree; the second is to encode and transmit the characters themselves based on the static tree structure.

Table 4.1 is an example of Huffman coding. First, these messages are listed in descending order of their frequencies. The two messages with the smallest



Message Probabilities												
Original Message Ensemble	*Auxiliary Message Ensembles*											
	1	2	3	4	5	6	7	8	9	10	11	12
												1.00
											0.60	
										0.40	0.40	
									0.36	0.36		
								0.24	0.24	0.24		
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20			
							0.20	0.20	0.20			
0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18				
						0.18	0.18	0.18				
					0.14	0.14	0.14					
0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10					
0.10	0.10	0.10	0.10	0.10	0.10	0.10						
0.10	0.10	0.10	0.10	0.10	0.10	0.10						
				0.10	0.10							
		0.08	0.08	0.08	0.08							
			0.08	0.08								
0.06	0.06	0.06	0.06	0.06								
0.06	0.06	0.06	0.06									
0.04	0.04	0.04	0.04									
•0.04	0.04	0.04										
0.04	0.04	0.04										
0.04	0.04											
	0.04											
0.03												
0.01												

Table 4.1: Optimum Binary Coding Procedure

i	$P(i)$	$L(i)$	$P(i)L(i)$	Code
1	0.20	2	0.40	10
2	0.18	3	0.54	000
3	0.10	3	0.30	011
4	0.10	3	0.30	110
5	0.10	3	0.30	111
6	0.06	4	0.24	0101
7	0.06	5	0.30	00100
8	0.01	5	0.20	00101
9	0.04	5	0.20	01000
10	0.01	5	0.20	01001
11	0.04	5	0.20	00110
12	0.03	6	0.18	001110
13	0.01	6	0.06	001111
			$L_{av} = 3.42$	

Table 4.2: Results of Optimum Binary Coding Procedure

frequencies are combined into a node with the lowest frequency. Then, the two nodes with the lowest frequencies (including joint frequencies) are merged. The tree is built up the same way for the remaining nodes as shown in table 4.1. By assigning 0 (1) to every left (right) edge emanating from each node, one can derive the Huffman code for each message[24].

The Huffman length of a code word is the number of its bits. The lengths of all the encoded messages derived from table 4.1 are given in table 4.2 Thus this code uses 3.42 bits per message. The binary fixed length code for 13 messages needs 4 bits. Therefore Huffman coding saves 0.58 bits per message.

4.2 Huffman coding in EEG data compression

In this project the length of sampled data is 12 bits long. Thus the value of every 12-bit-sampled data of EEG is between -2048 and 2048. Actually, in all the data files the value is between -1000 and 1000 and most of the files are between -300 and 300 except for some special waveforms. Thus the data variable range is very large. It is very difficult to use Huffman coding to code EEG data because if we use Huffman coding directly to code this file, we should give every value a different code. We will therefore obtain up to thousands different codes for the EEG file. It will become very complicated. In other words it is difficult to be implemented in EEG data coding.

In using Huffman coding in this application, the numbers of data value have to be reduced. In this project we combined a number of data value which have a small difference between them into a group and assign a specific value for this group. We use the way as table 4.3 shows to combine the data. For example, if the data values are between 0 and 8, we set these data to a value of 4. Thus these data become identical. As we will see, we can combine data in this way we will get the same quantization error as in ADM in chapter 3 ($\delta = 8$). Table 4.4 shows the combined data value and its corresponding frequency of the data files we used in chapter 3. For example, the frequency of 4 is 385 and frequency of -4 is 483. The number of data values is reduced to about one hundred. Thus we can use Huffman coding to code this file.

The Huffman coding result for EEG data corresponding to the relations given in table 4.4 is shown table 4.5. Table 4.5 shows that the shortest word length is 4 bits; the longest word length is 13 bits and the average word is

Table 4.3: Combine the data in to group

actual value	set value
$0 \rightarrow \pm 8$	± 4
$\pm 9 \rightarrow \pm 16$	± 12
$\pm(8i+1) \rightarrow \pm(i+1)8$	$\pm 8i+4$

5.08 bits. Thus, this method saves 6.92 bits per sample. Since Huffman coding does not produce an error[25], the only error we get here is during the data combination. Comparing table 4.3 with table 3.1 we can see the error produced from data combination is the same as by using ADM method while $\delta = 8$. Because the data combination quantizes the data the same as in the ADM method, then for the same original data file we can get the same combining data value. Thus we can get the same reconstructed waveform as we get by using ADM ($\delta = 8$). (Fig-3.8, Fig-3.11 and Fig-3.14).

```
A[4]=385 B[-4]=483
A[12]=400 B[-12]=405
A[20]=362 B[-20]=389
A[28]=317 B[-28]=445
A[36]=266 B[-36]=455
A[44]=251 B[-44]=348
A[52]=202 B[-52]=337
A[60]=160 B[-60]=312
A[68]=138 B[-68]=305
A[76]=98 B[-76]=301
A[84]=76 B[-84]=262
A[92]=48 B[-92]=222
A[100]=57 B[-100]=194
A[108]=47 B[-108]=160
A[116]=34 B[-116]=143
A[124]=29 B[-124]=128
A[132]=36 B[-132]=105
A[140]=21 B[-140]=96
A[148]=18 B[-148]=78
A[156]=9 B[-156]=67
A[164]=3 B[-164]=74
A[172]=6 B[-172]=48
A[180]=7 B[-180]=50
A[188]=3 B[-188]=40
A[196]=1 B[-196]=38
A[204]=0 B[-204]=42
A[212]=1 B[-212]=26
A[220]=0 B[-220]=36
A[228]=0 B[-228]=25
A[236]=0 B[-236]=24
A[244]=0 B[-244]=28
A[252]=0 B[-252]=16
A[260]=0 B[-260]=19
A[268]=0 B[-268]=12
A[276]=0 B[-276]=15
A[284]=0 B[-284]=14
A[292]=0 B[-292]=12
A[300]=0 B[-300]=9
A[308]=0 B[-308]=13
A[316]=0 B[-316]=7
A[324]=0 B[-324]=10
A[332]=0 B[-332]=17

A[340]=0 B[-340]=8
A[348]=0 B[-348]=14
A[356]=0 B[-356]=12
A[364]=0 B[-364]=9
A[372]=0 B[-372]=12
A[380]=0 B[-380]=11
A[388]=0 B[-388]=18
A[396]=0 B[-396]=15
A[404]=0 B[-404]=11
A[412]=0 B[-412]=12
A[420]=0 B[-420]=7
A[428]=0 B[-428]=12
A[436]=0 B[-436]=7
A[444]=0 B[-444]=4
A[452]=0 B[-452]=7
A[460]=0 B[-460]=5
A[468]=0 B[-468]=4
A[476]=0 B[-476]=7
A[484]=0 B[-484]=8
A[492]=0 B[-492]=7
A[500]=0 B[-500]=5
A[508]=0 B[-508]=2
A[516]=0 B[-516]=1
A[524]=0 B[-524]=4
A[532]=0 B[-532]=4
A[540]=0 B[-540]=0
A[548]=0 B[-548]=3
A[556]=0 B[-556]=2
A[564]=0 B[-564]=2
A[572]=0 B[-572]=1
A[580]=0 B[-580]=0
A[588]=0 B[-588]=1
A[596]=0 B[-596]=0
A[604]=0 B[-604]=0
A[612]=0 B[-612]=0
A[620]=0 B[-620]=0
A[628]=0 B[-628]=0
A[636]=0 B[-636]=0
A[644]=0 B[-644]=0
A[652]=0 B[-652]=0
A[660]=0 B[-660]=0
```

Table 4.4: Relationship between data value and the frequency

Table 4.5: Result of Huffman coding for EEG data

frequency	length of code
1	13 bits
2	12 bits
5	11 bits
3, 4, 8, 10, 11	10 bits
6, 13, 14, 15, 16, 17, 19	9 bits
9, 12, 25, 26, 27, 28, 29, 30 31, 33, 34, 35, 36, 38, 44	8 bits
65, 67, 70, 74, 76, 82, 84	7 bits
46, 48, 49, 52, 98, 102, 125 131, 132, 138, 142, 162	6 bits
85, 91, 170, 190, 199, 209 220, 224, 228, 231 236, 237, 253, 255, 265, 271	5 bits
274, 277, 280, 284, 295, 307	4 bits

Chapter 5

The combination of ADM and Huffman coding

After applying adaptive delta modulation to compress the EEG data we can get a codeword file. For the same sampled data file the length of the codeword depends on the value of δ. In this project we obtained two different codeword lengths from the same data file by using two different δs ($\delta_1 = 8$; $\delta_2 = 16$). In the first codeword file the range of the codeword value is between 00000 and 11111; in the second, the values vary from 0000 to 1111. Thus, it is very easy to apply Huffman coding to these codewords. Tables 5.1 and 5.2 are the Huffman coding trees for $\delta = 8$ and $\delta = 16$, respectively and tables 5.3 and 5.4 are the results. From table 5.3, we can see the shortest length of codeword is 2 and the average wordlength is 3.8235. In table 5.4 the shortest length of codeword is 2 and the average wordlength is 2.7986. Comparing to the delta modulation method, this method saves up to 1.1765 bits per word and 1.2014 bits per word respectively. As we discussed in chapter 4, Huffman coding never produces any error[25]. Therefore if we combine ADM with Huffman coding we will get the same error as we get with adaptive delta coding and the same reconstructed waveforms we obtained in chapter 3. However, the average number of bits per word will be significantly reduced.

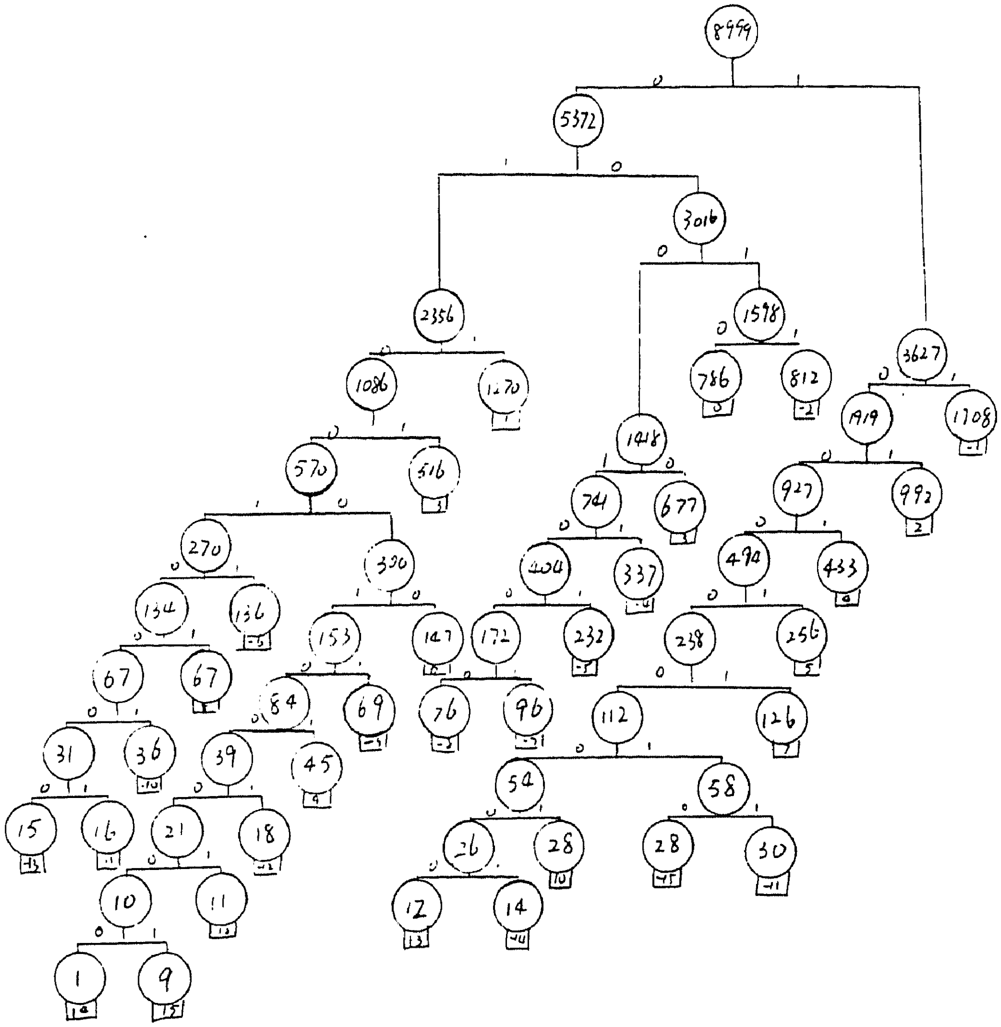


Table 5.1: Huffman coding tree $\delta = 8$

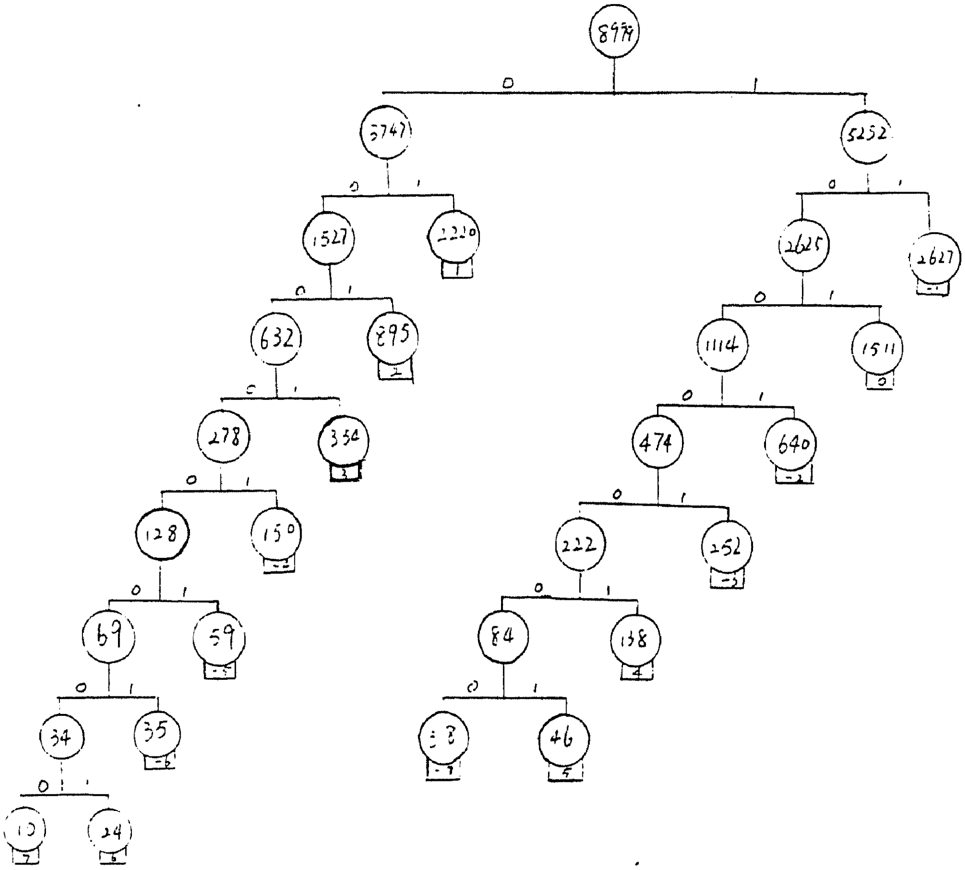


Table 5.2: Huffman coding tree $\delta = 16$

Table 5.3: Result of $\delta = 8$

i	f(i)	L(i)	code
-1	1708	2	11
1	1270	3	011
2	992	3	101
-2	812	4	0011
0	786	4	0010
3	677	4	0000
-3	516	4	0101
4	433	4	1001
-4	337	5	00010
5	256	5	10001
-5	232	6	000111
6	147	6	010000
-6	136	6	010011
7	126	6	100001
-7	96	7	0001101
-8	76	7	0001100
8	67	7	0100101
-9	69	7	0100011
9	45	8	01000101
-10	36	8	01001001
10	28	9	000110001
-11	30	9	000110011
11	16	9	010010001
-12	18	9	010001001
-13	15	9	010010000
-15	28	9	000110010
12	11	10	0100010001
13	12	10	0001100000
-14	14	10	0001100001
14	1	11	01000100000
15	9	11	01000100001

Table 5.4: Result of $\delta = 16$

i	f(i)	L(i)	code
-1	2627	2	11
1	2220	2	01
0	1511	3	101
2	895	3	001
-2	640	4	1001
3	354	4	0001
-3	252	5	10001
-4	150	5	00001
4	138	6	100001
-5	59	6	010001
5	46	7	1000001
-7	38	7	1000000
-6	35	7	0000001
6	24	8	00000001
7	10	8	00000000

Chapter 6

Discussion

In the previous sections, direct data compression methods and transformation technique used for EEG data compression have been discussed. Table 6.1 provides a summary of EEG data compression techniques in terms of compression ratio (CR) and error. The compression ratio calculation of each technique is based on comparing compression parameters with the number of samples in the original data file and comparing the word length of the compression parameters with the word length of samples in the original data file. The former is used to calculate the compression ratio for transformation techniques (Walsh transformation technique). The latter is used to calculate the direct data compression methods. The error is calculated the same as we discussed in chapter 2. For the transformation method, the error comes from reducing the number of transformation coefficients. This error occurring in the ADM method is the quantization error. We notice that the Walsh transformation method produced larger error than the direct data compression methods for the same degree of compression, and that some high frequency signals are cut off. This is because the distribution of power in the frequency spectrum of the EEG is very wide[26]. Therefore when it appears in the Walsh transformation the power spectrum is very diffuse[27]. During the data compression, we set some Walsh coefficients to zero. Actually these data are not near zero. As a result a large error occurs as well as the loss of high frequency signals.

In the direct data compression methods, we have two methods (1). delta; (2). Huffman coding. In the first method we can see from table 6.1 that if $\delta = 8$ we get $CR = 0.416$ (58.4% compressed) and an error of 2.5 and if $\delta = 16$ we get $CR = 0.333$ (66.7% compressed) and a corresponding error of 6.019. It seems that if the CR is reduced by 0.1 the error will become three times larger than before. But when the error is compared with the sampled data value (usually the range is 600), it is still very small. From the reconstructed waveform (Fig-3.8 and Fig-3.9; Fig- 3.11 and Fig-3.12; Fig-3.14 and Fig-3.15, we can see there are not big differences between $CR = 0.416$ and $CR = 0.333$ and also no big effect on the spikes and the particular waveforms. Therefore we prefer to use $CR = 0.333$, in other words we can get 66.7% data compressed.

One of the advantages of Huffman coding is that it never produces an error in the reconstructed file[25]. However Huffman coding is impossible to use directly in EEG data compression as we discussed in chapter 4. In chapter 4 we made some assumption got almost the same result (table 6.1) as we get from delta coding ($\delta = 8$).

The Huffman coding method is hard to apply directly to the EEG data file even if we did some data combining. But we can apply this method to the ADM system, in other words the codewords of the second difference EEG data were encoded again by Huffman coding. It can be seen from the table that ADM encoding followed by Huffman encoding resulted in a high data compression ratio while the error remains the same as ADM encoding. In other words the reconstructed waveform is the same as that reconstructed from ADM encoding.

The normal EEG, spikes and the particular EEG waveform such as slow rhythm can be reconstructed the same as we obtained in the chapter 3 (Fig- 3.8, Fig-3.9, Fig-3.11, Fig-3.12, Fig-3.14 and Fig-3.15) while the data file are compressed 10% more than ADM coding.

From the above analysis, it seems the delta modulation followed by Huffman coding is the best way to compress the EEG data among these data compression methods used in this project. However before we compress the data we have already made some assumptions during the A/D convertion. We fixed the sampling frequency at 0.2 KHz and fixed the wordlength as 12 bits. Conceptually, data compression is the process of detecting and eliminating redundancies in a given data set. According to Shannon's definition[28] the redundancy is a fraction of a message or datum which is unnecessary and hence repetitive in the sense that if it were missing, the message would still be essentially complete, or at least could be completed. Hence the first step towards EEG data compression is of the selection of minimum sampling rate and wordlength. If we reduced the sampling frequency to exactly two times of the EEG frequency and exactly the length of word to present the EEG signal, it will not affect the result of the Walsh transform, because it will not affect the distribution of EEG frequency. Thus the Walsh coefficients will not be affected. However in adaptive delta modulation we use the varied steps to trace the EEG waveform. If the sampling rate is reduced, the difference between the adjacent pair samples will be increased, since the δ is selected by using the largest difference divided by the length of codeword. Therefore if we still use the same length of codeword to code the difference, the δ will be increased and hence the quantization error will be increased. Also the delta modulation and Huffman code method compress the data by reducing the wordlength. Therefore, if we reduce the wordlength before using any data compression method, then the data compression degree in the delta coding and Huffman coding will be decreased. And the Walsh transformation method will not be affected, because the Walsh transform method compresses the data by reducing the number of samples rather than the wordlength.

The EEG frequency band is between 0 and 30 Hz[1] except for some special spikes. Thus the high frequency cut off for the low-pass filter in the EEG machine is 70 Hz. Also, the value of the data varies from -1000 to 1000 including some special cases as we discussed in Chapter 4. Therefore, 140 Hz sampling rate and 11 bits long wordlength are enough for the EEG waveform. If we use these parameters it will not affect the Walsh transformation as we discussed above while in the delta coding and the Huffman coding methods the data compression degree will decrease and the error will increase. In this situation, maybe, the Walsh transformation method may be better than the direct data compression methods. The reason for using 0.2 KHz sampling and 12 bits long wordlength is because when we use the DASH-16 in the A/D converting the wordlength of sampled data is fixed as 12 bits long[18]. Also the sampling rate in the DASH-16 can only be 0.1 KHz, 0.2 KHz, ..., etc and it can not be 0.14 KHz[18]. Therefore the lowest sampling frequency for the EEG waveform can be chosen is 0.2 KHz. Under this situation, we found the adaptive delta modulation followed by Huffman coding is the best for EEG data compression among all signal data compression methods we used in this project.

Table 6.1: Summary some EEG data compression schemes

Technique	CR=compressed data file /original data file		$error = \frac{\sum i=1^{N}\sqrt{(x_i-\hat{x_i})^2}}{N}$
Walsh transformation	0.8	the last 1/5 Walsh spectrum was ignored	18.4
	0.4	the last 3/5 Walsh spectrum was ignored	25.8
	0.32	the last 17/25 Walsh spectrum was ignored	26.5
Adaptive delta coding	0.416	$\delta = 8$	2.547
	0.333	$\delta = 16$	6.019
Huffman coding	0.423		2.547
ADM followed by Huffman coding	0.311	$\delta = 8$	2.547
	0.233	$\delta = 16$	6.019

Appendix A

Computer Program

The following includes programs to perform Adaptive delta modulation, reconstruct ADM waveforms, calculate frequency of occurrence of data for Huffman coding, perform data combination for Huffman coding, and calculate errors. All the programs are written in Pascal language and the flow charts for the programs are in the Appendix B.

(1). Adaptive delta modulation

In this program we kept the first original data for reconstructing and also as the first predictor data. By subtracting the current data with the predictor data we obtained a suitable step $k_i\delta$, then we output k_i as the codeword and adjust the predictor. Proceeding in this way, we can code the whole data file.

(2). Reconstruct signal from ADM

Because we have already stored the first original data, we can reconstruct the data by adding the $k_i\delta$ to the previous one. k_i is the codeword in the code file.

(3). Calculate the frequency of the data value

By counting the number of occurrences of the same data value we can obtain the corresponding frequency of occurrence.

(4). Calculate the frequency of data group value

Comparing the data with 8i, if the data is between $8(i-1)$ and 8i, we set it to $8(i-1)+4$. Then we count the number of data in the same interval $8(i-1)$ and 8i.

(5). Error calculation

We compare the data from original file and the reconstructed file by using the normalized square error.

```
************************************************************
**         PROGRAM 1 ADAPTIVE DELTA MODULATION            **
**        -------------------------------------           **
** The  original  data  comes from  "infile"  and  the **
** compressed data sends to "outfile". The first original **
** data is kept in outfile for reconstructing and also as **
** the  first  predictor  data "a".  By  subtracting  the **
** current data "b" with "a",  we obtain a suitable  step **
** "n*d". Output n  as the  codeword and add "a" by "n*d" **
** to get a new predictor.                                **
************************************************************

program code (input,output,infile,outfile);
    var
      infile,outfile:text;
      a,b,n,c:integer;
    begin
      assign (infile,'a:gar07.u');
      assign (outfile,'a:gar07.c');
      reset (infile);
      rewrite (outfile);
      readln (infile,a);
      writeln (outfile,a);
      c:=a;
      while not eof(infile) do
        begin
          readln (infile,b);
          n:=0;
          if a>b then
            begin
              while (a>b) and (n<15) do
                begin
                  b:=b+7;
                  n:=n+1;
                end;
              b:=b-3;
              if (a<b) and (n>1) then
                n:=-(n-1)
              else
                n:=-n;
            c:=a-((-n)*7);
            end
          else
            if a<b then
              begin
                while (a<b) and (n<15) do
                  begin
                    b:=b-7;
                    n:=n+1;
                  end;
                b:=b+3;
                if a>b then
                  n:=n-1;
                c:=a+(n*7);
              end
            else
              n:=0;
          writeln (outfile,n);
          a:=c;
        end;
        close (outfile);
      end.
************************************************************
**           PROGRAM 2 RECONSTRUCTED FROM ADM             **
**         ------------------------------------           **
** The codewords come from "infile" and the reconstructed **
** data   send  to   "outfile".  Because  we have already **
** reserved the first  original data.  Therefore  we  can **
** reconstruct  the  data  by  adding  the  "n*d" to  the **
** previous one. Here n is the codeword in the code file. **
************************************************************

  program recov (input,output,infile,outfile);
    var
      infile,outfile:text;
      a,n:integer;
    begin
      assign (infile,'a:gar07.c');
      assign (outfile,'a:gar07.r');
      reset (infile);
      rewrite (outfile);
      readln (infile,a);
      writeln (outfile,a);
      while not eof(infile) do
        begin
          readln (infile,n);
          a:=a+7*n
          writeln (outfile,a);
        end;
        close (outfile);
    end.


************************************************************
**  PROGRAM 3 CALCULATE THE FREQUENCY OF THE DATA VALUE   **
** ------------------------------------------------------ **
** Count  the  number  of same data  value  from "infile" **
** and send  the  data  value   with  the   corresponding **
** frequency to "outfile".                                **
************************************************************


program st (input,output,infile,outfile);
  var
    infile,outfile:text;
    i,j,k:integer;
    A:array [0..9,0..9,0..9] of integer;
    B:array [0..9,0..9,0..9] of integer;
  PA:array [0..9,0..9,0..9] of real;
  PB:array [0..9,0..9,0..9] of real;
  c:real;
begin
  assign(infile,’a:gar07.u’);
  reset (infile);
  assign(outfile,’a:gar07.s’);
  rewrite (outfile);
  for i:=0 to 9 do
    begin
      for j:=0 to 9 do
        begin
          for k:=0 to 9 do
            begin
              A[i,j,k]:=0;
              B[i,j,k]:=0;
            end;
        end;
    end;
  readln (infile,c);
  while not eof (infile) do
    begin
      if c>0 then
        begin
          c:=trunc(c);
          i:=trunc(c/100);
          j:=trunc((c-i*100)/10);
          k:=trunc(c-i*100-j*10);
          A[i,j,k]:=A[i,j,k]+1;
        end
      else
       begin
        c:=abs(c);
        c:=trunc(c);
        i:=trunc(c/100);
        j:=trunc((c-i*100)/10);
        k:=trunc(c-i*100-j*10);
        B[i,j,k]:=B[i,j,k]+1;
       end;
       readln(infile,c);
    end;
   for i:=0 to 9 do
    begin
     for j:=0 to 9 do
      begin
       for k:=0 to 9 do
        begin
         writeln(outfile,’A[’,i,j,k,’]=’,A[i,j,k],’   ’
                ’ B[-’,i,j,k,’]=’,B[i,j,k],);
        end;
      end;
    end;
   close (outfile);
end.
```

```
*************************************************************
** PROGRAM 4 CALCULATE THE FREQUENCY OF DATA GROUP  VALUE **
** ------------------------------------------------------ **
** Get the data from the "infile" and  compare  the  data **
** with i*8. If the data between (i-1)*8  and i*8, set it **
** to  (i-1)*8+4.  Count  the number  of data in the same **
** interval (i-1)*8, i*8, output the value (i-1)*8+4  and **
** the corresponding frequency to "outfile". i=1,2, ... . **
*************************************************************


program st (input,output,infile,outfile);
  var
    infile,outfile:text;
    i,j,k:integer;
    A:array [0..125] of integer;
    B:array [0..125] of integer;
    c:real;
  begin
    assign(infile,'gar07.u');
    reset (infile);
    assign(outfile,'gar07.s');
    rewrite (outfile);
    for i:=0 to 125 do
      begin
        A[i]:=0;
        B[i]:=0;
      end;
    readln (infile,c);
    while not eof (infile) do
      begin
        if c>0 then
          begin
            c:=trunc(c);
            i:=trunc(c/8);
            A[i]:=A[i]+1;
          end
        else
         begin
          c:=abs(c);
          c:=trunc(c);
          i:=trunc(c/8);
          B[i]:=B[i]+1;
         end;
         readln(infile,c);
      end;
     for i:=0 to 125 do
       begin


      writeln(outfile,'A[',i*8+4,']=',A[i],
                  'B[',i*8+4,']=',B[i]);

        end;
        close (outfile);
   end.
```

```
*************************************************************
**               PROGRAM 5 ERROR CALCULATION              **
**      ---------------------------------------           **
** Get  the  data  from  original  file (infile1) and the **
** reconstructed file (infile2). calculated the error as: **
**  $\frac{\sum_{i=0}^{N}\sqrt{(x_i - \hat{x}_i)^2}}{N}$   **
**                                                        **
*************************************************************
```

```
program sums (input, infile1,infile2);
  var
     infile1,infile2:text;
     a,b,sum,x:real;
     n:integer;
  begin
    assign (infile1,'a:gar07.u');
    assign (infile2,'a:gar07.r');
    reset (infile1);
    reset (infile2);
    sum:=0;
    n:=1;
    while no eof (infile1) and (infile2) do
       begin
         readln (infile1,a);
         readln (infile2,b);
         x:=(b-a)*(b-a);
         x:=sqrt(x);
         sum:=x+sum;
         n:=n+1;
       end;
    sum:=sum/n;
    writeln(sum);
  end.
```

Appendix B

Flow-Chart for the programs

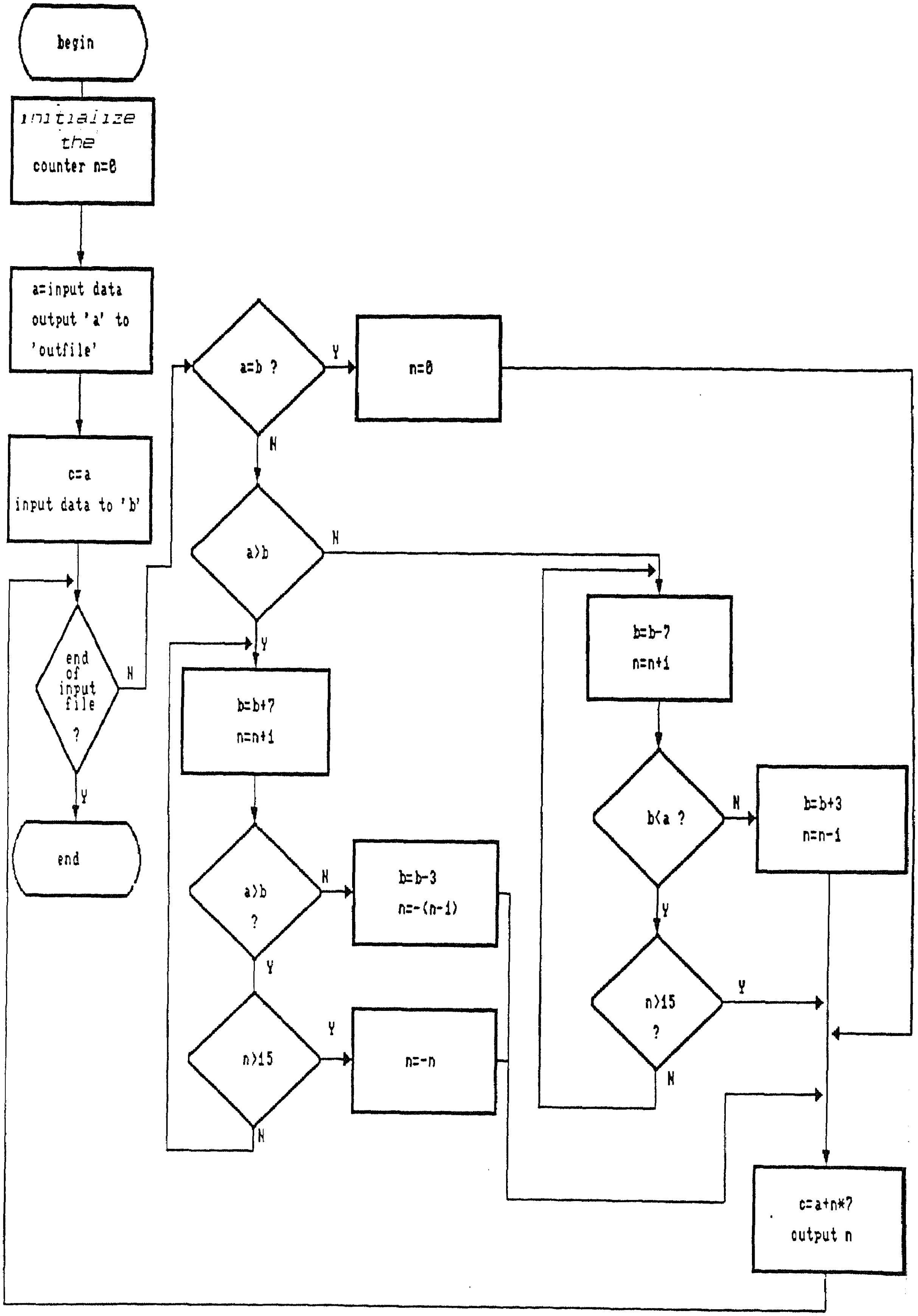


Table B.1: Adaptive delta modulation

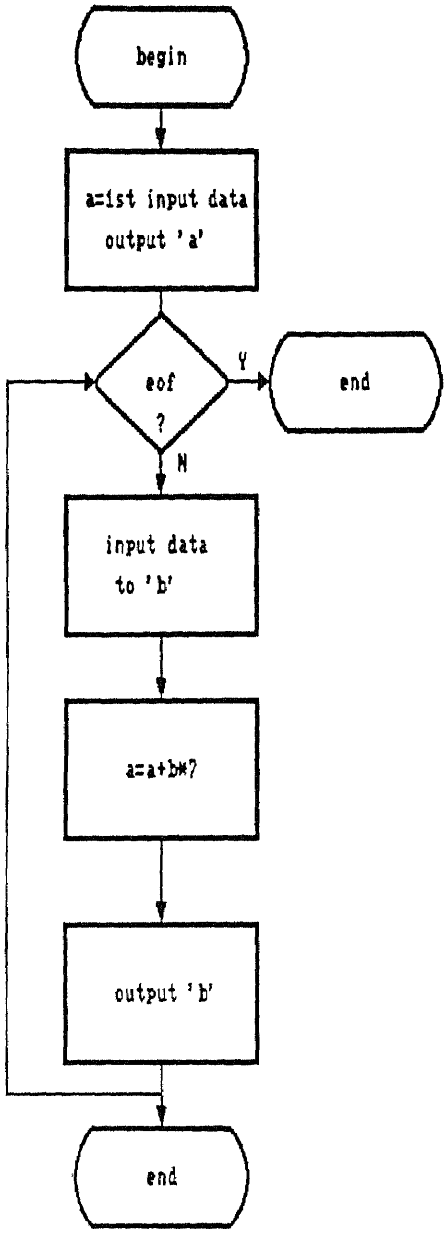


Table B.2: Reconstructed signal

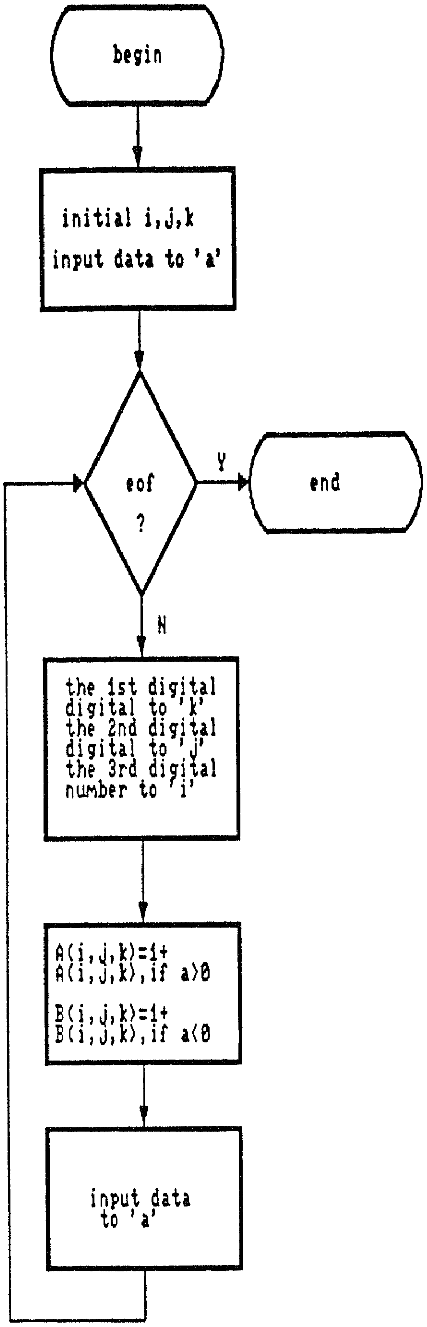


Table B.3: Calculation the data frequency

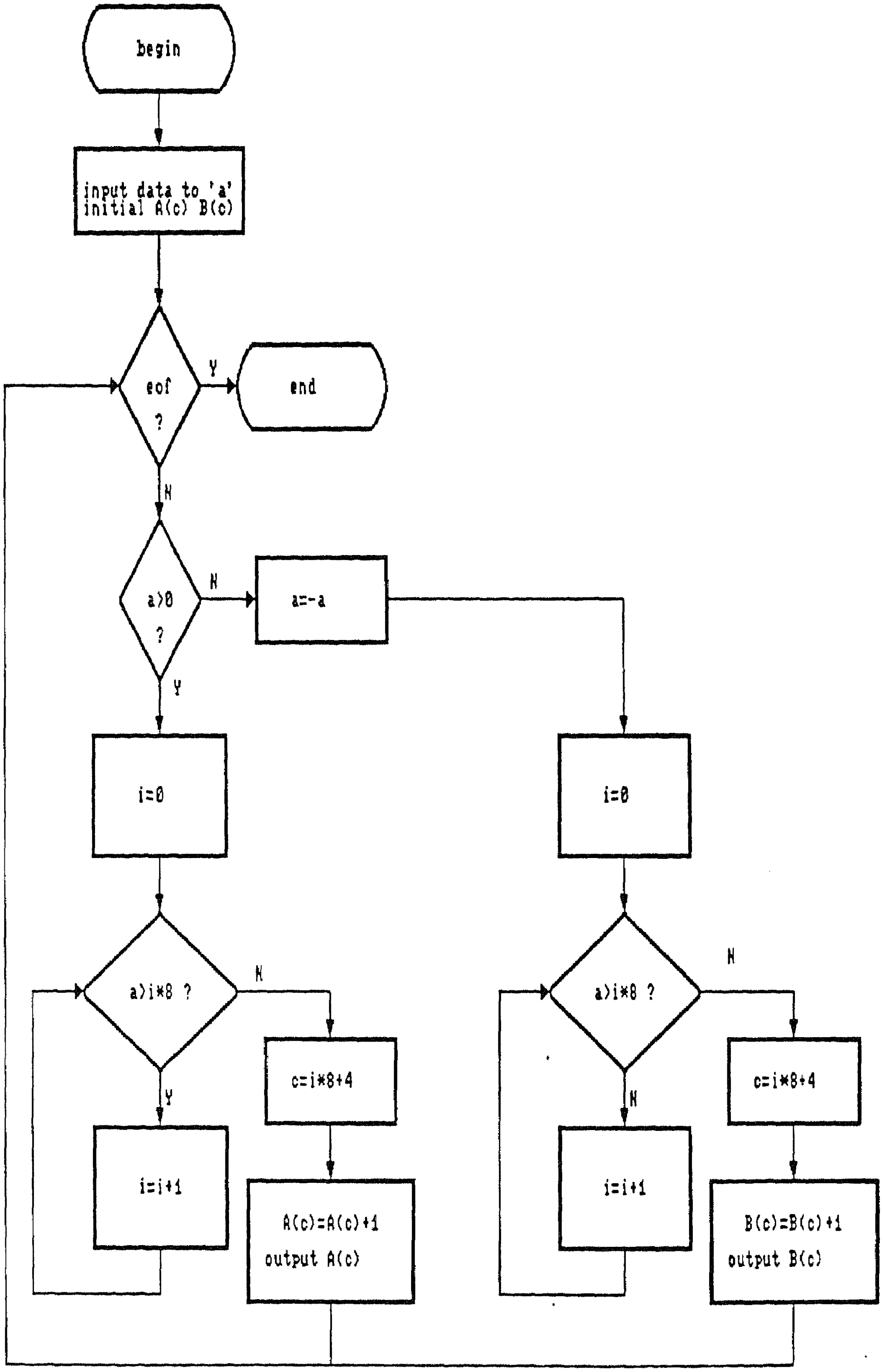


Table B.4: Data combination

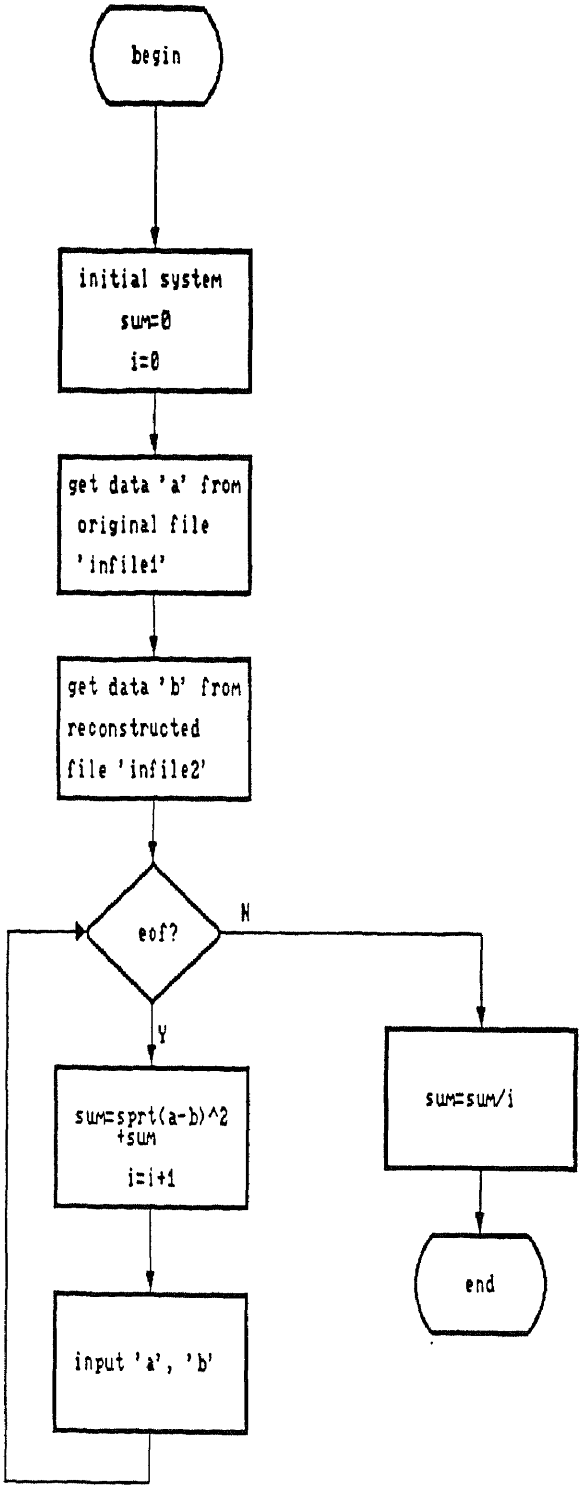


Table B.5: Error calculation

Bibliography

[1] E. Niederrneyer and F. Lopes da Silva, *Electroencephalography Basic Principles, Clinical Applications and Related Fields* Amsterdam:Elsevier. 1987

[2] A.Remond, *Handbook of Electroencephalography and clinical neurophysiology.* Amsterdam:Elsevier. 1975.

[3] B.R.S.Reddy and A.S.N.Murthy," ECG data compression using Fourier descriptors." *IEEE Trans. Biomed. Eng.,* Vol.BME-33, pp.428-434. April 1968.

[4] M.S.Sateh, J.Chriswell, G.Hutchens, R. D.Strattan, W.A.Coberly, "ECG data compression techniques – a unified approach." *IEEE Trans.Biomed. Eng.,* Vol 37. No.4, April 1990.

[5] J.R.Cox, F.M.Nolle, H.A.Fozzard and G.C.Oliver. "AZTEC a preprocessing program for real-time ECG rhythm analysis." *IEEE Trans. Biomed. Eng.,* Vol BME-15. pp.128-129. Apr. 1968.

[6] J.R.Cox, F.M.nolle and R.M.Arthur, "Digital analysis of electroencephalogram, the blood pressure wave and the ECG." *Proc. IEEE,* Vol 60. pp.1137-1164. 1972.

[7] W.J.Tompkins and J.G.Webster, *Design of microcomputer-based medical instrumentation.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[8] J.P.Abenstein and W.J.tompkins. "A new data reduction algorithm for real-time ECG analysis." *IEEE Trans. Biomed. Eng.,* Vol BME-29. pp.43-48. Jan. 1982.

[9] H.K.Wolf, J.Sherwood and P.M.Rantaharju. "Digital transmission of electrocardiograms– A new approach." *Proc. 4th Com. Med. Biol. Conf.,* pp.39a- 39b, 1972.

[10] P.B.Jayakav, E.Brusse, J.P.Patrick, E.Shwedyk and S.S.Seshia, "Computer database of ambulatory EEG signal." *Electroencephalography and Clinical Neurophysiology,* pp.82-88. 1987.

[11] D.Stewart, G.E.Dower and O.Suranyi, "An ECG Compression Code." *J.Electrocard.,* Vol 6. pp.175-176, 1973.

[12] V.Cappellini, "Data compression Application to Biomedical Images." *Data Compression and Error Control Techniques Applications.* Academic Press, LTD, pp.268-272, 1985.

[13] M.Shridhar and N.Mohankrishnan, "Data compression techniques for electrcardiograms," *Can. Elec. Eng.J.,* vol.9, no.4, pp.126-131, 1984.

[14] V.Cappellini, "Some digital techniques for biomedical signal and image processing." *Elektrotehnika, Proc. Symposium Medicine and Techniques, Zagreb, Yugoslavia,* pp.283-285. 1977.

[15] V.Cappellini and A.N.Venetsanopoulous, "Two- dimensional digital filters with applications to biomedical image processing." *Proc. Congresso Brasileiro de Engenhoria Biomedical,* 8th, Florianopolis, Brasil, Nov. 1983.

[16] M.E.Womble, J.S.Halliday, S.K.Mitter, M.C.Lancaster and J.H.Triebwasser. "Data compression for storing and transmitting ECG's/VCG's." *Proc. IEEE Vol 65,* pp.702-716. May, 1977.

[17] W.S.Kuklinski, "Fast Walsh transform data compression algorithm: E.C.G. applications." *Comput. Biol.Medi.* vol.21 pp.465-472. 1983

[18] *Metrobyte Data acquisition and control - - Dash-16 Manual.* Copy-right by MBC 1984.

[19] N.Ahmed and K.R.Rao. "Data compression using orthogonal transforms." *Applications of Walsh Functions and sequency theory.* IEEE Cat. pp.191-209. 1974

[20] N.Ahmed, D.J.Milne and S.G.Harries. "Electrocardiographic data compression via orthogonal transforms." *IEEE Trans. Biomed. Eng.,* Vol BME-22. pp.484-492, Nov. 1975.

[21] R.Dzwonczyk, M.B.Howie and J.S.Mcdonald. "A comparison between Walsh and Fourier analysis of the Electroencephalogram for tracking the effects of Anesthesia." *IEEE Trans. Biomed. Eng.,* Vol BME-31. No.8, August 1984.

[22] R.Steele and C.Eng. "Linear Delta Modulation." *Delta Modulation.* Halsted Press. 1975.

[23] T.J.Lynch "Delta Modulation." *Data Compression Techniques and Application.* pp.94-97. Van Nostrand Reinhold. 1985

[24] D.A.Huffman, "A method for the construction of minimum redundancy codes." *Proc. IRE,* 40(9), pp.1098-1101. 1952.

[25] V.Cappellini "Huffman coding." *Data compression and error control techniques with applications.* Academic Press, Ltd, pp.20-21. 1985.

[26] H.Larsen and D.C.Lai, "Walsh spectral estimates with applications to the classification of EEG signal." *IEEE Trans. Biomed. Eng,.* Vol BME-27. NO.9, Sept. 1980

[27] B.W.Weide and L.T.Andrews and A.M.Iannone. "Real-time analysis of EEG using Walsh transforms." *Comput. Biol. Med.* Vol 8. pp.255-263.

[28] C.E.Shannon and W.Weaver. *The mathematical Theory of Communication.* Urbana. IL: Univ. Illinois Press. 1949.